TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN

ThS. Hoàng Thị Cành

ThS. Nguyễn Hồng Tân

ThS. Phạm Thị Thương

ThS. Nguyễn Thu Phương

TS. Quách Xuân Trưởng

ThS. Nguyễn Thị Dung

BÀI GIẢNG

NHẬP MÔN CÔNG NGHỆ PHẦN MỀM

Tài liệu lưu hành nội bộ

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN

ThS. Hoàng Thị Cành

ThS. Nguyễn Hồng Tân

ThS. Phạm Thị Thương

ThS. Nguyễn Thu Phương

TS. Quách Xuân Trưởng

ThS. Nguyễn Thị Dung

BÀI GIẢNG

NHẬP MÔN CÔNG NGHỆ PHẦN MỀM

Thái Nguyên, tháng 12 năm 2022

Mục lục

Các từ viết tắt

TT	Từ viết tắt	Ý nghĩa của từ
1	SE	Software Engineering – Công nghệ phần mềm
2	OS	Operating System – Hệ điều hành
3	IT	Information Technology – Công nghệ thông tin
4	SQL	Structured Query Language – Ngôn ngữ truy vấn có cấu trúc
5	WWW	WorldWideWeb – Mạng lưới toàn cầu
6	HTTP	HyperText Transfer Protocol – Giao thức truyền tải siêu văn bản
7	HTMP	HyperText Markup Language – Ngôn ngữ đánh dấu siêu văn bản
8	RAD	Rapid Application Development – Phát triển ứng dụng nhanh
9	PHP	Personal Home Page – Ngôn ngữ lập trình kịch bản web
10	DSDM	Dynamic Systems Development Method – Phương pháp phát triển hệ thống động
11	ASP	Active Server Page – Ngôn ngữ kịch bản kích hoạt phía server
12	RUP	Rational Unified Process – Tiến trình hợp nhất Rational
13	XP	Extreme Programming – Lập trình cực đoan
14	DevOps	Development (Dev) & Operations (Ops) Framework – Khung làm việc hợp nhất Dev & Ops
15	SAFe	Scaled Agile Framework - Khung làm việc Scrum với qui mô lớn
16	LeSS	Large-Scale Scrum Framework - Khung làm việc Scrum với qui mô lớn cho nhiều teams
17	JVM	Java Virtual Machine – Máy áo Java
18	AI	Artificial Intelligence – Trí tuệ nhân tạo
19	VR	Virtual Reality – Thực tại ảo
20	AR	Augmented Reality – Thực tế ảo tăng cường
21	IoT	Internet of Things – Vạn vật kết nối
22	RPA	Robotic Process Automation – Tự động hoá quy trình bằng Robot
23	GNP	Gross National Product – Tổng sản lượng quốc gia

24	CASE	Computer-Aided Software Engineering – Công nghệ phần mềm được trợ giúp bởi máy tính
25	IDE	Integrated Development Environment – Môi trường phát triển tích hợp
26	SDLC	Software Development Life Cycle – Vòng đời phát triển phần mềm
27	CS	Computer Science – Ngành khoa học máy tính
28	IS	Information System – Ngành hệ thống thông tin
29	ACM	Association for Computing Machinery – Hiệp hội máy tính
30	IEEE	Institute of Electrical and Electronics Engineers – Hiệp hội kỹ sư điện và điện tử
31	WIPO	World Intellectual Property Organization – Tổ chức sở hữu trí tuệ thế giới
32	WTO	World Trade Organization – Tổ chức thương mại quốc tế
33	KE	Knowledge Engineering – Kỹ sư tri thức
34	DBA	Database Administrator – Người quản trị cơ sở dữ liệu
35	QA	Quality Assurance – Người chịu trách nhiệm đảm bảo chất lượng sản phẩm
36	RE	Requirement Engineering – Kỹ nghệ yêu cầu
37	LYBSYS	Library System – Hệ thống thư viện
38	SRS	Software Requirements Specification Document – Tài liệu đặc tả yêu cầu phần mềm
39	UC	Use Case – Trường hợp sử dụng
40	SUPL	Supplementary Requirements – Yêu cầu bổ sung
41	ATM	Automatic Teller Machine – Máy rút tiền tự động
42	RMP	Requirements Management Plan – Bản kế hoạch quản lý yêu cầu
43	STRQ	Stakeholder Requests – Các yêu cầu từ phía đối tác
44	FEAT	FEATures – Các tính năng của sản phẩm
45	CI/CD	Continuous Integration/Continuous delivery – Tích hợp và phát hành ứng dụng liên tục

46	UML	Unified Modeling Language – Ngôn ngữ mô hình hoá hợp nhất
47	STLC	Software Testing Life Cycle - Vòng đời kiểm thử phần mềm
48	UT	Unit Testing – Kiểm thử đơn vị
49	SOA	Service - Oriented Architecture – Kiến trúc hướng dịch vụ
50	SVN	Subversion - Hệ thống quản lý phần tài nguyên (code, hình ảnh, video…) của một dự án

Một số thuật ngữ

TT	Thuật ngữ	Diễn giải ý nghĩa
1	Bảo trì hoàn thiện	Là những thay đổi liên quan đến việc nâng cấp, hiệu chỉnh các tính năng & khả năng sử dụng của phần mềm bằng cách tinh chỉnh, xóa, thêm các tính năng mới
2	Bảo trì phần mềm	Là việc sửa đổi phần mềm sau khi phát hành nhằm chỉnh sửa các lỗi phát sinh, cải thiện hiệu năng, các thuộc tính của phần mềm hoặc làm cho phần mềm thích ứng với môi trường vận hành mới
3	Bảo trì phòng ngừa	Là những hoạt động phòng ngừa, ngăn chặn sự xuống cấp của phần mềm theo thời gian
4	Bảo trì sửa lỗi	Là hoạt động sửa chữa các lỗi, sai sót và khiếm khuyết trong phần mềm; thường xuất hiện dưới dạng các bản cập nhật nhanh, nhỏ & thường xuyên
5	Bảo trì thích ứng	Là những thay đổi liên quan đến cơ sở hạ tầng phần mềm (hệ điều hành, phần cứng, và nền tảng mới) để giữ cho chương trình tương thích với chúng
6	CASE tools	Là các sản phẩm phần mềm nhằm mục đích hỗ trợ tự động hoặc bán tự động cho các hoạt động trong quy trình phần mềm
7	Công nghệ	Là cách thức hay phương pháp để làm một việc gì đó có thể là cụ thể hoặc trừu tượng, có áp dụng các thành tựu của khoa học và được thực hiện một cách có bài bản, hệ thống
8	Công nghệ phần mềm	Là một chuyên ngành nghiên cứu thuộc lĩnh vực công nghệ thông tin
9	Công nghệ thông tin	Là ngành kỹ thuật sử dụng máy tính và phần mềm máy tính để chuyển đổi, lưu trữ, bảo vệ, xử lý, truyền tải và thu thập thông tin
10	Chuyên gia bảo mật	Là những người chịu trách nhiệm về các vấn đề liên quan đến bảo mật, an toàn, an ninh của hệ thống, khắc phục, sử lý sự cố về bảo mật
11	Chuyên gia hỗ trợ phần mềm	Là những người chịu trách nhiệm cài đặt, bảo dưỡng các gói phần mềm được sử dụng đội dự án và bởi khách hàng

12	Chuyên gia ứng dụng	Là những người chịu trách nhiệm tư vấn cho đội dự án về các loại ứng dụng phần mềm cần sử dụng
13	Chuyên gia mạng cục bộ	Là những người chịu trách nhiệm thiết kế, hướng dẫn, lắp đặt mạng cục bộ; quản lý, duy trì khả năng hoạt động của mạng cục bộ; giám sát tài nguyên cung cấp qua mạng cục bộ, quản lý cấu hình và khắc phục các vấn đề về mạng cục bộ
14	Chuyên gia đảm bảo chất lượng	Là những người chịu trách nhiệm lập kế hoạch đảm bảo chất lượng dự án, chịu trách nhiệm giám sát, đánh giá và quản lý các vấn đề về chất lượng liên quan đến dự án phần mềm
15	Đào tạo viên	Là những người chịu trách nhiệm tìm hiểu, nắm bắt các công nghệ, công cụ, kiến thức mới và đào tạo đội dự án cũng như đào tạo khách hàng sử dụng sản phẩm
16	Hệ thống phần mềm	Là sản phẩm tích hợp nhiều phần mềm cộng thêm yếu tố phần cứng liên quan đến môi trường vận hành
17	Hệ thống thông tin	Là ngành nghiên cứu về các cách thức tổ chức và quản lý các hệ thống thông tin
18	Người hỗ trợ sản phẩm	Là những người chịu trách nhiệm làm việc với nhóm người dùng cuối (cài đặt, thiết lập cấu hình,…); bán hàng; trực đường dây nóng hỗ trợ khách hàng sử dụng sản phẩm; là người có khả năng giao tiếp tốt.
19	Kiểm soát viên	Là những người kiểm tra, giám sát quá trình triển khai dự án so với lịch biểu, lập báo cáo trình lãnh đạo nếu phát hiện vấn đề, rủi ro phát sinh
20	Kỹ nghệ ngược	Là quy trình phân tích hệ thống để xác định các thành phần và mối quan hệ giữa chúng từ đó tạo dạng biểu diễn mới của hệ thống ở mức độ trừu tượng hoá cao hơn
21	Kỹ nghệ yêu cầu	Là quy trình xác định mục tiêu dự án; định nghĩa; lập tài liệu yêu cầu; và quản lý các yêu cầu trong suốt tiến trình dự án
22	Kỹ sư phần mềm	Là những người áp dụng các nguyên tắc của công nghệ phần mềm vào việc thiết kế, phát triển, bảo trì, kiểm thử, và đánh giá phần mềm

23	Kỹ sư trí tuệ nhân tạo	Là những người chịu trách nhiệm cố vấn cho đội dự án xác định, thiết kế và cài đặt trí tuệ vào các ứng dụng. Kỹ sư trí tuệ nhân tạo thường có trình độ chuyên môn về AI cao hơn các kỹ sư tri thức
24	Kỹ sư tri thức	Tương tự như các kỹ sư phần mềm nhưng được chuyên môn hoá các kỹ năng để xây dựng các hệ thống trí tuệ nhân tạo, hệ tri thức
25	Khoa học máy tính	Là ngành nghiên cứu về máy tính và các hệ thống tính toán, quy trình và cách hoạt động của máy tính, cải thiện và nâng cao hiệu suất cho các thuật toán, công nghệ mới, giao tiếp giữa máy tính và con người
26	Chuyên gia lập kế hoạch công nghệ	Là những người chịu trách nhiệm nắm bắt các xu hướng phát triển công nghệ, tư vấn lựa chọn và áp dụng các công nghệ thích hợp cho tổ dự án
27	Lập trình viên	Là những người sử dụng các ngôn ngữ lập trình và công cụ để xây dựng và bảo trì các sản phẩm phần mềm
28	Lập trình viên hệ thống	Là những người chịu trách nhiệm cài đặt, bảo dưỡng các hệ điều hành, các ứng dụng hỗ trợ phần mềm; có thể giám sát hàng trăm ứng dụng vận hành trên nền tảng, theo dõi, khắc phục những sự cố liên quan đến phần mềm hệ thống
29	Mô hình quy trình phần mềm	Là một thể hiện đơn giản của một quy trình phần mềm nhìn từ góc độ cụ thể
30	Người viết các chuẩn & kỹ thuật	Là những người chịu trách nhiệm xây dựng các chính sách, thủ tục chuẩn hoá cho tổ chức phần mềm phù hợp với các quy định chung về nghề nghiệp
31	Nhà phân tích & kỹ sư truyền thông	Là những người chịu trách nhiệm phân tích, thiết kế, đàm phán và lắp đặt các thiết bị và phần mềm truyền thông; có thể làm việc trên mainframe hoặc các mạng truyền thông, kiến thức nền tảng phải có gồm: điện tử, kỹ thuật, các ứng dụng truyền thông, khoa học máy tính và các công nghệ truyền thông.
32	Nhà tư vấn	Là những người có sự hiểu biết rộng về mọi vấn đề, có kiến thức và kinh nghiệm thực hành, thực tế về dự án phần mềm. Số năm kinh nghiệm càng cao thì kiến thức tích luỹ càng nhiều

33	Phát triển yêu cầu	Là một tập các hoạt động nhằm thu thập, phân tích, xác định, đặc tả và thẩm định các yêu cầu phần mềm
34	Phần cứng	Là các thiết bị, cấu kiện mang tính vật lý có thể tiếp xúc bằng tay được như máy in, ổ đĩa, máy quét, các mạch tích hợp, ...
35	Phần mềm	Theo nghĩa đơn giản nhất, phần mềm là tệp dữ liệu và các chương trình chứa các dòng lệnh chỉ thị cho máy tính thực hiện một công việc nào đó
36	Phương pháp SE	Là cách tiếp cận chi tiết, có cấu trúc chỉ ra cách thức giúp phát triển phần mềm một cách dễ dàng, đảm bảo chất lượng và chi phí hiệu quả
37	Quản lý yêu cầu	Là tập các hoạt động nhằm quản lý, thiết lập, duy trì các thương lượng với khách hàng và người dùng về các yêu cầu phần mềm
38	Quản trị dữ liệu	Là những người quản trị cơ sở dữ liệu, phân tích, thiết kế, xây dựng, bảo lưu, quản lý quyền truy cập vào cơ sở dữ liệu của hệ thống
39	Quy trình phần mềm	Là một tập hợp các hoạt động có cấu trúc, được thực hiện theo trình tự nhất định (song song, tuần tự) nhằm xây dựng mới hoặc nâng cấp phần mềm đang tồn tại. Quy trình phần mềm còn được biết với tên gọi "vòng đời phát triển phần mềm" (SDLC)
40	Stakeholder	Là các tổ chức/cá nhân cung cấp yêu cầu cho dự án phần mềm
41	Tái cấu trúc	Là hoạt động chuyển dịch hệ thống từ dạng biểu diễn này sang dạng biểu diễn khác tối ưu hơn
42	Tái kỹ nghệ phần mềm	Là quy trình cải tiến khả năng bảo trì của sản phẩm bằng cách chuyển dịch nó sang dạng biểu diễn mới trong khi vẫn đảm bảo tính nguyên vẹn của các chức năng phần mềm
43	Nhân viên tiếp thị sản phẩm	Là những người chịu trách nhiệm tiếp thị sản phẩm, quảng bá sản phẩm; cần có kỹ năng giao tiếp tốt và các kiến thức về tiếp thị, chăm sóc khách hàng

44	Thẩm định yêu cầu	Là hoạt động kiểm tra đánh giá các tiêu chí chất lượng của từng yêu cầu nhằm phát hiện các vấn đề tiềm ẩn.
45	Thuộc tính của yêu cầu	Là thông tin cần quản lý của yêu cầu
46	Triển khai phần mềm	Là quá trình lắp/cài đặt thêm các tính năng mới hoặc tích hợp phần mềm mới vào môi trường vận hành của người dùng, nơi các ứng dụng và dịch vụ khác của họ đang vận hành
47	Vòng đời kiểm thử phần mềm	Gồm một loạt các hoạt động do các Testers thực hiện theo phương pháp có sẵn để kiểm thử sản phẩm phần mềm có đáp ứng được yêu cầu đề ra hay không
48	Vòng đời phát triển phần mềm	Là một chuỗi các hoạt động được thực hiện bởi các Developers để thiết kế và phát triển thành một phần mềm chất lượng cao
49	Yêu cầu phần mềm	Là một phát biểu/mô tả/đặc tả về dịch vụ mà hệ thống cung cấp, hoặc mô tả về một ràng buộc/điều kiện/phụ thuộc mà hệ thống phải thoả mãn; hoặc cũng có thể là một mục tiêu mà hệ thống phần mềm phải đạt được
50	Yêu cầu chức năng	Là các yêu cầu phản ánh trực tiếp các chức năng của phần mềm
51	Yêu cầu phi chức năng	Là các ràng buộc, điều kiện, phụ thuộc, mục tiêu, thuộc tính chất lượng của phần mềm
52	Yêu cầu miền	Là các yêu cầu được phát biểu gắn với các tri thức/thuật ngữ của miền áp dụng

Mở đầu

Bài giảng Nhập môn Công nghệ phần mềm được tập thể giảng viên thuộc bộ môn Công nghệ phần mềm biên soạn nhằm phục vụ cho việc giảng dạy của giảng viên và học tập của sinh viên Trường Đại học Công nghệ thông tin và Truyền thông - Đại học Thái Nguyên. Tập bài giảng này được biên soạn theo nội dung đề cương chi tiết học phần Nhập môn Công nghệ phần mềm ở trình độ đại học.

Nội dung tài liệu cung cấp cho sinh viên những kiến thức cơ bản liên quan đến các đối tượng chính yếu trong lĩnh vực công nghệ phần mềm như qui trình phát triển phần mềm, lập kế hoạch, quản lý, phân tích, thiết kế, công cụ và môi trường phát triển phần mềm, kiểm thử .... Đồng thời, kiến thức của môn học này làm cơ sở để sinh viên học tốt các học phần chuyên ngành ở các học kỳ tiếp theo.

Để biên soạn tài liệu này, chúng tôi đã tham khảo nhiều cuốn sách được dùng phổ biến trên thế giới về kỹ nghệ phần mềm. Cũng như sử dụng thêm các tài liệu nghiên cứu gần đây để cập nhật các phương pháp và kết quả nghiên cứu mới về lĩnh vực này như được nêu trong phần tài liệu tham khảo ở cuối cuốn tài liệu. Nội dung tài liệu gồm 8 chương:

Chương 1. Tổng quan về công nghệ phần mềm

Chương 2. Quy trình phần mềm

Chương 3. Kỹ nghệ yêu cầu (RE)

Chương 4. Thiết kế phần mềm

Chương 5. Cài đặt phần mềm

Chương 6. Kiểm thử phần mềm

Chương 7. Triển khai và bảo trì phần mềm

Chương 8. Các xu hướng mới trong công nghệ phần mềm

Mặc dù tập thể tác giả đã rất nỗ lực dành nhiều thời gian và công sức để biên soạn, song khó tránh khỏi các thiếu sót. Chúng tôi kính mong quý thầy cô và các bạn sinh viên đóng góp ý kiến để cuốn bài giảng được hoàn thiện hơn. Chúng tôi xin trân trọng cảm ơn.

Thái Nguyên, Tháng 12 năm 2022

Các tác giả

CHƯƠNG I: TỔNG QUAN VỀ CÔNG NGHỆ PHẦN MỀM

Nội dung chính của chương

1.1 Giới thiệu tổng quan về Công nghệ phần mềm (SE)

1.2 Một số khái niệm cơ bản

1.3 Sự khác biệt giữa SE và các lĩnh vực nghiên cứu khác

1.4 Các trách nhiệm đạo đức và nghề nghiệp

1.5 Nhân tố con người và sự phân hóa nghề nghiệp trong SE

Mục tiêu cần đạt được của chương:

- Hiểu được vấn đề của xã hội, kinh tế và môi trường trong ngành công nghệ thông tin, đặc biệt là lĩnh vực công nghệ phần mềm.
- Hiểu được sự tác động của ngành phần mềm đối với xã hội.
- Hiểu được bức tranh tổng quan về SE (từ khi SE hình thành cho đến nay, các thành tựu nghiên cứu đạt được qua từng giai đoạn phát triển). Nắm được các khái niệm cơ bản về SE. Xác định các vị trí công việc khi tham gia vào một dự án phát triển phần mềm và các yêu cầu công việc, trách nhiệm, đạo đức nghề nghiệp, các kỹ năng cần trang bị đối với từng vị trí tương ứng. Từ đó có kế hoạch tự học hỏi, mở rộng các kiến thức chuyên ngành đáp ứng nhu cầu nghề nghiệp trong tương lai.

Bài 1: Tổng quan về Công nghệ phần mềm (Số tiết: 03 tiết)

1.1 Giới thiệu tổng quan về Công nghệ phần mềm (SE)

SE nhìn dưới góc độ lịch sử có thể chia thành các giai đoạn như sau [7]:

1945 - 1965: Nguồn gốc

- **1946:** máy tính điện tử (computer) được phát minh.
- **1947**: Ra mắt ngôn ngữ lập trình bậc thấp Assembly.
- **1948**: Ra mắt phần mềm/chương trình máy tính đầu tiên, được viết bởi Tom Kibum, chạy trên máy tính Manchester Baby, nhằm tính hệ số cao nhất của số nguyên 2^18 = 262,144.
- **1950**: Máy tính được thương mại hoá, phần mềm bắt đầu được đầu tư phát triển.
- **1955s**: Các ngôn ngữ lập trình mới (mức cao hơn - gần con người hơn) xuất hiện: Autocode (1954, bởi các ĐH của Manchester, Cambridge và London); Fortran (1954-1958, bởi John Backus); Lips (1958-1960, bởi John McCarthy), Combol (1959, bởi CODASYL), Basic (1963, được viết bởi ĐH Dartmouth), ….
- **1960**: Thất bại về phát triển phần mềm (chương trình nhỏ, chủ yếu là tính toán chuyên dụng (xử lý số, theo lô); quản lý rời rạc, thiết kế phần mềm chưa được

chú trọng, lập trình tuần tự,..). Phát triển phần mềm gặp nhiều vấn đề ở giai đoạn này như:

- Nhiều dự án chạy vượt ngân sách và tiến độ. Ví dụ: hệ điều hành OS / 360 (là dự án lớn đầu tiên – 1000 lập trình viên) => dự án kéo dài hàng thập kỷ từ những năm 1960, cuối cùng đã tạo ra một trong những hệ thống phần mềm phức tạp nhất vào thời điểm đó; Fred Brooks tuyên bố rằng ông đã mắc sai lầm hàng triệu đô la khi không phát triển một kiến trúc mạch lạc trước khi bắt đầu phát triển hệ thống này.
- Thiệt hại về tài sản: Các lỗi phần mềm có thể gây ra thiệt hại về tài sản. Bảo mật phần mềm kém cho phép tin tặc đánh cắp danh tính, gây tốn kém thời gian, tiền bạc và danh tiếng.
- Sự sống và cái chết: Lỗi phần mềm có thể gây chết người. Một số hệ thống nhúng được sử dụng trong các máy xạ trị đã thất bại thảm hại đến mức chúng sử dụng các liều bức xạ gây chết người bệnh. Nổi tiếng nhất trong số những thất bại này là sự cố Therac-25.
- Peter G. Neumann đã lưu giữ một danh sách về các sự cố và thảm họa của phần mềm đương thời, xem tại [8].

1965 – 1970s: Khủng hoảng PM, SE được đề xướng hình thành

- **1965: Khủng hoảng phần mềm**

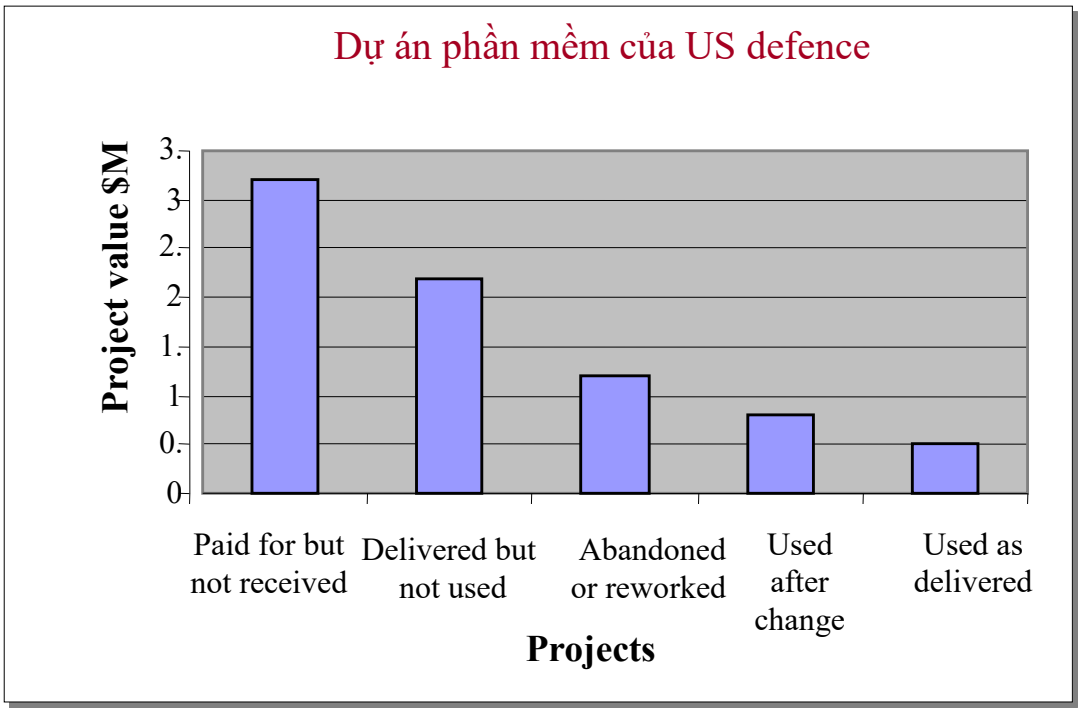
  Nối tiếp các thất bại năm 1960, phát triển phần mềm tiếp tục gặp thất bại:

  - Phát triển phần mềm không bắt cùng nhịp với sự tiến bộ về công nghệ phần cứng.
  - Nhu cầu ứng dụng công nghệ thông tin trong mọi lĩnh vực đời sống, kinh tế, văn hoá xã hội ngày càng tăng, đòi hỏi phần mềm ngày càng nhiều, ngày càng lớn và trở nên phức tạp; bao phủ mọi lĩnh vực kinh tế, đời sống và các hoạt động xã hội, dẫn đến thực trạng: Phần mềm không đáp ứng nhu cầu.
  - Các sản phẩm phần mềm không đáp ứng kịp các yêu cầu của người sử dụng. Phần mềm tiềm ẩn nhiều rủi ro khi sử dụng, nhiều lỗi, chất lượng không đảm bảo (tiêu chí chất lượng giai đoạn này: chạy nhanh, giải được bài toán lớn (dùng bộ nhớ hiệu quả) vì công nghệ sử dụng giai đoạn này là Bóng điện tử (tính toán chậm, bộ nhớ nhỏ)).
  - Dự án kéo dài, thiệt hại về tài sản, lợi nhuận thu được thấp, …

    Cụ thể [30]:

    - 6 dự án triển khai thì 2 dự án thất bại
    - Trung bình thời gian thực hiện bị kéo dài 50% (cá biệt lên tới 200 – 300%)
    - Các dự án lớn dễ bị thất bại
    - 3/4 các hệ thống lớn có lỗi khi thực thi

- Quá trình phân tích yêu cầu (5% công sức): Để lại 55% lỗi, có 18% phát hiện được
- Quá trình thiết kế (25% công sức): Để lại 30% lỗi, có 10% phát hiện được
- Quá trình mã hóa, kiểm thử và bảo trì : Để lại 15% lỗi, có 72% phát hiện được
- Chi cho phát triển lớn, lợi nhuận thấp



Hình 1.1: Chi phí chi cho các dự án phần mềm của Bộ quốc phòng Mỹ (1970)

⇨ ***Dẫn đến tình trạng khủng hoảng phần mềm trên toàn thế giới.***

- **1968: Hội nghị kỹ thuật phần mềm NATO**

Để giải quyết vấn đề trên một hội nghị thế giới đã được tổ chức để bàn về cách giải quyết - Hội nghị kỹ thuật phần mềm NATO bàn về vấn đề khủng hoảng phần mềm lần đầu tiên được tổ chức vào năm 1968. Lần thứ hai diễn ra một năm sau đó. Hội nghị đã thiết lập các hướng dẫn, các thực tiễn tốt nhất để phát triển phần mềm. Michael A.Jackson đã xác định nguyên nhân chính gây ra khủng hoảng phần mềm, đó là việc sản xuất phần mềm thiếu chuyên nghiệp theo phương pháp thủ công. Phương pháp này không thích hợp cho việc phát triển các sản phẩm phần mềm lớn và phức tạp. Phương pháp thủ công thể hiện như sau:

- Làm theo cảm tính: Dựa vào kinh nghiệm, không có phương pháp đủ tốt
- Phương tiện thô sơ: Chủ yếu là ngôn ngữ lập trình bậc thấp
- Làm đơn lẻ: Do một hoặc một số cá nhân thực hiện

Khắc phục khủng hoảng phần mềm:

Xây dựng phần mềm theo công nghệ ~ công nghiệp hóa quá trình sản xuất phần mềm. Từ đó Khái niệm công nghệ phần mềm được đưa ra. Và từ đó công nghệ phần mềm thực sự trở thành một nghành nghiên cứu không thể thiếu được trong lĩnh vực CNTT.

Để đáp ứng đòi hỏi của phát triển phần mềm cần có lý thuyết, kỹ thuật, phương pháp, công cụ đủ tốt để điều khiển tiến trình phát triển hệ thống phần mềm. Công nghệ phần mềm nhằm nghiên cứu tất cả các khía cạnh liên quan đến việc sản xuất các sản phẩm phần mềm chuyên nghiệp. Nó liên quan tới lý thuyết, quy trình, phương pháp và công cụ cần để phát triển phần mềm.

Mục tiêu của công nghệ phần mềm: Sản xuất phần mềm độc lập, đúng hạn, phù hợp kinh phí và đáp ứng mọi yêu cầu người sử dụng.

Để xây dựng hệ thống phần mềm tốt ta cần [5]

- Xác định đúng đắn tiến trình phát triển phần mềm: Các pha của hoạt động, Sản phẩm của mỗi pha
- Phương pháp và kỹ thuật áp dụng trong từng pha và mô hình hóa sản phẩm của chúng
- Công cụ phát sinh ra sản phẩm

- **1970s: Lập trình có cấu trúc/hướng thủ tục**
  - Những năm 1970 là thời kỳ mà SE bắt đầu phát triển khi các ý tưởng, ngôn ngữ và phần cứng mới được giới thiệu. Thành tựu nổi bật nhất đạt được ở giai đoạn này là phương pháp lập trình có cấu trúc ra đời. Phần mềm thường là sản phẩm đa nhiệm, đa người dùng. Các hoạt động xử lý dữ liệu dạng số, ký tự, theo lô và thời gian thực. Giai đoạn này cũng xuất hiện hình thức lưu trữ dữ liệu trực tuyến (cơ sở dữ liệu). Công nghệ bán dẫn được sử dụng (tính toán nhanh hơn, bộ nhớ lớn). Các tiêu chí đánh giá phần mềm giai đoạn này thường là: tính nhanh, giải bài toán lớn, nhiều người dùng, dễ bảo trì sửa lỗi, thích nghi.
  - **1970** – Pascal, một ngôn ngữ lập trình cấu trúc (lệnh cấu trúc, chương trình có cấu trúc, cấu trúc dữ liệu, ..) được giới thiệu. Ngôn ngữ này được thiết kế bởi Nicklaus Wirth.
  - **1970+** - SQL (IBM) được công bố.
  - **1972** - Dennis MacAlistair Ritchie bắt đầu phát triển ngôn ngữ lập trình C. Từ đó, C đã được phát triển và trở thành một trong những ngôn ngữ lập trình phổ biến. Hệ điều hành Unix do Ritchie và Ken Thompson phát triển ra mắt lần đầu tiên (1973).

- **1975** – Máy tính PC đầu tiên xuất hiện (phục vụ chủ yếu cho doanh nghiệp)
- **1975+** - SQL (IBM).

1980s: Kết thúc khủng hoảng phần mềm, phân tích & thiết kế cấu trúc hoàn thiện, LT HĐT ra đời

Những năm 1980 tiếp tục có những thay đổi lớn trong ngành và tình trạng khủng hoảng phần mềm bắt đầu kết thúc. Các ngôn ngữ và công cụ mới bắt đầu xuất hiện hướng tới kỹ thuật tốt hơn – phát triển phần mềm hướng đối tượng.

- **1980** - Ngôn ngữ lập trình Ada ra mắt, được thiết kế bởi Jean Ichbiah
- **1982** - Các công cụ CASE bắt đầu xuất hiện trên thị trường nhằm nâng cao chất lượng của phần mềm, giảm chi phí và thời gian phát triển sản phẩm.
- **1985** - Ngôn ngữ lập trình C ++ (Bjane Strousop) lần đầu tiên được mắt. C++ hỗ trợ cả 2 biểu đồ lập trình hướng đối tượng và hướng thủ tục, C++ liên tục được cập nhật và trở thành một trong các ngôn ngữ được sử dụng phổ biến hiện nay.
- **1987** – Pert được phát triển bởi Larry Wall, là một ngôn ngữ kịch bản Unix đa mục đích để tạo sinh và xử lý các báo cáo dễ dàng hơn.
- **1989** - Các công ty bắt đầu cung cấp quyền truy cập vào internet, quyền truy cập chủ yếu được cấp cho các nhà khoa học và trong quân đội.

1990s: Kỷ nguyên internet; Các quy trình phần mềm phát triển mạnh

Thập kỷ này có nhiều bước tiến lớn trong ngành như: Lập trình hướng đối tượng bắt đầu trở nên phổ biến, Internet ra mắt và phát triển mạnh; một số cách tiếp cận phát triển phần mềm được ra đời:

- **1990** - Tim Berners-Lee phát triển WorldWideWeb - trình duyệt web đầu tiên và tạo ra HTTP, HTML, các trang web đầu tiên. "Dữ liệu lớn – Big Data" bắt đầu phát triển.
- **1991** - Ngôn ngữ lập trình Python ra mắt lần đầu tiên, và hiện nay đã trở thành một trong những ngôn ngữ lập trình phổ biến nhất. Phát triển ứng dụng nhanh (RAD) ra đời.
- **1994**: PHP (Danish-Canadian) ra mắt, là một ngôn ngữ kịch bản đa mục tiêu, hướng tới phát triển các website động. Phương pháp phát triển hệ thống động (DSDM) hình thành.
- **1995** - Ngôn ngữ lập trình Java (James Gosling) được phát hành. Java là ngôn ngữ được sử dụng phổ biến nhất với tên gọi "*Viết một lần, chạy mọi nơi*". JavaScript ra đời và được sử dụng phổ biến trong lập trình web. Mô hình phần mềm Scrum ra mắt.
- **1996** - Học viện Công nghệ Rochester giới thiệu chương trình cử nhân đầu tiên về kỹ thuật phần mềm.

- **1997 –** C#, VB.Net & ASP.Net (Microsoft) được trình diện trong thế giới lập trình.
- **1998 -** Trường Sau đại học Hải quân Hoa Kỳ cung cấp chương trình tiến sĩ đầu tiên về kỹ thuật phần mềm. RUP (IBM, 1998)
- **1999 -** Kent Beck giới thiệu XP, một kiểu phát triển phần mềm linh hoạt (Agile) được thiết kế để đáp ứng nhanh chóng các yêu cầu thay đổi của người dùng.

2000s: Các quy trình PM linh hoạt, gọn nhẹ (Light Weight Methodologies)

Thiết kế với các ngôn ngữ không còn được chú trọng, giai đoạn này tập trung vào: (1) hoàn thiện & cải tiến những gì đã đạt được trong hai thập kỷ trước; (2) xây dựng các quy trình phần mềm linh hoạt, gọn nhẹ, đáp ứng tốt hơn các yêu cầu thay đổi của khách hàng, tăng hiệu xuất và tính tiện ích của quy trình. Lập trình trí tuệ nhân tạo bắt đầu được quan tâm.

- **2001** - Tuyên ngôn phát triển phần mềm Agile được xuất bản. Nó bao gồm các giá trị và nguyên tắc của phát triển phần mềm Agile, tập trung vào việc phát triển thông qua nỗ lực hợp tác của các nhóm chức năng chéo và khách hàng.
- **2006 –** Scrum bản đầy đủ bắt đầu được sử dụng phổ biến. Scrum - một quy trình nhanh sử dụng khuôn khổ lặp đi lặp lại và gia tăng để phát triển phần mềm phức tạp, được giới thiệu bởi Ken Schwaber và Mike Beedle. Phương pháp này đã xuất hiện từ năm 1995 nhưng chỉ bắt đầu được sử dụng rộng rãi ở những năm 2000.
- **2019:** Ngôn ngữ lập trình GO ra mắt, được thiết kế bởi các kỹ sư của Google nhằm tạo ra phần mềm đáng tin cậy và hiệu quả sử dụng cách tiếp cận phát triển gọn nhẹ. Go cũng sử dụng một tập hợp các gói để quản lý phụ thuộc một các hiệu quả. Rất nhiều các tổ chức sử dụng Go như Google, Cloudflare, Dropbox, MongoDB, Netflix, SoundCloud, Twitch và Uber.

2010s: Điện toán đám mây; Cách thức training; Lập trình mobile đa nền tảng

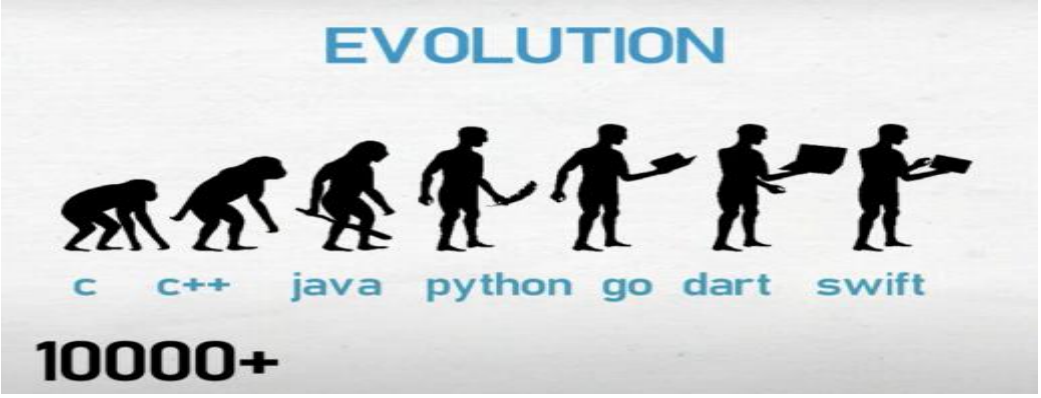
Trọng tâm của giai đoạn này hướng tới công nghệ điện toán đám mây và phát triển phần mềm hướng dịch vụ. Việc giải quyết nhu cầu về nguồn lực kỹ sư phần mềm với phong cách học tập mới cũng được chú trọng. Các quy trình phần mềm linh hoạt tiếp tục được phát triển như SAFe; LeSS (Large-Scale Scrum) & DevOps Framework. Các ngôn ngữ lập trình đa nền tảng, lập trình mobile ra đời và phát triển mạnh mẽ.

- **2010 -** Điện toán đám mây bắt đầu phát triển, dẫn đến nhu cầu về phần mềm dưới dạng dịch vụ tăng và tạo ra hướng đi mới cho các kỹ sư SE.
- **2011 -** Các bootcamps mã hóa bắt đầu phát triển. Trong vòng chưa đầy 8 năm, khoảng 95 bootcamps được giới thiệu. Bootcamps là một cách để giảng dạy công nghệ mới nhất theo một chương trình chuyên sâu được thiết kế để giúp sinh viên đáp ứng nhu cầu nghề nghiệp.

- Các ngôn ngữ lập trình đa nền tảng, lập trình mobile ra đời và phát triển mạnh mẽ:

- **Kotlin (2011)** ngôn ngữ lập trình mới cho JVM, đa nền tảng, hiện đại, được thiết kế bởi công ty phát triển phần mềm JetBrains, giúp phát triển các ứng dụng di động Android nhanh hơn, hiệu quả hơn, mã nguồn an toàn, khách hàng hài lòng hơn. Kotlin đã được gần 60% các nhà phát triển Android chuyên nghiệp sử dụng.
- **Dart (2013)** ngôn ngữ mã nguồn mở được phát triển bởi Google, là một ngôn ngữ tối ưu client hỗ trợ tạo các ứng dụng chạy nhanh trên mọi nền tảng (mobile, desktop, và Web platform, …).
- **Swift (2014, Chris Lattner et al,.)** được giới thiệu tại hội thảo của Apple năm 2014. Swift hỗ trợ nhiều biểu đồ lập trình, là ngôn ngữ lập trình trực quan, tiềm lực cho các hệ điều hành iOS, iPadOS, macOS, tvOS, and watchOS.

Chúng ta có tất cả hơn 10.000 ngôn ngữ lập trình khác nhau đã được hình thành và phát triển đến nay, và chúng ta vẫn tiếp tục đếm! Hình 1.2 chỉ ra sự tiến hoá của các ngôn ngữ lập trình theo thời gian.



Hình 1.2: Sự tiến hoá của một số ngôn ngữ lập trình

2020s: Xu hướng các công nghệ mới [9]

Thời kỳ này, trọng tâm hướng tới phát triển các công nghệ như trí tuệ nhân tạo (AI), thực tế ảo (VR), thực tế ảo tăng cường (AR), và các công nghệ khác, …

Công nghệ Internet of things (IOT): Những năm 2020, theo thống kê, số lượng người dùng các thiết bị như smartphone, laptop, TV thông minh, PC,… có thể lên tới 21 tỷ. IOT hướng tới việc xây dựng các hệ thống bảo mật, an ninh kết nối, hình thành và xử lý dữ liệu lớn (big data), tăng cường khả năng tự động hóa, phân tích số liệu lớn và lập trình điều khiển luồng dữ liệu thời gian thực, …

Công nghệ xác thực không cần mật khẩu: Mật khẩu đã được sử dụng để xác thực trong một thời gian quá dài. Vì vậy theo các chuyên gia bảo mật, mật khẩu không còn được xem là cách xác thực "rất an toàn" nữa. Ngày nay, xác thực không mật khẩu đang dần trở nên phổ biến, ví dụ: xác thực thông qua nhận dạng khuôn mặt, sinh trắc học, …

Thực tại ảo: Thực tế ảo (Virtual Reality – VR) và thực tế ảo tăng cường (Augmented Reality – AR) đang ngày càng trở nên phổ biến. Ngày nay VR & AR không chỉ tập trung vào lĩnh vực giải trí và phát triển ứng dụng game, trên thực tế, lực lượng quốc phòng của một số quốc gia cũng đang nghiên cứu và cố gắng tận dụng tối đa hai công nghệ AR và VR cho việc huấn luyện quân sự.

Tự động hóa quy trình bằng robot: Tự động hóa quy trình bằng robot (RPA) nhằm tăng năng xuất hoạt động, giảm bớt những công đoạn thủ công trong quy trình làm việc. RPA hướng tới lập trình tự động hóa hoàn toàn quy trình làm việc/sản xuất bằng robot cho các tổ chức, doanh nghiệp.

Công nghệ AI: Công nghệ AI & xử lý Big Data đã và đang được áp dụng trong nhiều doanh nghiệp, mang lại những lợi ích vượt trội. Ngoài ra, các công nghệ hiện đại như tìm kiếm bằng giọng nói, nhận dạng giọng nói, … cũng đang dần trở nên phổ biến.

1.2 Một số khái niệm cơ bản

1.2.1 Phần cứng

Là các thiết bị, cấu kiện mang tính vật lý có thể tiếp xúc bằng tay được như máy in, ổ đĩa, máy quét, các mạch tích hợp, ...

1.2.2 Phần mềm

Nghĩa đơn giản nhất của phần mềm là chương trình chứa các dòng lệnh chỉ thị cho máy tính thực hiện một công việc nào đó [1].

Theo nghĩa hoàn thiện hơn, phần mềm được xem là một sản phẩm bao gồm ba phần:

(1) Các chương trình máy tính đơn lẻ:
- *Các file mã nguồn*
- *Các file mã máy (file text): mã nguồn sau khi dịch và đóng gói.*

(2) Dữ liệu:
- *Cấu trúc làm việc (bộ nhớ trong): các cấu trúc dữ liệu*
- *Cấu trúc lưu trữ (bộ nhớ ngoài): cơ sở dữ liệu/file dữ liệu*

(3) Các tài liệu liên quan:
- *Tài liệu hướng dẫn sử dụng (dành cho người dùng cuối)*
- *Tài liệu tham khảo kỹ thuật (dành cho người bảo trì phần mềm)*
- *Tài liệu phát triển (dành cho nhà phát triển)*

Ngoài ra cần một Website cung cấp các thông tin về sản phẩm phần mềm & các bản cập nhật mới nhất về phần mềm.

Đặc trưng của phần mềm:

- Không mòn cũ nhưng thoái hoá theo thời gian
- Khó lắp ráp từ các mẫu có sẵn

- Phức tạp, khó hiểu, vô hình (để hiểu phải tư duy, hình dung vì không nhìn thấy)
- Thay đổi là bản chất của phần mềm (khác sản phẩm khác, thay đổi để tạo phiên bản mới phải thực hiện trên chính phiên bản hiện tại)
- Phát triển theo nhóm.

Vai trò của phần mềm:

Các phần mềm máy tính ngày nay đóng vai trò quan trọng trong mọi lĩnh vực đời sống, kinh tế, xã hội của mọi quốc gia trên thế giới. Cụ thể:

Các phần mềm là "linh hồn" của các hệ thống máy tính. Chúng đóng vai trò nền tảng cho mọi hoạt động trong đời sống, kinh tế, xã hội, và các hoạt động vui chơi giải trí.

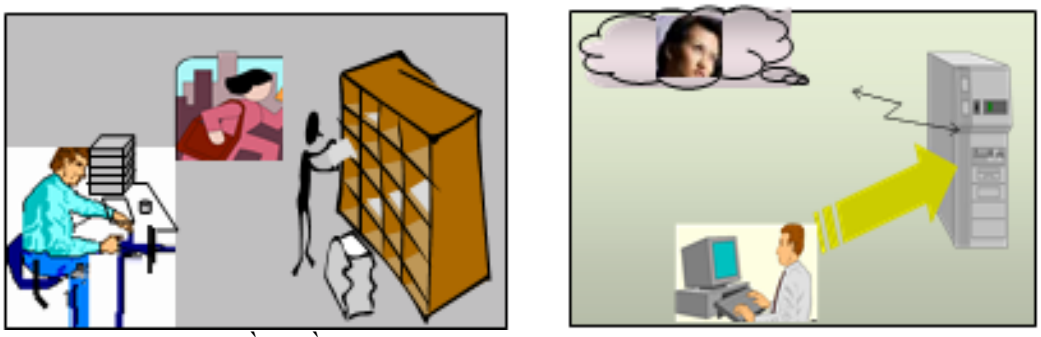
Hình 1.3: Một số hình ảnh minh hoạ vai trò của phần mềm

Nền kinh tế của các quốc gia phụ thuộc rất nhiều vào các sản phẩm phần mềm. Ví dụ[1]:

- Thu, chi từ phần mềm chiếm đáng kể trong tổng GNP của mọi quốc gia:
- 2006 ấn độ xuất gần 30 tỉ USD phần mềm
- Thế giới có >7 triệu kỹ sư IT tạo ra 600 tỉ $/năm
- Chi phí cho phần mềm năm 2000 lên tới 770 tỉ $
- Phần mềm sai hỏng, kinh tế tổn thất lớn:
- Vệ tinh Ariane 5 hỏng do lỗi phần mềm (1996) thiệt hại 500 triệu $.
- Các website dừng hoạt động 1 ngày có thể gây mất hàng triệu $.

Phần mềm tạo nên sự khác biệt giữa các tổ chức, cá nhân về phong cách làm việc và năng xuất lao động (Hình 1.4). Càng ngày càng nhiều nghiệp vụ được phần mềm điều khiển, trợ giúp nhằm tăng năng xuất lao động và sự hiệu quả trong công việc. Ví dụ: phần mềm trợ giúp bán hàng tại các siêu thị, hỗ trợ giáo dục, đào tạo trực tuyến, thảo luận, hội nghị trực tuyến, tự động hoá dây chuyền sản xuất, …

[1] [Pankaj Jalote. CMM in Practices, Addison – Wesley, tr.1, 11]

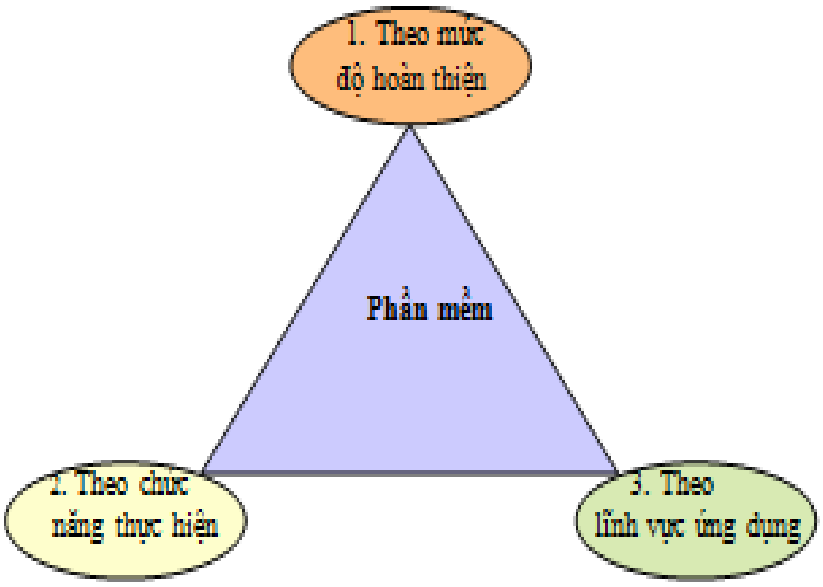


Hình 1.4: Phần mềm tạo nên sự khác biệt trong phong cách làm việc

Phân loại phần mềm:

Phần mềm có thể được phân loại theo 3 tiêu chí (Hình 1.5):

- Theo mức độ hoàn thiện của sản phẩm
- Theo chức năng thực hiện của sản phẩm
- Theo lĩnh vực nghiệp vụ mà phần mềm hỗ trợ



Hình 1.5: Các tiêu chí phân loại phần mềm

1) Phân loại theo mức độ hoàn thiện

Phần mềm được phân loại theo mức độ hoàn thiện tăng dần: (i) phần mềm là chương trình; (ii) phần mềm là sản phẩm; và (iii) phần mềm là hệ thống. Tính phức tạp của phần mềm cũng tăng nhanh (9 lần) theo mức độ hoàn thiện: từ chương trình → sản phẩm → hệ thống (Hình 1.6).

❖ **Chương trình**

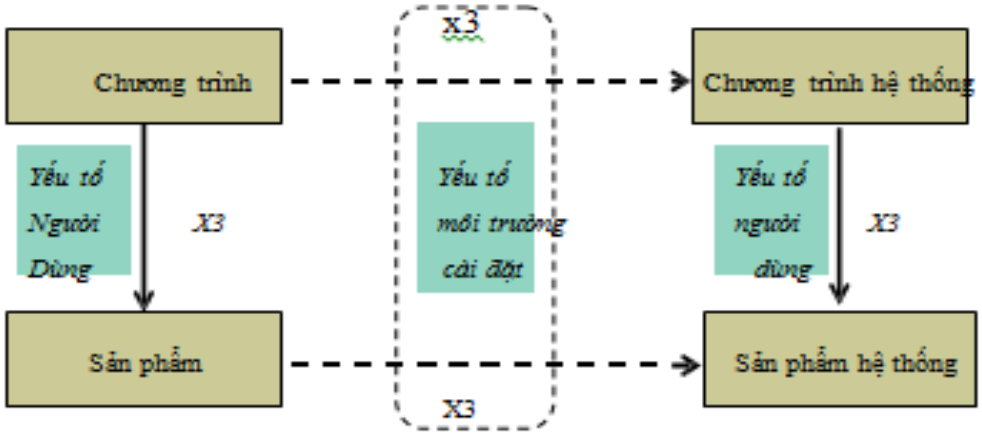
- Người viết chương trình đồng thời cũng là người dùng chương trình đó.
- Thường không có tài liệu đính kèm và không kiểm thử đầy đủ.

❖ **Sản phẩm**

- Nhiều người viết, nhiều người dùng,
- Cần có các tài liệu đính kèm. Cần được kiểm thử đầy đủ trước khi bàn giao đến người dùng

❖ **Hệ thống**

- Tích hợp nhiều phần mềm cộng thêm yếu tố phần cứng liên quan đến môi trường vận hành
- Thường lớn, độ phức tạp cao, đầy đủ tài liệu và phải được kiểm thử đầy đủ trước khi đưa vào thực tế vận hành.



Hình 1.6: Độ phức tạp phần mềm tăng theo mức độ hoàn thiện

2) *Phân loại theo chức năng thực hiện*

Tuỳ theo các chức năng mà phần mềm cung cấp, chúng ta có thể phân chia phần mềm thành các loại như sau:

❖ **Phần mềm hệ thống**: Thường cung cấp các chức năng:

- Điều hành hoạt động máy tính, thiết bị và chương trình (OS, ...)
- Cung cấp các tiện ích (tổ chức tệp tin, nén, dọn ổ đĩa, ...).

❖ **Phần mềm nghiệp vụ**: Thường cung cấp các chức năng:

- Hỗ trợ các hoạt động nghiệp vụ, ví dụ: quản lý thư viện, quản lý điểm, quản lý nhân sự, …
- Có số lượng lớn, đa dạng và thường được chia làm 2 loại theo cách thức phát triển:
  - ***Sản phẩm đặt hàng***: sản xuất theo đơn đặt hàng; đáp ứng các yêu cầu đặc thù riêng của khách hàng.
  - ***Sản phẩm chung*** (sản phẩm đại trà, thương mại): được phát triển để bán rộng rãi trên thị trường; cần xây dựng đảm bảo đáp ứng các yêu cầu chung của thị trường người dùng.

Sự phân loại này ngày càng trở nên mờ nhạt, vì nhiều công ty phần mềm hiện này thường bắt đầu từ các sản phẩm chung (dòng sản phẩm) và tùy biến chúng theo các yêu cầu của khách hành riêng lẻ để trở thành sản phẩm theo đơn đặt hàng.

❖ **Phần mềm công cụ (CASE tools)**: Thường cung cấp các tính năng hỗ trợ cho các hoạt động trong SE. Ví dụ: các phần mềm ngôn ngữ lập trình; trình biên dịch; gỡ rối; hỗ trợ phân tích, thiết kế; quản lý dự án; kiểm thử; và các môi trường tích hợp phát triển ứng dụng (IDE) như Netbean, Eclipse, Visual Studio, …

3) *Phân loại theo lĩnh vực ứng dụng*

Phân loại theo lĩnh vực nghiệp vụ mà phần mềm hỗ trợ, chúng ta có thể chia phần mềm thành các loại như sau:

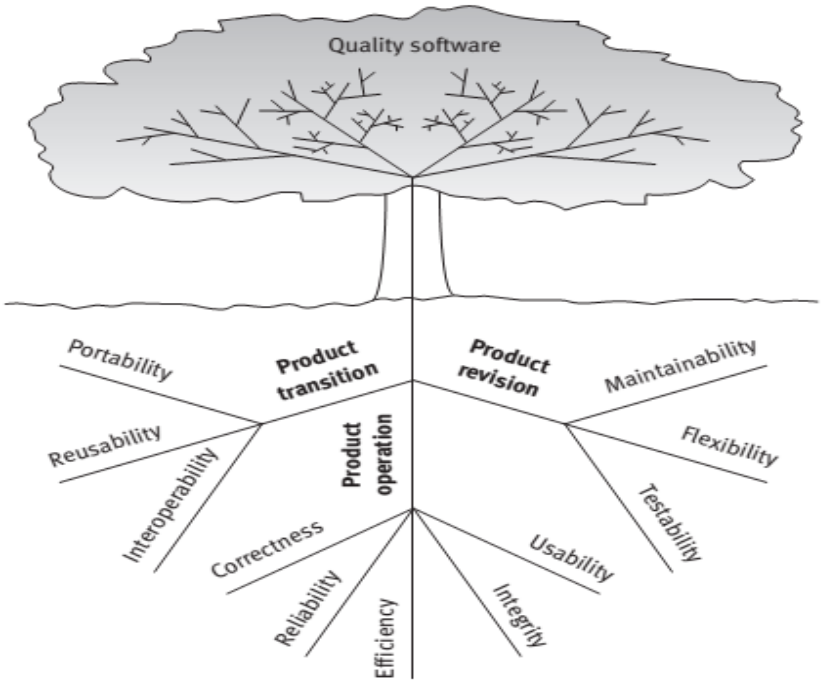
- **Phần mềm hệ thống:** điều khiển các thiết bị phần cứng; làm nền tảng/môi trường để vận hành các phần mềm khác. Ví dụ: các hệ điều hành, các tiện ích quản lý tệp tin, dọn ổ đĩa, …
- **Phần mềm thời gian thực:** thu thập, xử lý các dữ liệu thời gian thực. Ví dụ: phần mềm truyền phát tín hiệu vệ tinh, đo nhiệt độ, chất lượng không khí, …
- **Phần mềm nghiệp vụ:** xử lý các thông tin nghiệp vụ, hỗ trợ kinh doanh, thương mại, …
- **Phần mềm khoa học kỹ thuật:** Mô phỏng các hiện tượng vật lý, hoá học, phản ứng hạt nhân, … .Thường dùng trong các ngành khoa học, phục vụ nghiên cứu, thí nghiệm.
- **Phần mềm nhúng:** còn được xem là một hệ lập trình chuyên biệt được lắp đặt trên các thiết bị phần cứng cố định để điều khiển, xử lý các tác vụ cụ thể trên thiết bị. Ví dụ: hệ thống giao thông thông minh, giải pháp cổng IoT, máy móc, thiết bị thông minh như máy ảnh kỹ thuật số, lò vi ba, máy photocopy, máy in Laser, máy FAX, máy giặt, các bảng quảng cáo sử dụng hệ thống đèn LED….

Ngoài ba tiêu chí phân loại trên đây, phần mềm còn có thể được phân loại dựa trên các chính sách phát triển của sản phẩm như: *phần mềm tự do nguồn mở; phần mềm đóng* (được đóng gói, có bản quyền chống vi phạm, …).

Tiêu chí đánh giá chất lượng phần mềm:

Chất lượng phần mềm đóng vai trò quan trọng trong sự thành công của Công nghệ phần mềm. Mô hình chất lượng phần mềm đầu tiên được đề xuất bởi McCall (1977) gồm 11 yếu tố và được nhóm thành 3 loại tập trung vào 3 khía cạnh quan trọng của phần mềm tạo lên cây chất lượng/tam giác chất lượng (xem Hình 1.7):

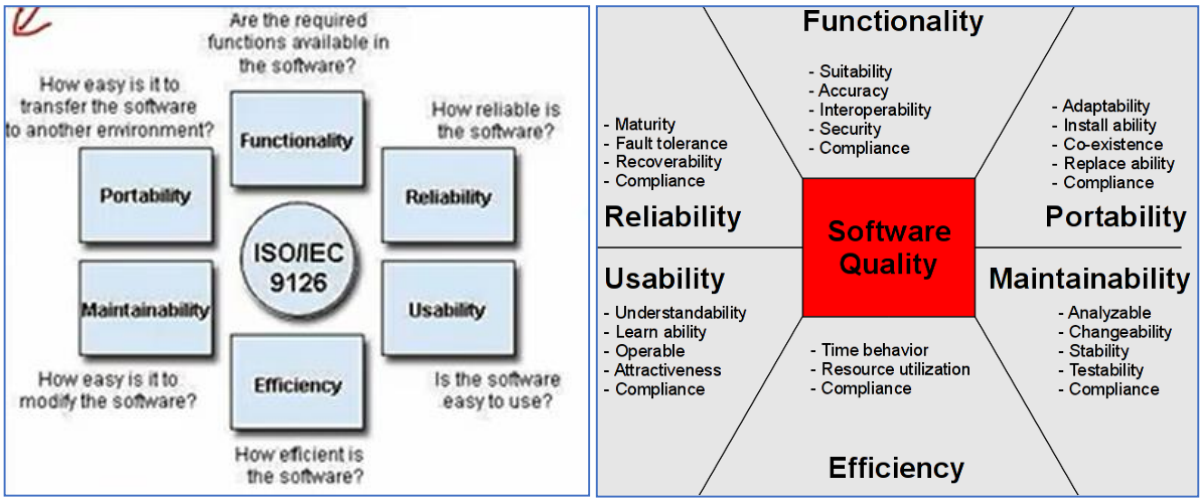
(1) Các đặc tính vận hành của phần mềm (Product Operation) ;
(2) Khả năng chịu được sự thay đổi của phần mềm (Product Revision); và
(3) Khả năng thích nghi với môi trường mới (Product Transition).



Hình 1.7: Mô hình yếu tố chất lượng của McCall (1977)

Thời kỳ này, các tiêu chí đánh giá chất lượng phần mềm chưa được chuẩn hoá. Tiếp theo mô hình chất lượng của McCall đã xuất hiện nhiều mô hình chất lượng biến thể khác mở rộng từ mô hình McCall's như của Evans&Marciniak's; Deutsc&Willis's, …

Mô hình chất lượng được chuẩn hoá đầu tiên xuất hiện trong bộ chuẩn IEEE 1061. Ngày nay, để đánh giá chất lượng phần mềm, chúng ta thường sử dụng các tiêu chí được mô tả trong Bộ chuẩn ISO/IEC 9126. Đây là bộ chuẩn mô tả các thuộc tính chất lượng chính của phần mềm cần đạt được như mô tả trong Hình 1.8.



Hình 1.8: Các yếu tố chất lượng phần mềm trong ISO/IEC 1926

Lưu ý: SV có thể tham khảo các chuẩn IEEE về SE tại link [10]

1.2.3 Công nghệ

Công nghệ là cách thức hay phương pháp để làm một việc gì đó có thể là cụ thể hoặc trừu tượng, có áp dụng các thành tựu của khoa học và được thực hiện một cách có bài bản, hệ thống.

Mọi công nghệ đều thường đề cập đến việc sản xuất một sản phẩm theo quy trình. Một quy trình (process) thường quy định: Ai (Who); Làm gì (What); Làm khi nào (When); và Làm như thế nào (How) để đạt tới mục đích mong muốn.

1.2.4 Công nghệ phần mềm

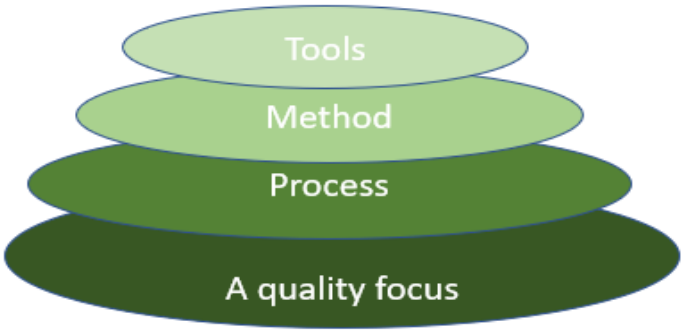
Công nghệ phần mềm là một công nghệ liên quan đến tất cả các khía cạnh của quy trình phát triển một sản phẩm phần mềm như: sản xuất ntn? đánh giá, đảm bảo chất lượng, quản lý ra sao? ....

Quy trình phần mềm có thể là quy trình phát triển mới sản phẩm hoặc quy trình nâng cấp một phần mềm đang tồn tại. Nó mô tả đầy đủ các hoạt động cần thiết để chuyển đổi phần mềm từ mức khái niệm (tập yêu cầu khách hàng) đến mức vận hành (hệ thống phần mềm hoàn chỉnh, đóng gói).

Ngoài cách thức định nghĩa SE qua khái niệm Công nghệ, SE còn có thể định nghĩa như sau:

Theo định nghĩa của Fritz Bauer: "*Công nghệ phần mềm là sự thiết lập và sử dụng các nguyên tắc khoa học nhằm mục đích tạo ra các sản phẩm phần mềm một cách kinh tế mà các sản phẩm phần mềm lại hoạt động một cách hiệu quả và tin cậy trên các máy tính".*

Theo quan điểm của các nhà nghiên cứu, SE ngày nay có thể xem như một mô hình gồm 4 tầng (xem Hình 1.9)*: "SE có thể xem như một mô hình gồm 4 tầng: tầng quy trình, tầng phương pháp, tầng công cụ và tầng chất lượng hướng tới mục tiêu xây dựng các sản phẩm phần mềm đáp ứng các yêu cầu người dùng, đảm bảo chất lượng, không vượt quá kinh phí, thời hạn cho trước".*



Hình 1.9: Mô hình Công nghệ phần mềm nhìn dưới góc độ nghiên cứu

- ✓ ***Tầng chất lượng (Quality)*:** Liên quan đến các nguyên tắc về đảm bảo chất lượng phần mềm. Cung cấp các cơ chế đảm bảo tính an toàn, an ninh của sản phẩm. Tầng này cũng tập trung vào những vấn đề liên quan đến khả năng bảo trì và khả năng sử dụng của sản phẩm.
- ✓ ***Tầng quy trình (Process***): Liên quan tới các vấn đề về quản trị phát triển phần mềm như lập kế hoạch, quản trị chất lượng, tiến độ, chi phí, mua bán sản phẩm phụ, cấu hình phần mềm; quản trị sự thay đổi; quản trị nhân sự (trong môi trường làm việc nhóm); chuyển giao, đào tạo, và xây dựng các tài liệu cần thiết;
- ✓ ***Tầng phương pháp (Method):*** Tập trung vào việc phát triển các cách thức, công nghệ, kỹ thuật cụ thể cho các công đoạn phát triển và bảo trì phần mềm. Các phương pháp xây dựng cần dựa trên những nguyên lý chung áp dụng cho tất cả các lĩnh vực công nghệ khác, kể cả các hoạt động mô hình hoá và kỹ thuật mô tả.
- ✓ ***Tầng công cụ (Tool):*** Tập trung xây dựng các công cụ, phương tiện hỗ trợ tự động hoặc bán tự động cho các tầng còn lại của mô hình (tầng phương pháp, tâng quy trình và tầng đảm bảo chất lượng).

Lưu ý: Từ mô hình trên ta thấy, SE không chỉ đề cập đến từng tầng riêng lẻ mà còn tới cả cách thức phối hợp các tầng, phối hợp các công nghệ, phương pháp và công cụ theo một quy trình nghiêm ngặt để tạo ra các sản phẩm phần mềm có chất lượng, đáp ứng yêu cầu, không vượt quá kinh phí, thời hạn cho trước.

1.2.5 Quy trình phần mềm, vòng đời phần mềm

Quy trình phần mềm là một tập hợp các hoạt động có cấu trúc, được thực hiện theo trình tự nhất định (song song, tuần tự) nhằm xây dựng mới hoặc nâng cấp phần mềm đang tồn tại. Quy trình phần mềm còn được biết với tên gọi "vòng đời phát triển phần mềm" (SDLC) [4].

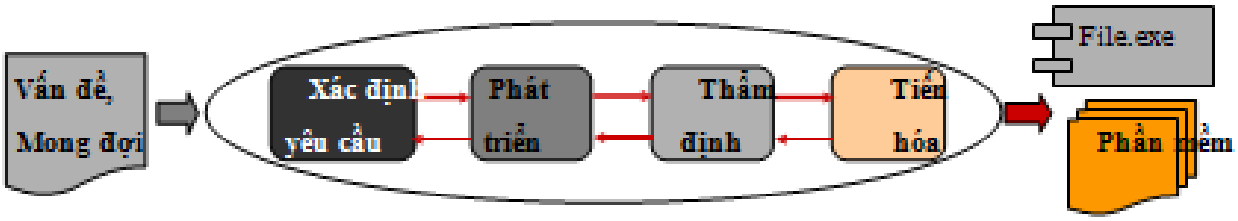
Một quy trình cụ thể phải trả lời được câu hỏi: *Làm gì? Khi nào làm? Ai làm? Làm như thế nào? Làm bằng gì? Ở đâu? Kết quả? Tiêu chí đánh giá?*

Như vậy, mỗi quy trình phần mềm thường có các đặc trưng: (1) Có cấu trúc nhất định (công việc gì, trình tự ra sao, sử dụng phương pháp, công cụ gì, ai làm); (2) Gắn với dự án cụ thể; & (3) Các thành phẩm & sản phẩm cuối cùng cần xây dựng và bàn giao.

Một quy trình phần mềm thường chia thành các giai đoạn chính như sau (xem Hình 1.10):

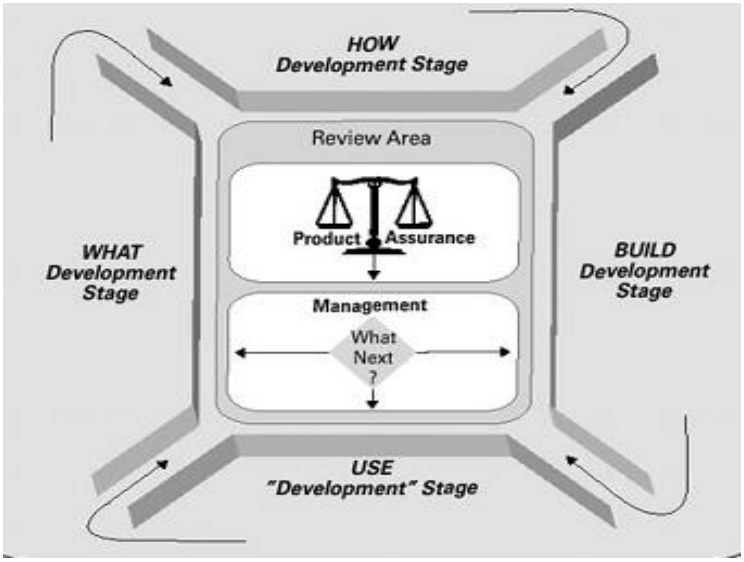
- ❖ *Phân tích, xác định yêu cầu:* Thu thập, hiểu yêu cầu từ phía đối tác khách hàng
- ❖ *Phát triển*: Thiết kế và cài đặt chương trình
- ❖ *Thẩm định*: kiểm thử đánh giá xem phần mềm có đáp ứng các yêu cầu của khách hàng không, có thoả mãn các tiêu chí chất lượng không.

❖ *Tiến hóa phần mềm*: Thay đổi phần mềm để đáp ứng các yêu cầu thay đổi (người dùng, môi trường).



Hình 1.10: Các giai đoạn trong quy trình phần mềm

Cũng có thể nhìn quy trình phần mềm dưới góc độ vòng đời phần mềm gồm 4 giai đoạn: WHAT, HOW, BUILD, USE và các nhóm nguyên tắc bổ sung gồm: Đảm bảo chất lượng, Quản lý dự án như mô tả trong Hình 1.11.



Hình 1.11: Các giai đoạn trong vòng đời phần mềm & các hoạt động quản lý và đảm bảo chất lượng

Lưu ý:

Những loại phần mềm khác nhau sẽ cần những quy trình phát triển khác nhau. Ví dụ, hệ thống thời gian thực yêu cầu phải xây dựng bản đặc tả hệ thống trước khi chuyển sang giai đoạn xây dựng. Tuy nhiên, với phần mềm thương mại điện tử, cho phép người phát triển vừa đặc tả vừa xây dựng chương trình một cách đồng thời. phần mềm hiện nay cũng có thể được tạo bằng cách mở rộng hoặc sửa đổi từ hệ thống đang tồn tại hoặc bằng cách cấu hình và tích hợp các thành phần có sẵn của các hệ thống hoặc phần mềm khác.

=> Lựa chọn sử dụng quy trình phù hợp sẽ mang lại những lợi ích nhất định cho việc triển khai dự án phần mềm cụ thể.

1.2.6 Mô hình quy trình phần mềm

Mô hình quy trình phần mềm là một thể hiện đơn giản của một quy trình phần mềm nhìn từ góc độ cụ thể. Ví dụ: mô hình thác nước, mô hình mẫu thử, mô hình xoắn ốc, mô hình RUP, RAD, …

1.2.7 Phương pháp Công nghệ phần mềm

Phương pháp công nghệ phần mềm là cách tiếp cận chi tiết, có cấu trúc chỉ ra cách thức giúp phát triển phần mềm một cách dễ dàng, đảm bảo chất lượng và chi phí hiệu quả. Mỗi phương pháp thường bao gồm:

- Các mô hình hệ thống,
- Các ký pháp: Ký hiệu dùng trong các mô hình,
- Các quy tắc/luật: chỉ ra các ràng buộc được áp dụng cho các mô hình hệ thống, ví dụ: mọi thực thể trong mô hình hệ thống phải có một tên duy nhất, …;
- Các hướng dẫn áp dụng phương pháp.

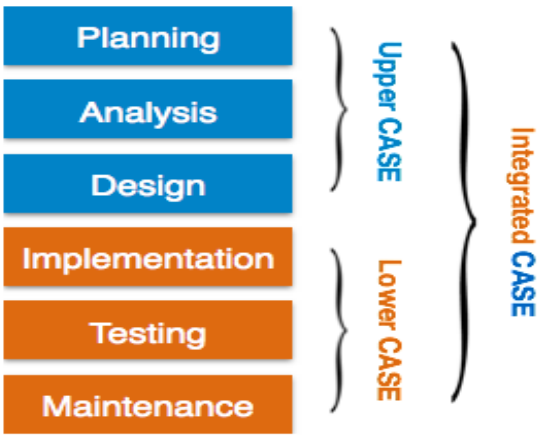
Ví dụ:

- Phương pháp phát triển phần mềm hướng chức năng
- Phương pháp phát triển phần mềm hướng đối tượng
- Phương pháp phát triển phần mềm hướng dịch vụ, …

1.2.8 CASE (Computer-Aided Software Engineering)

CASE là các sản phẩm phần mềm nhằm mục đích hỗ trợ tự động hoặc bán tự động cho các hoạt động trong quy trình phần mềm. Có ba loại CASE (xem Hình 1.12):

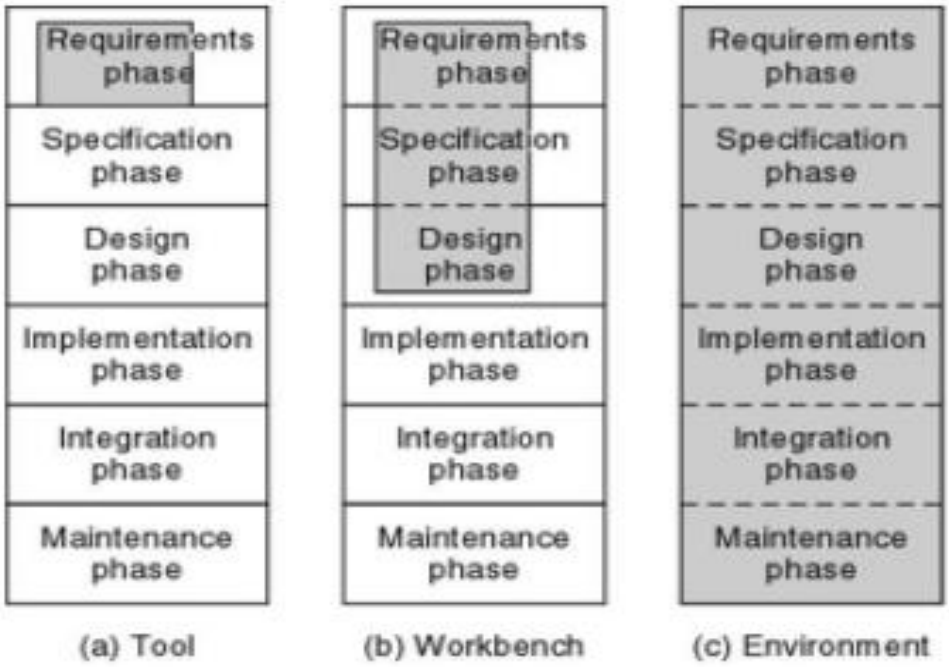
- *Upper-CASE*: Hỗ trợ các giai đoạn đầu của quy trình phần mềm như lập kế hoạch, phân tích và thiết kế.
- *Lower-CASE*: Hỗ trợ các giai đoạn tiếp theo như cài đặt, kiểm thử, bảo trì
- *Integrated CASE:* Hỗ trợ mọi giai đoạn trong vòng đời phần mềm (WHAT, HOW, BUILD, USE)



Hình 1.12: Phân loại CASE

Ngoài ra có thể phân loại CASE theo mức độ hoàn thiện (xem Hình 1.13) gồm:

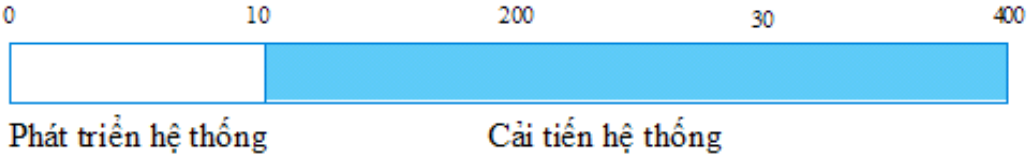
- *CASEs là Tools*: hỗ trợ các hoạt động cụ thể trong vòng đời của phần mềm.
- *CASEs là Workbenches*: phối hợp hai hoặc nhiều công cụ tập trung vào giai đoạn cụ thể của vòng đời phần mềm
- *CASEs là Enviroments*: phối hợp hai hoặc nhiều công cụ hoặc nhiều bàn làm việc và hỗ trợ vòng đời phần mềm hoàn chỉnh.



Hình 1.13: Phân loại CASE theo mức hoàn thiện

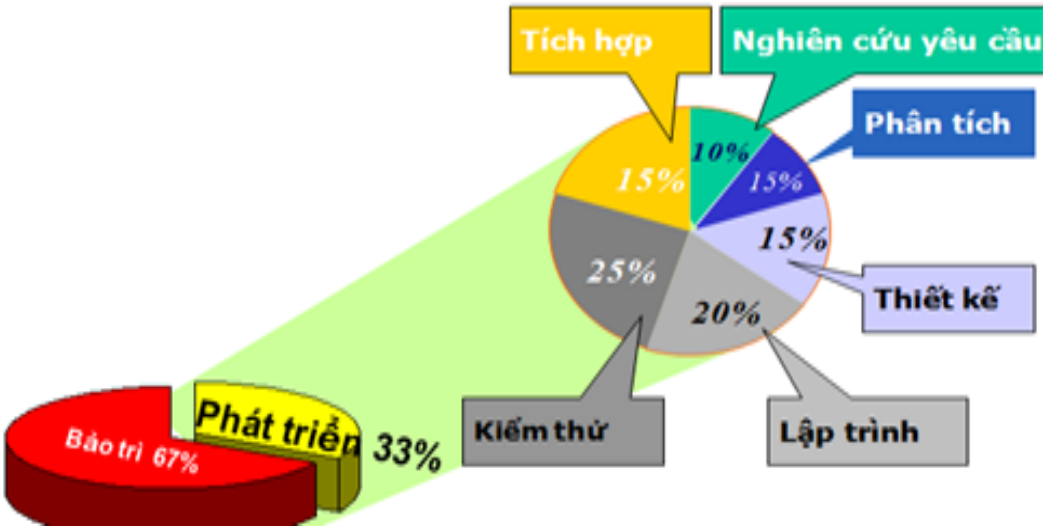
1.2.9 Chi phí của công nghệ phần mềm?

Chi phí của một hệ thống phần mềm thường gồm: (1) chi phí phần cứng; (2) chi phí phần mềm. Chi phí phần mềm thường lớn hơn chi phí phần cứng, đặc biệt với những sản phẩm lớn, phức tạp, thời gian sống lâu. Trong chi phí phần mềm, một phần lớn là dành cho bảo trì, thường gấp nhiều lần so với chi phí phát triển sản phẩm lần đầu (ví dụ, xem Hình 1.14):



Hình 1.14: Phân bố chi phí cho các hệ thống có thời gian sống dài

Ví dụ về phân bố chi phí phần mềm những năm 1990s (xem Hình 1.15)

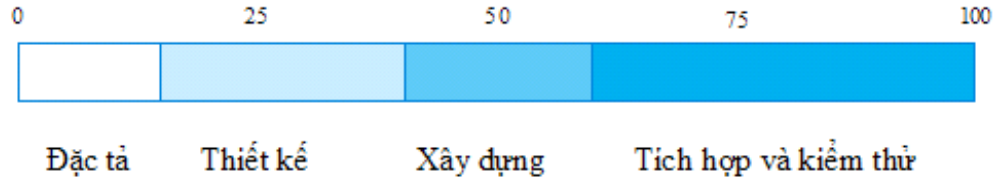


Hình 1.15: Phân bố chi phí cho các hoạt động trong vòng đời phần mềm

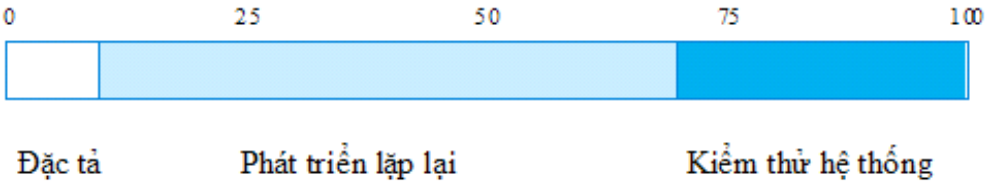
Chi phí phần mềm thường biến đổi tuỳ thuộc vào các yếu tố như:

- Loại ứng dụng được triển khai (là đơn giản hay phức tạp);
- Các yêu cầu đặt ra (nhiều, ít, cao, thấp);
- Mức độ hoàn thiện cao hay thấp;
- Năng lực của tổ chức phần mềm (nhân lực, công cụ, công nghệ, kỹ năng có được, ...);
- Quy trình phần mềm, phương pháp, công cụ sử dụng:

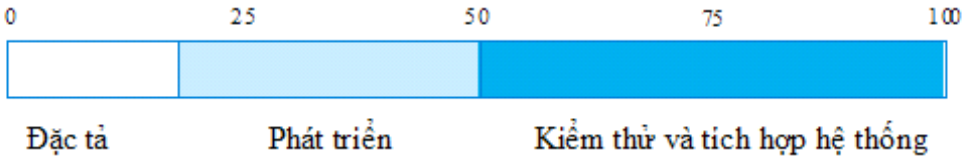
  Ví dụ: Hình 1.16; Hình 1.17; & Hình 1.18 chỉ ra % phân bố chi phí phát triển sử dụng các quy trình phần mềm thác nước, phát triển lặp và phát triển dựa trên cấu phần:



Hình 1.16: Phân bố chi phí trong mô hình thác nước



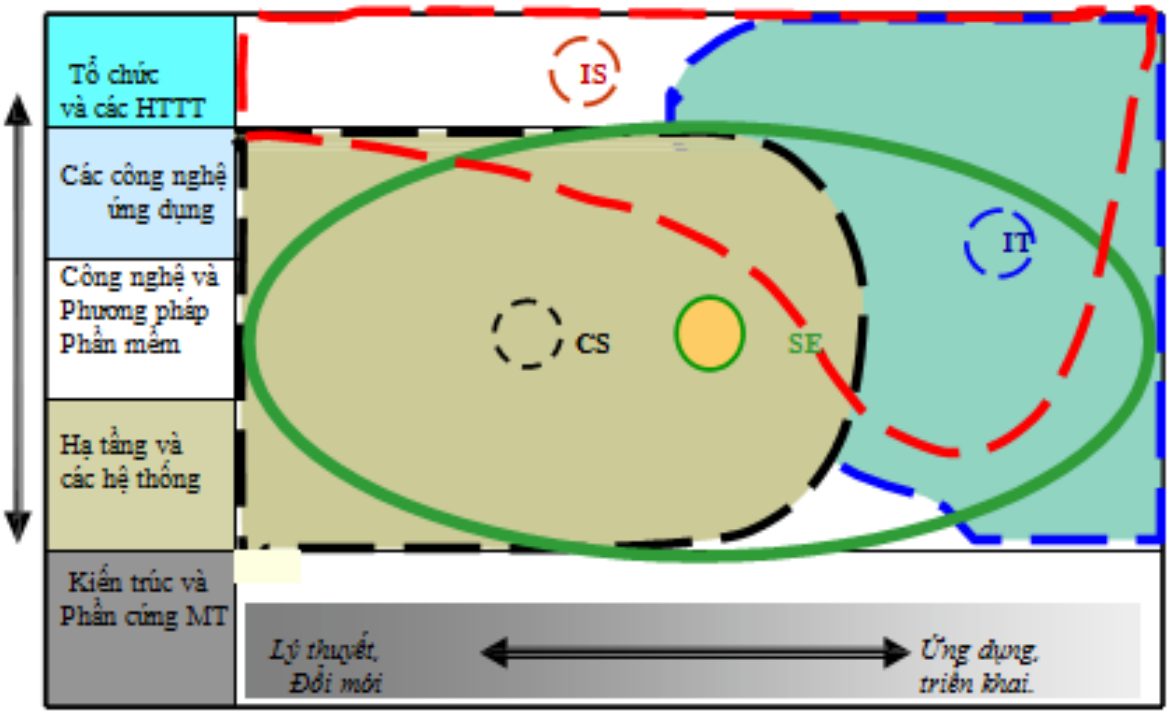
Hình 1.17: Phân bố chi phí trong mô hình phát triển lặp



Hình 1.18: Phân bố chi phí trong mô hình hướng thành phần

1.3 Các lĩnh vực nghiên cứu liên quan đến SE

Các lĩnh vực nghiên cứu liên quan đến Công nghệ phần mềm [computing curricula 11/2004 -ACM, AIS, IEEE] được biểu diễn như Hình 1.19.



Hình 1.19: SE và các lĩnh vực nghiên cứu liên quan

Trong đó: Mỗi lĩnh vực tính toán được giới hạn bằng một vùng biên có mầu sắc như sau:

- CS: Khoa học máy tính
- IS: Hệ thống thông tin
- IT: Công nghệ thông tin
- SE: Công nghệ phần mềm

Từ sơ đồ biểu diễn trên ta thấy, Công nghệ phần mềm chiếm một phần lớn ở trung tâm, đây chính là lý do tại sao SE lại là một ngành nghiên cứu phức tạp và quan trọng.

Câu hỏi:

Câu 1. Các hệ thống CASE thường được sử dụng để làm gì?

a) Quản lý dự án
b) Đánh giá độ tin cậy của phần mềm
c) Đánh giá chất lượng của phần mềm
d) Hỗ trợ các hoạt động trong quy trình xây dựng phần mềm

Câu 2. Chọn phát biểu đúng nhất trong các phát biểu sau?

a) Mục tiêu của Công nghệ Phần mềm là phát triển các sản phẩm phần mềm đáp ứng mọi yêu cầu của khách hàng
b) Mục tiêu của Công nghệ Phần mềm là phát triển các sản phẩm phần mềm dễ sử dụng, giao diện đẹp, thân thiện
c) Mục tiêu của Công nghệ Phần mềm là sản xuất phần mềm độc lập, đúng hạn, phù hợp kinh phí và đáp ứng mọi yêu cầu người sử dụng
d) Mục tiêu của Công nghệ Phần mềm là phát triển các sản phẩm phần mềm trong thời gian ngắn đáp ứng mọi yêu cầu

Câu 3. Mục đích của tiêu chuẩn ISO?

a) Làm chuẩn để phân loại các sản phẩm phần mềm
b) Làm chuẩn để đánh giá sản phẩm, hoàng hóa, dịch vụ
c) Xây dựng các tiêu chuẩn về sản xuất, thương mại và thông tin
d) Giám sát các hoạt động kinh tế - xã hội

Câu 4. Một trong các kết quả nghiên cứu nổi bật đạt được trong những năm 1970 của công nghệ phần mềm là gì ?

a) Sự ra đời của công nghệ hướng đối tượng
b) Phương pháp lập trình có cấu trúc
c) Ngôn ngữ thế hệ thứ 4 ra đời như LIPS, Prolog
d) Sự phát triển của các mô hình quản lý.

Câu 5. Khái niệm Công nghệ Phần mềm được ra đời năm nào?

a) 1960
b) 1968
c) 1970
d) 1980

Bài 2: Sự phân hoá nghề nghiệp trong Công nghệ phần mềm (Số tiết: 03 tiết)

1.4 Các trách nhiệm đạo đức và nghề nghiệp [1][2]

Để trở thành một kỹ sư phần mềm chuyên nghiệp, ngoài kiến thức về chuyên môn và kinh nghiệm, chúng ta cần tuân thủ các quy tắc về chuẩn mực đạo đức và ứng xử nghề nghiệp được thiết lập bởi cộng đồng chuyên môn. Các chuẩn mực này được sử dụng như là thước đo năng lực kỹ thuật đối với kỹ sư phần mềm và là cơ sở cho việc được công nhận, cấp giấy phép, điều kiện được ra nhập cộng đồng kỹ sư SE chuyên nghiệp.

1.4.1 Các quy định về trách nhiệm, ứng xử nghề nghiệp

- *Cẩn mật*: Tôn trọng các thông tin bí mật và sự cẩn mật của người lao động, của khách hàng bất chấp có hay không bản hợp đồng về sự sự cẩn mật được ký kết.
- *Năng lực:* Không được xuyên tạc năng lực của mình, phải biết chấp nhận những công việc nào là vượt quá khả năng của mình
- *Các quyền sở hữu trí tuệ:* Cần nắm vững các luật hiện thời về sở hữu trí tuệ như các bằng sáng chế, bản quyền. Họ cần đảm bảo rằng các quyền sở hữu trí tuệ của khách hàng và của người lao động được bảo vệ.
- *Không được lạm dụng máy tính của người khác:* Các kỹ sư phần mềm không được sử dụng các kỹ năng nghề nghiệp của mình để dùng sai máy tính của người khác. Việc lạm dụng máy tính giới hạn từ những hành động không đáng kể như chới game trên MT người dùng đến những hành động nghiêm trọng như reo rắc virus trên MT của người khác.

Kỹ sư SE vi phạm các quy định về trách nhiệm và ứng xử nghề nghiệp có thể bị phạt (theo quy định) và có thể bị tước bỏ giấy phép công nhận sự chuyên nghiệp.

1.4.2 Tập các chuẩn mực về đạo đức

Năm 1999, Hội đồng Hiệp hội máy tính & Hiệp hội kỹ sư điện và điện tử (ACM & IEEE) đã phê duyệt tập các chuẩn mực/quy tắc về đạo đức đối với các kỹ sư SE. Tập chuẩn mực này gồm 8 chuẩn mực ở dạng ngắn và dạng dài:

- Xã hội (Public): Kỹ sư phần mềm luôn làm việc vì lợi ích xã hội.
- Khách hàng và người lao động (Client and employer): Kỹ sư phần mềm có thái độ làm việc đảm bảo tối đa lợi ích của khách hàng và của người lao động, bên cạnh việc đảm bảo lợi ích xã hội.
- Sản phẩm (Product): Kỹ sư phần mềm đảm bảo phần mềm của họ cũng sửa đổi (nâng cấp) phần mềm có liên quan đạt chuẩn chuyên môn cao nhất có thể.
- Phán xét (Judgment): Kỹ sư phần mềm đảm bảo giá trị đạo đức của bản thân và tính độc lập trong các phán xét chuyên môn của mình.

- Quản lý (Management): Các nhà quản lý và lãnh đạo công nghệ phần mềm cần cam kết công khai ủng hộ và đẩy mạnh việc quản lý tuân theo các chuẩn mực đạo đức trong phát triển và quản lý chất lượng phần mềm.
- Nghề nghiệp (Profession): Kỹ sư phần mềm biết quảng bá các giá trị đạo đức và tiếng tăm của nghề nghiệp trong công nghệ phần mềm đi liền với các lợi ích xã hội.
- Đồng nghiệp (Colleagues): Kỹ sư phần mềm luôn công bằng và luôn sẵn sàng hỗ trợ đồng nghiệp của mình.
- Bản thân (Self): Kỹ sư phần mềm không ngừng học tập những kiến thức liên quan đến ngành nghề của mình và sẽ luôn quảng bá và đẩy mạnh yếu tố đạo đức trong nghề nghiệp của họ.

1.4.3 Các vấn đề về pháp lý

Các vấn đề pháp lý đáng chú ý liên quan tới kỹ nghệ phần mềm bao gồm các vấn đề liên quan đến: tiêu chuẩn; thương hiệu; bằng sáng chế; bản quyền; bí mật kinh doanh; trách nhiệm pháp lý; yêu cầu pháp lý; tuân thủ luật thương mại; và tội phạm công nghệ.

✓ **Các chuẩn**

Các tiêu chuẩn trong kỹ nghệ phần mềm cung cấp các hướng dẫn cho việc thực hiện các pha trong quy trình phát triển phần mềm. Tiêu chuẩn có chức năng định ra các yêu cầu và hỗ trợ thực hiện các yêu cầu đó khi thực hiện các hoạt động trong SE. Các tiêu chuẩn được xây dựng bằng cách liệt kê các đặc tính tối thiểu của các sản phẩm và quá trình thực hiện. Vì vậy, các doanh nghiệp, tổ chức phần mềm thường áp dụng chúng như là một phần trong chính sách tổ chức của họ. Hơn nữa, việc tuân thủ các tiêu chuẩn là một biện pháp đề phòng các rủi ro liên quan đến pháp lý và tránh khỏi những cáo buộc phi pháp.

✓ **Thương hiệu**

Một thương hiệu liên quan đến một từ, tên, biểu tượng, hoặc thiết bị được sử dụng trong các giao dịch. Thương hiệu được sử dụng "để chỉ ra nguồn gốc, xuất xứ của hàng hóa". Bảo vệ thương hiệu là bảo vệ tên, logo, hình ảnh, và bao bì của doanh nghiệp. Tổ chức Sở hữu trí tuệ thế giới (WIPO) là cơ quan đặt ra các quy tắc và quy định về thương hiệu, nhãn hiệu hàng hoá. WIPO là cơ quan của Liên Hợp Quốc dành riêng cho việc sử dụng tài sản trí tuệ như một phương pháp thúc đẩy sự đổi mới và sáng tạo.

✓ **Bằng sáng chế**

Bằng sáng chế bảo vệ các quyền của một nhà nhà sáng chế trong việc tạo ra và bán các ý tưởng của mình. Một bằng sáng chế bao gồm một tập hợp các đặc quyền được cấp bởi một chính phủ có chủ quyền cho một cá nhân/nhóm cá nhân/tổ chức trong một khoảng thời gian nhất định. Tài liệu để được cấp bằng sáng chế đòi hỏi phải được ghi chép một

cách cẩn thận trong quá trình dẫn đến các sáng chế. Các nhà sáng chế có thể thuê các đại diện sáng chế (luật sư đại diện) bảo vệ các quyền lợi hợp pháp của mình được quy định trong luật bản quyền. Lưu ý rằng, nếu sáng chế được thực hiện trong một hợp đồng SE, chủ sở hữu có thể thuộc về tổ chức phần mềm hoặc khách hàng hoặc các tổ chức phối hợp, chứ không phải thuộc về các kỹ sư phần mềm.

- ✓ **Bản quyền**

Bản quyền là thuật ngữ pháp lý mô tả quyền của người sáng tạo đối với các sản phẩm, tác phẩm văn học và nghệ thuật của họ. Bảo vệ bản quyền là tự động cho dù sản phẩm này có được đăng ký hay không. Ngay khi sản phẩm được viết ra, nó đã được bảo vệ.

- ✓ **Trách nhiệm pháp lý**

Các kỹ sư phần mềm phải quan tâm với các vấn đề về trách nhiệm trong chuyên môn. Khi cung cấp dịch vụ cho khách hàng hoặc nhà tuyển dụng cần phải tuân thủ các tiêu chuẩn và các quy định chung trong chuyên môn. Do đó, trách nhiệm pháp lý giúp người lao động & người sử dụng lao động tránh khỏi những cáo buộc hoặc thủ tục tố tụng hoặc liên quan đến sơ suất, bất cẩn, hoặc thiếu năng lực.

- ✓ **Yêu cầu pháp lý**

Kỹ sư phần mềm phải hoạt động trong phạm vi của các khuôn khổ pháp lý địa phương, quốc gia và quốc tế. Các thông tin cơ bản về khuôn khổ luật pháp quốc tế được quy định bởi tổ chức Thương mại quốc tế (WTO)

- ✓ **Tuân thủ luật thương mại**

Tất cả kỹ sư phải được nhận thức và tuân thủ những giới hạn được quy định bởi pháp luật về việc nhập, xuất, tái xuất sản phẩm dịch vụ và công nghệ; về phân loại, thu hồi giấy phép.

- ✓ **Tội phạm công nghệ**

Một số loại tôi phạm công nghệ có thể kể đến như gian lận, truy cập trái phép, spam, nội dung khiêu dâm, xúc phạm, đe dọa, quấy rối, trộm cắp dữ liệu cá nhân nhạy cảm hoặc bí mật thương mại, và sử dụng một máy tính để gây thiệt hại hay xâm nhập vào các máy tính khác trong mạng và hệ thống điều khiển tự động, … đều bị xử lý theo quy định của pháp luật hiện hành.

1.5 Nhân tố con người và sự phân hóa nghề nghiệp trong SE

1.5.1 Nhân tố con người trong ngành SE

Đối với một sản phẩn phần mềm, một người không thể hoàn thành mà là kết quả lao động của một nhóm người, gọi là nhóm phát triển phần mềm. Mỗi thành viên trong nhóm cần phải có các kiến thức cần thiết phụ thuộc vào vai trò trong nhóm; không được

vị kỷ; thành quả lao động của nhóm được xen như là thành quả chung và phải tuân thủ các quy tắc ứng xử, đạo đức nghề nghiệp.

Mặc dù đã có rất nhiều các CASE tools ra đời, nhưng tính tự động hoá chưa cao, do đó, con người vẫn là yếu tổ trung tâm, đóng vai trò quan trong trong quy trình triển khai dự án phần mềm.

1.5.2 Phân loại nghề nghiệp

Yêu cầu hiện nay của sự phát triển IT ở Việt nam đòi hỏi cần có những người lao động trong tất cả các ngành kinh tế biết sử dụng IT trong công việc của mình, và đồng thời cần có những người trực tiếp tham gia vào sản xuất, kinh doanh, vận hành quy trình sản xuất sử dụng IT; cần những lao động trực tiếp tạo ra các sản phẩm, hệ thống phần mềm, ... Do vậy, cần có những lớp người lao động sau:

- Những người biết vận dụng sáng tạo IT vào nghiệp vụ chuyên môn.
- Những người tham gia quản lí và vận hành các hệ thống IT
- Những người tham gia trực tiếp vào việc phát triển, xây dựng tạo ra các sản phẩm IT, ...

Có nhiều tiêu chí để phần loại nghề nghiệp thuộc IT như: *(a) phân loại theo mức độ kinh nghiệm; (b) phân loại theo loại hình công việc, ...*

a) Phân loại nghề nghiệp theo mức độ kinh nghiệm

- Sơ cấp: Nhân viên, cán bộ ở mức độ sơ đẳng nhất, làm việc cần có sự giám sát, hướng dẫn trực tiếp. Thời gian làm việc ở vị trí này trung bình khoảng 2 năm có thể chuyển sang mức trung cấp.
- Trung cấp: Nhân viên, cán bộ có thể làm việc độc lập theo đúng chuyên môn. Thời gian trung bình làm việc ở cấp độ này từ 2 – 5 năm thì có thể chuyển sang mức cao cấp.
- Cao cấp: Cán bộ, nhân viên có trình độ nhất định về công việc và kinh nghiệm tích luỹ, có thể đào tạo, huấn luyện người khác. Những cán bộ có từ 5 – 7 năm kinh nghiệm và có ít nhất là 3 năm để học các kỹ năng chuyên sâu có thể chuyển sang mức lãnh đạo.
- Lãnh đạo: Làm việc độc lập, kiêm các nhiệm vụ giám sát, còn được gọi là chuyên gia phụ trách dự án.
- Chuyên gia kỹ thuật: Chuyên gia kỹ thuật là người có kinh nghiệm rộng rãi trong nhiều lĩnh vực. Họ có thể làm việc ở các vị trí khác nhau trong quy trình dự án.
- Nhà quản lý: Có kinh nghiệm chuyên môn nghiệp vụ, có nhiều kiến thức, kinh nghiệm về quản lý dự án.

b) Phân loại nghề nghiệp theo loai hình công việc

Phân loại nghề nghiệp theo loại hình công việc được chia thành bốn nhóm ngành: *(1) phát triển ứng dụng; (2) hỗ trợ ứng dụng; (3) chuyên ngành kỹ thuật; (4) Nhân viên; và (5) Nhóm nghề khác.*

- Phát triển ứng dụng
  - Lập trình viên;
  - Kỹ sư phần mềm (phân tích viên, thiết kế viên, người lãnh đạo/quản lý dự án);
  - Kỹ sư tri thức (KE): kỹ sư tri thức tương tự như các kỹ sư phần mềm nhưng được chuyên môn hoá các kỹ năng để xây dựng các hệ thống trí tuệ nhân tạo, hệ tri thức, ...
- Hỗ trợ ứng dụng
  - Chuyên gia ứng dụng: Tư vấn cho đội dự án về các loại ứng dụng phần mềm cần sử dụng.
  - Quản trị dữ liệu: Quản trị cơ sở dữ liệu (DBA), phân tích, thiết kế, xây dựng, bảo lưu, quản lý quyền truy cập vào cơ sở dữ liệu của hệ thống.
  - Kỹ sư trí tuệ nhân tạo: Cố vấn cho đội dự án xác định, thiết kế và cài đặt trí tuệ vào các ứng dụng. Kỹ sư trí tuệ nhân tạo thường có trình độ chuyên môn về AI cao hơn các kỹ sư tri thức.
  - Nhà tư vấn: Có sự hiểu biết rộng về mọi vấn đề, có kiến thức và kinh nghiệm thực hành, thực tế về dự án phần mềm. Số năm kinh nghiệm càng cao thì kiến thức tích luỹ càng nhiều.
- Chuyên ngành kỹ thuật
  - Nhà phân tích và kỹ sư truyền thông: Phân tích, thiết kế, đàm phán và/ hoặc lắp đặt các thiết bị và phần mềm truyền thông; có thể làm việc trên mainframe hoặc các mạng truyền thông, kiến thức nền tảng phải có gồm: điện tử, kỹ thuật, các ứng dụng truyền thông, khoa học máy tính và các công nghệ truyền thông.
  - Chuyên gia về mạng cục bộ: Thiết kế, đi dậy, lắp đặt mạng cục bộ; quản lý, duy trì khả năng hoạt động của mạng cục bộ; giám sát tài nguyên cung cấp qua mạng cục bộ, quản lý cấu hình và khắc phục các vấn đề về mạng cục bộ.
  - Lập trình viên hệ thống: Cài đặt, bảo dưỡng các hệ điều hành, các ứng dụng hỗ trợ phần mềm; có thể giám sát hàng trăm ứng dụng vận hành trên nền tảng, theo dõi, khắc phục những sự cố liên quan đến phần mềm hệ thống – Đây là một nhiệm vụ quan trọng.
  - Chuyên gia hỗ trợ phần mềm: Cài đặt, bảo dưỡng các gói phần mềm được sử dụng đội dự án và bởi khách hàng.
- Nhân viên

- Chuyên gia bảo mật: Chịu trách nhiệm về các vấn đề liên quan đến bảo mật, an toàn, an ninh của hệ thống, khắc phục, sử lý sự cố về bảo mật.
- Kiểm soát viên: Kiểm tra, giám sát quá trình triển khai dự án so với lịch biểu, lập báo cáo trình lãnh đạo nếu phát hiện vấn đề, rủi ro phát sinh.
- Đào tạo viên: Nắm bắt các công nghệ, công cụ, kiến thức mới và training đội dự án cũng như training khách hàng sử dụng sản phẩm.
- Người viết các chuẩn và tài liệu kỹ thuật: Xác định các công việc cần chuẩn hoá, xây dựng các chính sách, thủ tục chuẩn hoá cho tổ chức phần mềm phù hợp với các quy định chung về nghề nghiệp. Khác với người viết các tài liệu kỹ thuật ở chỗ, người viết tài liệu kỹ thuật thường lấy thông tin về sản phẩm phần mềm/ứng dụng/công nghệ và biên soạn thành tài liệu mô tả về sản phẩm/ứng dụng/công nghệ, về các đặc điểm, chức năng, công dụng của chúng.
- Đảm bảo chất lượng (QA): Lập kế hoạch đảm bảo chất lượng dự án, chịu trách nhiệm giám sát, đánh giá và quản lý các vấn đề về chất lượng liên quan đến dự án phần mềm.
- Lập kế hoạch công nghệ: Nắm bắt các xu hướng phát triển công nghệ, tư vấn lựa chọn và áp dụng các công nghệ thích hợp cho tổ dự án.

❖ Nhóm nghề khác

- Hỗ trợ sản phẩm: Nhân viên việc với nhóm người dùng cuối (cài đặt, thiết lập cấu hình, …); bán hàng; trực đường dây nóng hỗ trợ khách hàng sử dụng sản phẩm; cần có khả năng giao tiếp tốt.
- Tiếp thị sản phẩm: Tiếp thị sản phẩm, quảng bá sản phẩm; cần có kỹ năng giao tiếp tốt và các kiến thức về tiếp thị, chăm sóc khách hàng.

Lưu ý: SV có thể tham khảo Roadmap về các kỹ năng cần thiết đối với kỹ sư SE tại [11] và một số website tuyển dụng tại [12] và một số dự án nguồn mở cho người mới bắt đầu tại [13]; dự án tictictoe tại [3].

Câu hỏi:

Câu 1. Chọn phát biểu đúng nhất về sản phẩm phần mềm trong các phát biểu sau?

a) Phần mềm gồm ba phần chính: chương trình máy tính, cấu trúc dữ liệu (ngoài và trong) và tài liệu.
b) Phần mềm là tên gọi khác của chương trình máy tính
c) Phần mềm gồm chương trình máy tính và phần cứng đi kèm
d) Phần mềm là các ứng dụng được cài đặt trên máy tính

Câu 2. MS Word thuộc loại phần mềm nào?

b) Phần mềm hệ thống
c) Phần mềm tiện ích
d) Phần mềm ứng dụng
e) Phần mềm khoa học kỹ thuật

Câu 3. Trong phát triển phần mềm, yếu tố nào quan trọng nhất

a) Con người
b) Quy trình
c) Sản phẩm
d) Thời gian

Câu 4. Phần mềm dự báo thời tiết thu thập các số liệu về nhiệt độ, độ ẩm, … xử lý tính toán để cho ra các dự báo thời tiết là 1 ví dụ của loại phần mềm nào?

a) Phần mềm hệ thống (System Software)
b) Phần mềm trí tuệ nhân tạo (Artificial Intelligence Software)
c) Phần mềm thời gian thực (Real time Software)
d) Phần mềm nghiệp vụ (Business Software)

Câu 5. Theo thống kê từ những thách thức đối với công nghệ phần mềm thì lỗi nhiều nhất là do

a) Kiểm thử và bảo trì
b) Phân tích yêu cầu
c) Thiết kế
d) Lập trình

Bài tập cuối chương

Bài 1.1.

a) Nêu khái niệm về phần mềm. Lấy ví dụ và mô tả về một phần mềm mà bạn sử dụng thường xuyên.
b) Liệt kê 5 thuộc tính chất lượng cho một phần mềm tốt. Hãy đánh giá phần mềm mà bạn đã lựa chọn ở ý trên với các thuộc tính chất lượng này.

Bài 1.2. Hãy lấy 5 ví dụ phần mềm mà bạn biết theo các loại phần mềm sau đây:

a) Phần mềm hệ thống (System Software)
a) Phần mềm trí tuệ nhân tạo (Artificial Intelligence Software)
b) Phần mềm thời gian thực (Real time Software)
c) Phần mềm nghiệp vụ (Business Software)

Bài 1.3. Hãy cho biết các nhiệm vụ cơ bản mà tất cả các dự án kỹ nghệ phần mềm phải triển khai là gì?

Bài 1.4. Trình bày ngắn gọn các nhiệm vụ đã liệt kê ở bài tập 1.3

Bài 1.5. Hãy liệt kê ít nhất 5 thứ cần chuẩn bị cho giai đoạn phát hành/deployment phần mềm.

Bài 1.6. Trước khi bắt đầu triển khai dự án SE, bạn cần chuẩn bị sẵn các công cụ/công nghệ nào để sử dụng xuyên suốt qua mọi giai đoạn tiến triển của dự án (beginning to end - từ giai đoạn thu thập yêu cầu của khách hàng → kết thúc dự án chuyển giao sản phẩm đến khách hàng)?

Bài 1.7. Liệt kê 7 tính năng mà một hệ thống quản lý tài liệu dự án cần có

Bài 1.8. Microsoft Word cung cấp một công cụ đơn giản để theo dõi thay đổi. Nó không phải là hệ thống quản lý tài liệu với đầy đủ các tính năng, nhưng nó đủ tốt cho các dự án nhỏ. Bạn hãy thực hành các bước sau để theo dõi thay đổi dự án sử dụng Microsoft Word:

a) Tạo một tài liệu ngắn trong Word và ghi lại
b) Theo dõi thay đổi trong tài liệu (với các phiên bản Word gần đây bạn làm như sau: go to the Review tab's Tracking group and click Track Changes.)
c) Sửa đổi tài liệu và ghi lại với một tên mới (bạn nên xem các thay đổi đã được cắm cờ trong tài liệu bằng cách go to the Review tab's Tracing group and use the drop-down to select Final: Show Markup)
d) Trên Review tab's Tracking group, click và nút Reviewing Pane để hiện thị trang reviewing. Ban nên xem các thay đổi ở đó
e) Trong Review tab's Tracking group, mở the Track Changes drop-down và lựa chọn Change User Name. Thay đổi tên user của bạn và các thiết lập ban đầu.
f) Tạo một thay đổi khác trên tài liệu, ghi lại file và xem cách thức Word chỉ ra các thay đổi.

Bài 1.9. Microsoft Word cũng cung cấp một công cụ so sánh tài liệu. Nếu bạn thực hiện theo đúng các hướng dẫn ở bài tập trước bạn sẽ có 2 phiên bản của cùng một tài liệu. Trong the Review tab's Compare group, mở the Compare drop-down và lựa chọn Compare. Lựa chọn 2 phiên bản của file và so sánh chúng. Hãy cho biết những điểm tương tự và điểm khác biệt/thay đổi giữa 2 phiên bản được thể hiện như thế nào? Tại sao bạn sử dụng công cụ này thay vì sử dụng một công cụ theo dõi thay đổi khác?

Bài 1.10. Giống như Microsoft Word, Google Docs cũng là các công cụ theo dõi thay đổi đơn giản. Đi đến địa chỉ: *https://www.google.com/docs/about/* để tìm hiểu thêm về công cụ này và đăng ký sử dụng. Sau đó tạo một tài liêu, ghi lại, đóng lại, mở lại tài liệu, tạo các thay đổi trong nội dung của tài liệu và làm các bước tương tự như bài tập 1.8. Để hiển thị các thay đổi, mở menu File và chọn See revision history. Click vào nút See more detailed revisions để xem các thay đổi của bạn.

Bài 1.11. Liệt kê các tài liệu cần xây dựng tại giai đoạn tiền dự án

Chương II: QUY TRÌNH PHẦN MỀM

Nội dung chính của chương

2.1 Quy trình phần mềm

2.2 Mô hình quy trình phần mềm

2.3 Lập kế hoạch quản lý dự án phần mềm

Mục tiêu cần đạt được của chương

- Hiểu rõ các khái niệm về quy trình phần mềm, xác định chi tiết những công việc phải làm trong quy trình phần mềm và cách thực hiện chúng.
- Nắm được một số mô hình phát triển phần mềm và khi nào chúng có thể được sử dụng
- Có thể ứng dụng những mô hình phát triển phần mềm đã nghiên cứu trên những hệ thống phần mềm cụ thể.

Bài 3: Quy trình phần mềm (Số tiết: 03 tiết)

2.1 Quy trình phần mềm

2.1.1 Khái niệm

Quy trình phần mềm là một tập hợp các hành động mà mục đích của nó là xây dựng và phát triển phần mềm. Những hành động thường được thực hiện trong các quy trình phần mềm bao gồm:

- Xác định yêu cầu: Đặc tả những gì hệ thống phải làm và các ràng buộc trong quá trình xây dựng hệ thống.

- Phát triển: Thiết kế và cài đặt phần mềm.

- Đánh giá phần mềm: phần mềm phải được đánh giá và thẩm định để đảm bảo rằng nó đã thoả mãn tất cả các yêu cầu.

- Cải tiến phần mềm: Điều chỉnh và thay đổi phần mềm tương ứng với sự thay đổi yêu cầu.

Những loại hệ thống khác nhau sẽ cần những quy trình phát triển khác nhau. Vì vậy, nếu ta không sử dụng một quy trình phát triển hệ thống thích hợp thì có thể làm giảm chất lượng của hệ thống và tăng chi phí xây dựng [1] [4].

2.1.2 Các giai đoạn trong quy trình phần mềm

2.1.2.1 Đặc tả phần mềm

Đặc tả phần mềm (hay còn gọi là kỹ thuật xác định yêu cầu) là quy trình tìm hiểu và định nghĩa những dịch vụ khách hàng yêu cầu và các ràng buộc trong quá trình vận hành và xây dựng hệ thống [2].

Quy trình xác định yêu cầu bao gồm bốn pha chính:

- Nghiên cứu tính khả thi: Nghiên cứu tính khả thi giúp xác định những yêu cầu của người sử dụng có thoả mãn những công nghệ hiện tại hay không. Về góc độ kinh doanh, nghiên cứu khả thi nhằm xác định hệ thống đưa ra có mang lại lợi nhuận không. Việc nghiên cứu khả thi nên được thực hiện một cách nhanh chóng và không quá tốn kém. Kết quả của việc nghiên cứu khả thi sẽ xác định có nên tiếp tục xây dựng hệ thống nữa hay không.

- Phân tích và rút ra các yêu cầu: đây là quy trình đưa ra các yêu cầu hệ thống thông qua một số phương pháp như: quan sát hệ thống hiện tại, phỏng vấn và thảo luận với người sử dụng, phân tích nhiệm vụ, phân tích tài liệu hoặc hệ thống cũ … Trong pha này, chúng ta có thể phải xây dựng một hoặc nhiều mô hình hệ thống và các mẫu thử.

- Đặc tả yêu cầu: Pha này sẽ tư liệu hoá những thông tin thu thập được. Có hai loại yêu cầu cần được xác định:

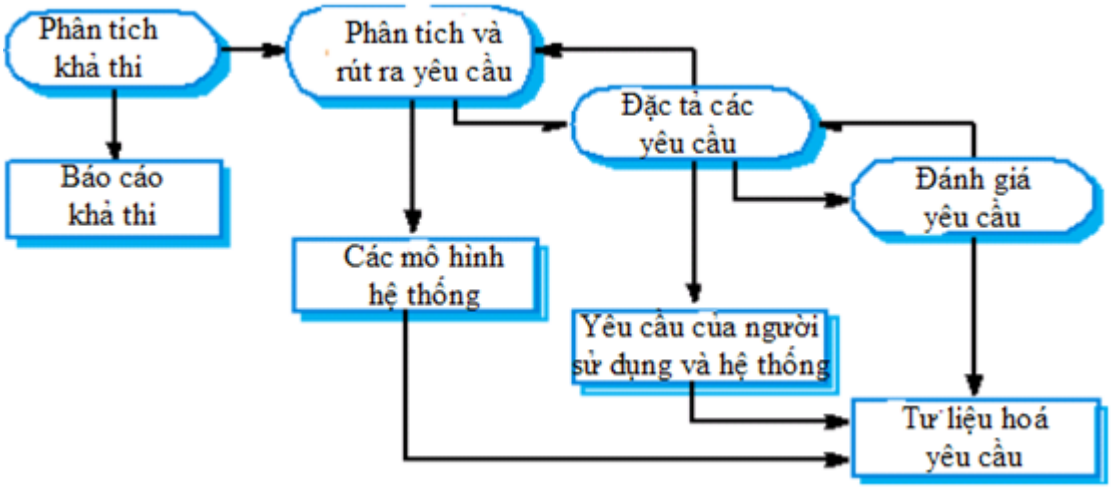
Yêu cầu của người sử dụng: là những yêu cầu bằng ngôn ngữ tự nhiên. Kiểu yêu cầu này được viết bởi người sử dụng.

Yêu cầu hệ thống: là những tài liệu có cấu trúc, đươc mô hình hoá, mô tả chi tiết về các chức năng, dịch vụ và các ràng buộc vận hành của hệ thống. Yêu cầu hệ thống sẽ định nghĩa những gì cần phải xây dựng, cho nên nó có thể trở thành bản hợp đồng giữa khách hàng và nhà thầu. Các yêu cầu hệ thống được chia làm 2 loại:

Các yêu cầu hệ thống chức năng: Là các dịch vụ mà hệ thống phải cung cấp

Các yêu cầu phi chức năng: Là các ràng buộc mà hệ thống phải tuân theo

- Đánh giá yêu cầu: pha này sẽ kiểm tra lại các yêu cầu xem chúng có đúng thực tế hay không, có thống nhất không, có đầy đủ không. Nếu phát hiện ra lỗi thì ta phải chỉnh sửa các lỗi này.



Hình 2.1: Quy trình xác định yêu cầu

2.1.2.2 Thiết kế và cài đặt phần mềm

Thiết kế phần mềm:

Là quá trình thiết kế cấu trúc phần mềm dựa trên những tài liệu đặc tả. Hoạt động thiết kế bao gồm những công việc chính sau:

- Thiết kế kiến trúc: Thiết kế các hệ thống con cấu thành lên hệ thống cần xây dựng và mối quan hệ giữa chúng được xác định và tư liệu hoá.

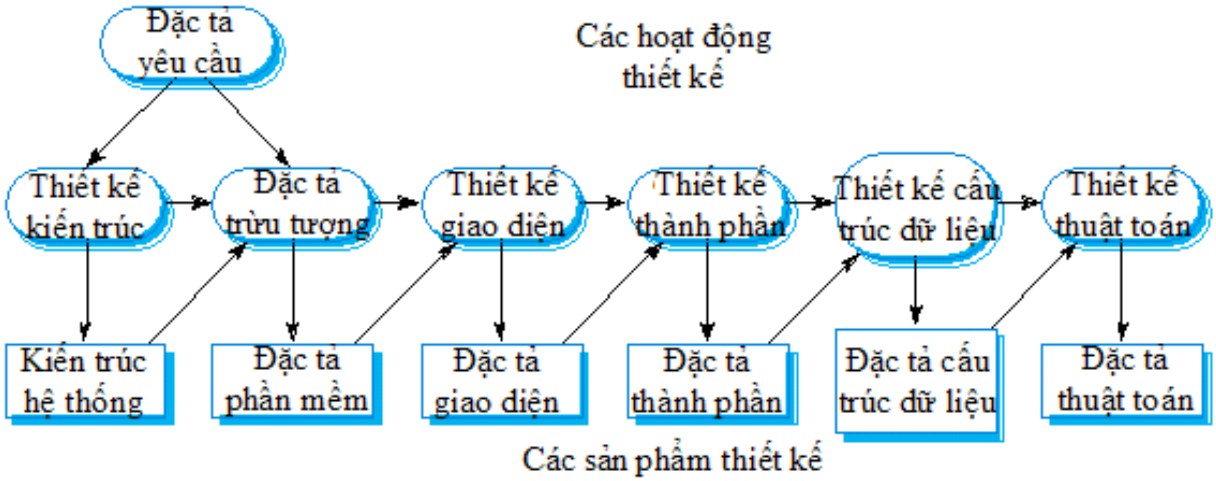
- Đặc tả trừu tượng: Với mỗi hệ thống con, phải có một bản đặc tả về các dịch vụ của nó và những ràng buộc khi nó vận hành.

- Thiết kế giao diện: Với mỗi hệ thống con, các giao diện của nó với những hệ thống con khác phải được thiết kế và tư liệu hoá.

- Thiết kế thành phần: Các dịch vụ cung cấp cho các thành phần khác và các giao diện tương tác với chúng phải được thiết kế.

- Thiết kế cấu trúc dữ liệu (thiết kế dữ liệu): Cấu trúc dữ liệu được sử dụng để cài đặt hệ thống phải được thiết kế một cách chi tiết và cụ thể.

- Thiết kế thuật toán: Các thuật toán được sử dụng để cung cấp các dịch vụ phải được thiết kế chi tiết và chính xác.



Hình 2.2: Mô hình chung của quá trình thiết kế

Cài đặt phần mềm:

Là quy trình chuyển đổi từ tài liệu đặc tả hệ thống thành một hệ thống thực, có thể vận hành được và phải loại bỏ các lỗi của chương trình.

Lập trình là một hành động cá nhân, không có quy trình lập trình chung. Người lập trình phải thực hiện một số kiểm thử để phát hiện ra lỗi trong chương trình và loại bỏ nó trong quy trình gỡ lỗi.

Nhận xét: *Trong ba giai đoạn: thiết kế, cài đặt và bảo trì thì thiết kế là giai đoạn quan trọng nhất, chịu trách nhiệm đến 80% đối với sự thành công của một sản phẩm. Cài đặt là việc thực thi những gì đã thiết kế. Nếu trong quá trình cài đặt có xuất hiện vấn đề thì phải quay lại sửa bản thiết kế. Quá trình thiết kế tốt là cơ sở để quản lý và giảm chi phí cho công việc bảo trì phần mềm sau này.*

2.1.2.3 Đánh giá phần mềm

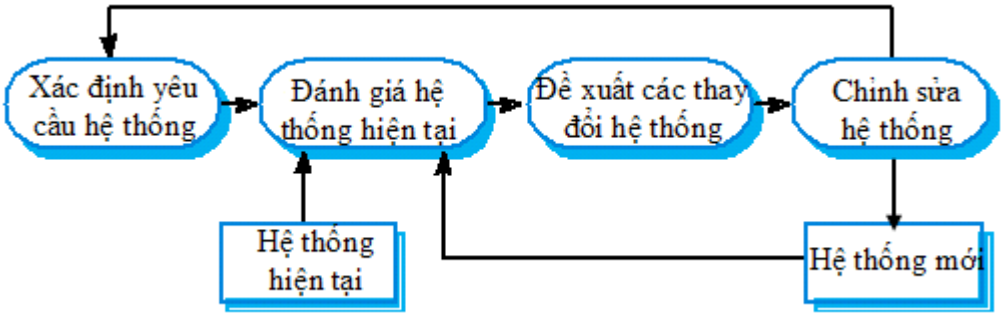
Đánh giá phần mềm hay còn gọi là xác minh và thẩm định (V&V - Verification and Validation) được sử dụng để chỉ ra rằng hệ thống đã thực hiện theo đúng các đặc tả và thoả mãn mọi yêu cầu của khách hàng.

Đánh giá phần mềm bao gồm các công đoạn: Kiểm tra, xem xét lại, và kiểm thử hệ thống. Kiểm thử hệ thống tức là cho hệ thống thực hiện trên những trường hợp có dữ liệu thật được lấy từ tài liệu đặc tả hệ thống. Các mức kiểm thử bao gồm:

- Kiểm thử đơn vị
- Kiểm thử tích hợp
- Kiểm thử hệ thống
- Kiểm thử chấp nhận

2.1.2.4 Cải tiến phần mềm

Khi các yêu cầu hệ thống thay đổi theo sự thay đổi của các yêu cầu nghiệp vụ thì phần mềm phải cải tiến và thay đổi để hỗ trợ khách hàng. Thông thường chi phí để bảo trì và cải tiến thường đắt hơn nhiều so với chi phí xây dựng phần mềm.



Hình 2.3: Cải tiến hệ thống

2.2 Mô hình quy trình phần mềm

Mô hình quy trình phần mềm là một thể hiện đơn giản của một quy trình phần mềm, và nó được biểu diễn từ một góc độ cụ thể. Sau đây là một số mô hình quy trình phổ biến [5]:

- Mô hình thác nước

- Mô hình chữ V
- Mô hình mẫu thử
- Mô hình xoắn ốc
- và một số mô hình khác

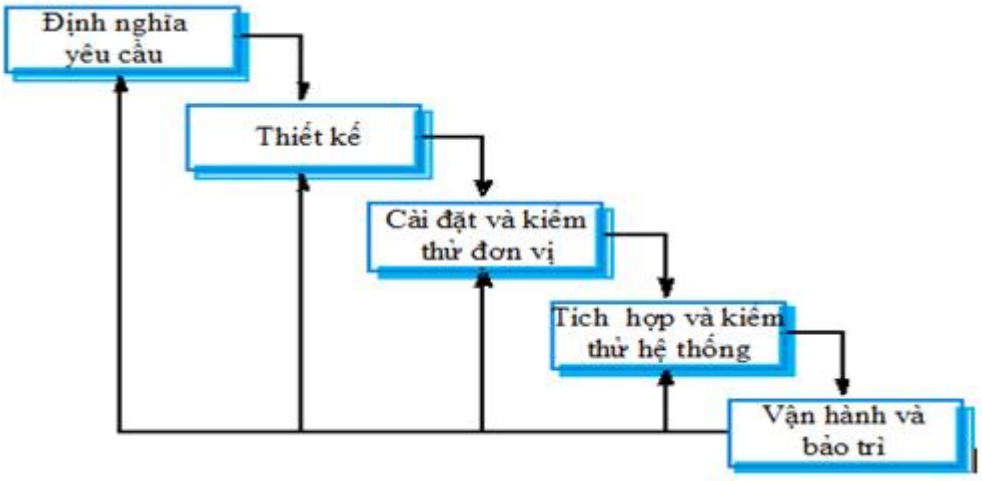
2.2.1 Mô hình thác nước (Waterfall model)

Mô hình thác nước được giới thiệu trong một bài báo của Winston Royce vào năm 1970. Đây là mô hình được công bố đầu tiên xuất phát từ các quy trình công nghệ trước đó. quy trình phát triển trông giống như một dòng chảy, với các pha được thực hiện theo trật tự nghiêm ngặt và không có sự quay lui hay nhảy vượt pha nên mô hình này được gọi là mô hình thác nước hay vòng đời phần mềm. Mô hình thác nước là một ví dụ về quy trình hướng kế hoạch, chúng ta cần phải lập kế hoạch và xếp lịch cho tất các hoạt động của quy trình trước khi bắt đầu thực hiện chúng [1].

Mô hình thác nước là một mô hình có tính trình tự trong phát triển phần mềm. Chú trọng vào sự phát triển logic và lần lượt trong suốt vòng đời phát triển phần mềm (software development life cycle-SDLC), trong đó những pha (phase) ở phía trước phải được hoàn thành trước khi những pha tiếp theo có thể bắt đầu. Mỗi pha trong mô hình thác nước có vai trò khác nhau trong SDLC.

Mặc dù sự phổ biến của mô hình này đã giảm nhiều trong vài năm trở lại đây khi các phương pháp linh hoạt (Agile) xuất hiện, tuy nhiên mô hình thác nước vẫn đang được ưa chuộng bởi nhiều kỹ sư CNTT vì tính logic và tuần tự của mô hình này.

Các pha của mô hình thác nước bao gồm:



Hình 2.4: Mô hình thác nước

Mô hình thác nước xem quá trình xây dựng một sản phẩm phần mềm bao gồm năm pha tách biệt, năm pha được thực hiện một cách tuần tự, kết thúc pha trước, rồi mới

được thực hiện pha tiếp theo. Sản phẩm đầu ra của pha trước trở thành đầu vào của pha sau.

Có hai hoạt động phổ biến được thực hiện trong mỗi pha là:

- Kiểm tra – phê chuẩn
- Quản lý cấu hình.

Tổng kết mỗi pha là sự kiểm tra, có phê chuẩn và quản lý cấu hình, đây chính là mục tiêu của sản phẩm. Việc kiểm tra đưa ra khuôn mẫu đúng đắn tương ứng giữa sản phẩm phần mềm và các đặc tính của nó. Sự phê chuẩn đưa ra chuẩn mực về sự phù hợp hay chất lượng của sản phẩm phần mềm đối với mục đích của quá trình hoạt động.

Nhận xét:

*** Ưu điểm:**

- Dễ quản lý. Đây chính là mô hình ưa thích của các nhà quản lý dự án. Thời gian hoàn thành dự án thường được dự báo với độ chính xác hơn so với các mô hình khác. Các tài liệu đầu ra của từng giai đoạn cũng được xây dựng đầy đủ và hệ thống hơn.
- Dễ dàng trong thẩm tra đánh giá.
- Mang tính tự nhiên khi sử dụng.
- Dễ sử dụng, dễ tiếp cận các pha và hoạt động được xác định rõ ràng
- Xác nhận ở từng pha đảm bảo phát hiện sớm các lỗi
- Dễ dàng triển khai và quy trình dễ hiểu. Yêu cầu đầu vào và đầu ra có sự rõ ràng, nên tiến trình làm việc rất dễ dàng và chất lượng.
- Với những dự án nhỏ, mô hình thác nước hoạt động hiệu quả và cho kết quả rất tốt.
- Vì quá trình phát triển rất chặt chẽ, do đó chất lượng mỗi phần và cả dự án sẽ rất chặt chẽ.
- Kết quả được ghi chép dễ dàng

*** Nhược điểm:**

- Mối quan hệ giữa các pha không được thể hiện

- Hệ thống phải được kết thúc ở từng giai đoạn do vậy rất khó thực hiện được đầy đủ những yêu cầu của khách hàng... vì trong mô hình này rất khó khăn trong việc thay đổi các pha đã được thực hiện. Giả sử, pha phân tích và xác định yêu cầu đã hoàn tất và chuyển sang pha kế tiếp, nhưng lúc này lại có sự thay đổi yêu cầu của người sử dụng; thì chỉ còn cách là phải thực hiện lại từ đầu.

- Ít tính linh hoạt và phạm vi điều chỉnh của mô hình khá là khó khăn, tốn kém.

- Khách hàng phải kiên nhẫn. Họ chỉ được tham gia vào dự án ở giai đoạn phân tích yêu cầu và test mà thôi. Ngoài ra sản phẩm sẽ chỉ được bàn giao khi tất cả các công việc liên quan đã được hoàn thành.

- Mô hình thác nước không phù hợp để triển khai với những dự án dài và phức tạp, có nhiều sự thay đổi trong yêu cầu

- Rủi ro cao hơn. Do quá trình kiểm thử trong mô hình thác nước xuất hiện gần cuối, các chiến thuật để giảm thiểu rủi ro không được triển khai từ sớm. Trong khi ở những mô hình linh hoạt, quá trình kiểm thử diễn ra song song với quá trình phát triển do đó việc giảm thiểu rủi ro hiệu quả hơn rất nhiều

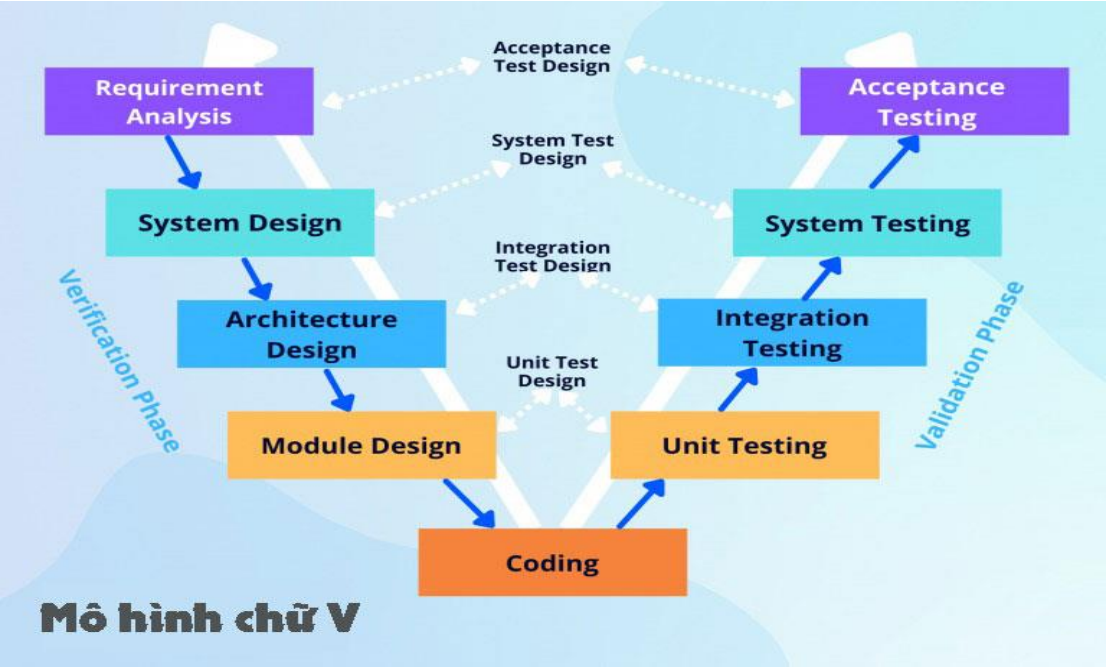
Mô hình thác nước chỉ nên được sử dụng khi đội dự án đã có kinh nghiệm, yêu cầu từ khách hàng được xác định rõ ngay từ đầu và ít có khả năng thay đổi, dự án ngắn và đơn giản.

2.2.2 Mô hình chữ V

Một trong những hạn chế lớn nhất của mô hình phát triển phần mềm thác nước là: Các khiếm khuyết được tìm thấy rất chậm trong quá trình phát triển vì kiểm thử được thực hiện vào cuối chu kỳ phát triển. Fix bug càng chậm thì càng khó khăn và tốn kém. Để khắc phục vấn đề này, một mô hình phát triển mới được giới thiệu gọi là "Mô hình chữ V".

Mô hình phát triển phần mềm V là một trong những quy trình phát triển phần mềm được sử dụng rộng rãi. Trong mô hình V việc thực hiện kiểm tra được diễn ra ngay từ giai đoạn xác định yêu cầu. V mô hình cũng được gọi là mô hình xác minh (verification) và mô hình thẩm định (validation).

Trong mô hình V, các hoạt động phát triển và đảm bảo chất lượng được thực hiện đồng thời. Không có pha rời rạc được gọi là kiểm thử, thay vào đó kiểm thử được bắt đầu ngay từ giai đoạn lấy yêu cầu. Các hoạt động xác minh và thẩm định đi liền với nhau [19].



Hình 2.5: Mô hình chữ V

Trong một quá trình phát triển theo mô hình V điển hình, bên trái là các hoạt động phát triển và bên tay phải là các hoạt động kiểm thử. Trong giai đoạn phát triển cả việc xác minh và thẩm định được thực hiện cùng với các hoạt động phát triển thực tế.

Giai đoạn xác minh (Phía bên trái):

Các hoạt động phía bên trái là hoạt động phát triển. Thông thường, chúng ta khó có thể thực hiện thử nghiệm trong giai đoạn phát triển, nhưng đây là ưu điểm của mô hình này, nó chứng tỏ rằng thử nghiệm cũng có thể được thực hiện ở tất cả các giai đoạn của các hoạt động phát triển.

1. Phân tích yêu cầu - Requirement Analysis

Trong giai đoạn này các yêu cầu được thu thập, phân tích và nghiên cứu. Ở giai đoạn này việc hệ thống chạy như thế nào không quan trọng, quan trọng là hệ thống có những chức năng gì. Brain storming/walkthrough, interviews cần được thực hiện để có mục tiêu rõ ràng.

- Hoạt động xác minh:Đánh giá yêu cầu (Requirements review).
- Hoạt động xác nhận: Tạo test case UAT (User acceptance test- kiểm thử chấp nhận) = Đầu ra cần có: Tài liệu hiểu về yêu cầu, UAT test case.

2. Yêu cầu hệ thống - High level design

Trong giai đoạn này, high level design của phần mềm được xây dựng. Nhóm sẽ nghiên cứu và điều tra về các yêu cầu có thể được thực hiện như thế nào. Tính khả thi về mặt kỹ

thuật của yêu cầu cũng được tìm hiểu. Nhóm cũng tìm hiểu về các mô-đun sẽ được tạo/ phụ thuộc, nhu cầu phần cứng/ phần mềm

- Hoạt động xác minh: Đánh giá thiết kế (Design reviews)
- Hoạt động xác nhận: Tạo test plan và test case, tạo ma trân truy vết (traceability metrics)
- Đầu ra cần có: System test cases, Feasibility reports, System test plan, tài liệu về yêu cầu phần cứng và các mô đun, ...

3. Thiết kế kiến trúc

Trong giai đoạn này, dựa trên thiết kế mức cao, kiến trúc phần mềm được tạo ra. Các mô-đun, mối quan hệ và sự phụ thuộc của họ, sơ đồ kiến trúc, bảng cơ sở dữ liệu, chi tiết về công nghệ đều được hoàn tất trong giai đoạn này.

- Hoạt động xác minh: Đánh giá thiết kế
- Hoạt động xác nhận: Kế hoạch thử nghiệm tích hợp và các trường hợp thử nghiệm.
- Đầu ra cần có: Tài liệu thiết kế, Kế hoạch kiểm thử tích hợp và các trường hợp thử nghiệm, Thiết kế bảng cơ sở dữ liệu,...

4. Thiết kế mô đun - Thiết kế cấp thấp

Trong giai đoạn này mỗi mô-đun hoặc các thành phần phần mềm đều được thiết kế riêng. Các method, class, giao diện, các kiểu dữ liệu vv đều được hoàn tất trong giai đoạn này.

- Hoạt động xác minh: Đánh giá thiết kế
- Hoạt động xác nhận: Tạo và xem xét các trường hợp kiểm tra đơn vị.
- Đầu ra cần có: Các đơn vị kiểm tra đơn vị

5. Code

Trong giai đoạn này, code được thực hiện.

- Hoạt động xác minh: Xem xét mã, kiểm tra các trường hợp kiểm tra
- Hoạt động xác nhận: Tạo các trường hợp kiểm tra chức năng.
- Đầu ra cần có: các trường hợp thử nghiệm, danh sách kiểm tra xem lại.

Giai đoạn thẩm định (Phía bên phải):

Phía bên phải thể hiện các hoạt động kiểm thử.

1. Unit Testing – Kiểm thử đơn vị: Loại bỏ lỗi ở cấp code hoặc đơn vị

2. Integration Testing – Kiểm thử tích hợp: Xác nhận thông tin nội bộ giữa các mô-đun trong hệ thống

3. System Testing – Kiểm thử hệ thống: Kiểm tra các yêu cầu chức năng và phi chức năng của ứng dụng đã phát triển.

4. User Acceptance Testing (UAT) – Kiểm thử chấp nhận người dùng: Xác nhận khả năng sử dụng của hệ thống đã phát triển

Nhận xét:

*** Ưu điểm:**

- Quá trình phát triển và quy trình quản lý có tính tổ chức và hệ thống
- Hoạt động tốt cho các dự án có quy mô vừa và nhỏ.
- Kiểm tra bắt đầu từ khi bắt đầu phát triển vì vậy sự mơ hồ được xác định ngay từ đầu.
- Dễ dàng quản lý vì mỗi giai đoạn có các mục tiêu và mục tiêu được xác định rõ ràng.

*** Nhược điểm:**

- Không thích hợp cho các dự án lớn và phức tạp
- Không phù hợp nếu các yêu cầu thường xuyên thay đổi.
- Không có phần mềm làm việc được sản xuất ở giai đoạn trung gian.
- Không có điều khoản cho việc phân tích rủi ro nên có sự không chắc chắn và có tính rủi ro.

Khi nào sử dụng mô hình chữ V?

- Khi các yêu cầu và mục tiêu phải rõ ràng
- Có sẵn các điều kiện kỹ thuật như nguồn lực và chuyên gia
- Các lỗi hệ thống được phát hiện có thể chấp nhận được
- Dự án có quy mô vừa và nhỏ.
- Công nghệ và công cụ được sử dụng không thường xuyên thay đổi.

2.2.3 Mô hình mẫu thử (Prototyping model)

Thông thường, khách hàng sẽ đưa ra mục tiêu của họ một cách chung chung mà họ không biết hoặc không đưa ra một cách cụ thể những cái vào, cái ra và các tiến trình xử lý chúng. Thêm vào đó, chúng ta cũng không thể không quan tâm đến thuật toán sử dụng, tính tương thích của sản phẩm phần mềm với môi trường của nó như: phần cứng,

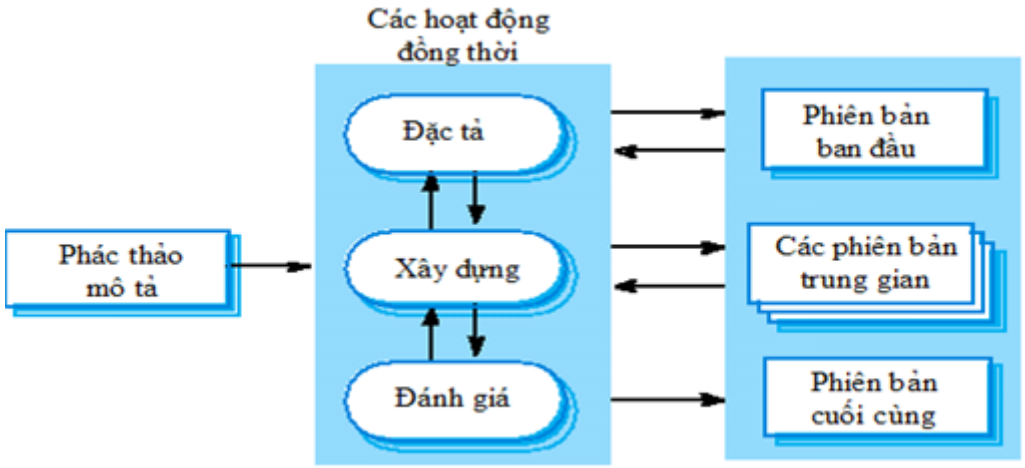
hệ điều hành...Trong trường hợp này, mô hình mẫu có thể là sự lựa chọn tốt hơn cho người lập trình.

Mô hình xây dựng tiến triển dựa trên ý tưởng xây dựng một mẫu thử ban đầu và đưa cho người sử dụng xem xét; sau đó, tinh chỉnh mẫu thử qua nhiều phiên bản cho đến khi thoả mãn yêu cầu của người sử dụng thì dừng lại [1].

- Có hai phương pháp để thực hiện mô hình này:

+ Phát triển thăm dò: mục đích của nó là để làm việc với khách hàng và để đưa ra hệ thống cuối cùng từ những đặc tả sơ bộ ban đầu. Phương pháp này thường bắt đầu thực hiện với những yêu cầu được tìm hiểu rõ ràng và sau đó, bổ sung những đặc điểm mới được đề xuất bởi khách hàng. Cuối cùng, khi các yêu cầu của người sử dụng được thoả mãn thì cũng là lúc chúng ta đã xây dựng xong hệ thống.

+ Loại bỏ mẫu thử: mục đích là để tìm hiểu các yêu cầu của hệ thống. Phương pháp này thường bắt đầu với những yêu cầu không rõ ràng và ít thông tin. Các mẫu thử sẽ được xây dựng và chuyển giao tới cho người sử dụng. Từ đó, ta có thể phân loại những yêu cầu nào là thực sự cần thiết và lúc này mẫu thử không còn cần thiết nữa. Như vậy, mẫu thử chỉ có tác dụng để làm sáng tỏ yêu cầu của người sử dụng.



Hình 2.6: Prototyping model

Nhận xét:

*** Ưu điểm**:

- Chú trọng tái sử dụng mẫu. Một phần của hệ thống có thể được phát triển ngay trong giai đoạn phân tích phát triển yêu cầu và thiết kế.
- Hoạt động nào cũng có sự tham gia của người dùng, nên sản phẩm cuối cùng mang tính khách quan, thỏa mãn yêu cầu của người sử dụng.
- Cho phép thay đổi yêu cầu và khuyến khích người sử dụng tham gia trong suốt chu kỳ dự án.

- Nhanh chóng xác định được các yêu cầu
- Tạo cơ sở ký kết hợp đồng
- Giúp đào tạo huấn luyện người sử dụng.
- Các hoạt đông xảy ra đồng thời, hoạt động này bổ sung cho hoạt động kia.

* **Nhược điểm** của mô hình xây dựng tiến triển là: thiếu tầm nhìn của cả quy trình; các hệ thống thường hướng cấu trúc nghèo nàn; yêu cầu các kỹ năng đặc biệt (Ví dụ: các ngôn ngữ để tạo ra mẫu thử nhanh chóng). Input/output chưa rõ ràng, Khó đánh giá tính hiệu quả của thuật toán.Khó khăn trong thiết kế: việc phát triển qua nhiều giai đoạn có thể làm phá vỡ kiến trúc tổng quan của PM. Khó khăn trong quản lý do các pha hoạt động đồng thời, mô phỏng nhanh nên tài liệu mập mờ không rõ ràng. Yêu cầu đội phát triển dự án phải có năng lực. Tốn chi phí do mỗi lần sửa đổi, đưa ra bản mẫu thử.

Mô hình xây dựng tiến triển chỉ nên áp dụng với những hệ thống có tương tác ở mức độ nhỏ hoặc vừa; trên một phần của những hệ thống lớn (ví dụ như giao diện người sử dụng); hoặc những hệ thống có thời gian chu kỳ tồn tại ngắn.

2.2.4 Mô hình xoắn ốc (Spiral Model)

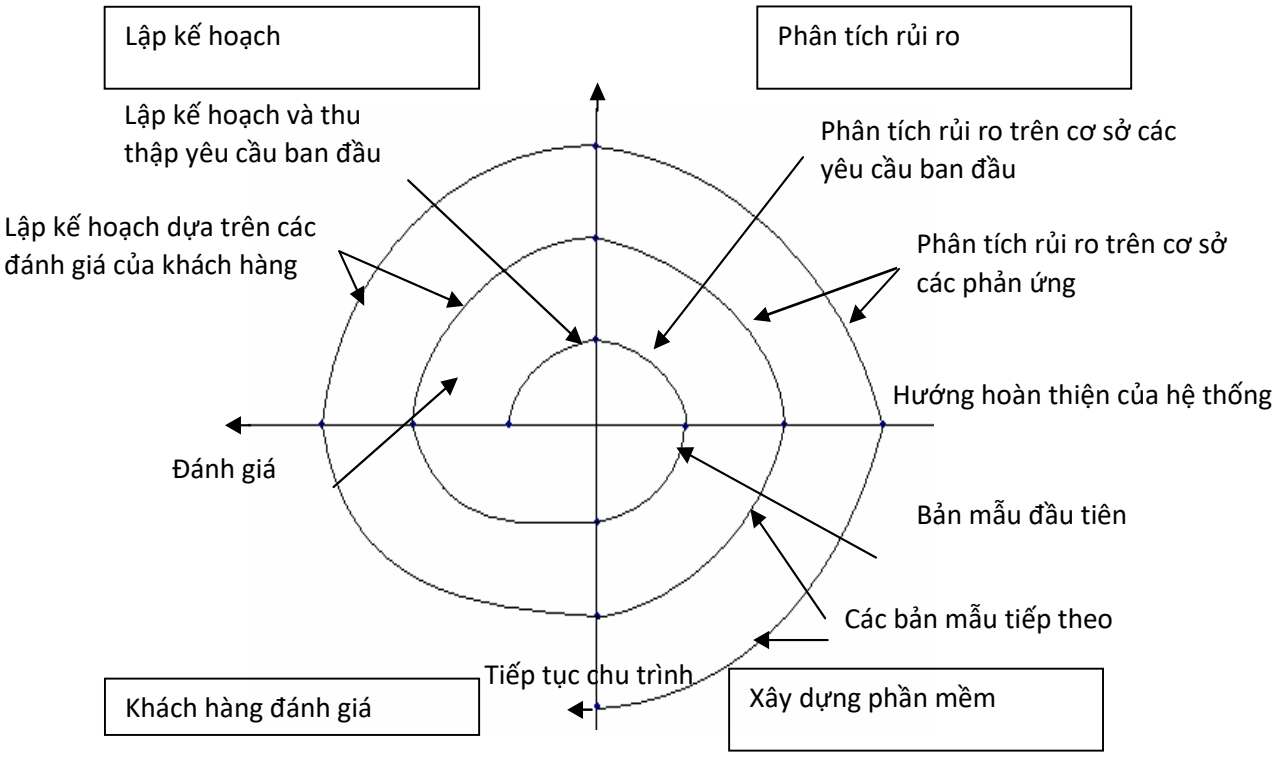
Mô hình này đượcBoehm đưa ra nên đôi lúc còn được gọi là mô hình Boehm's (The Boehm's spiral model - 1988). Nó có thể xem là sự kết hợp giữa mô hình thác nước và mô hình mẫu và đồng thời thêm một thành phần mới - phân tích rủi ro. Trong mô hình xoắn ốc, quy trình phát triển phần mềm được biểu diễn như một vòng xoắn ốc.

Mỗi vòng lặp trong mô hình biểu diễn một pha trong qui trình. Vòng lặp trong nhất là tính khả thi của hệ thống, vòng lặp tiếp theo là xác định các yêu cầu, vòng lặp tiếp nữa là thiết kế hệ thống…. Các rủi ro có thể được nhận rõ và được giải quyết trong suốt qui trình.

Các pha trong quy trình phát triển xoắn ốc bao gồm:

- Planning: Xác định mục tiêu, tương tác và ràng buộc.
- Risk analysis: Phân tích các lựa chọn và các chỉ định/giải quyết rủi ro.
- Engineering: Phát triển sản phẩm
- Customer evaluation: Đánh giá kết quả xây dựng.

Mô hình được tóm tắt như sau:



Hình 2.7: Mô hình xoắn ốc

Các giai đoạn của mô hình xoắn ốc:

Lập kế hoạch – Planning phase: Thu thập, phân tích yêu cầu từ của dự án từ phía khách hàng. Bao gồm các công việc: Ước lượng chi phí (estimating cost), lên lịch trình thực hiện dự án (shedule-master), xác định số lượng nhân lực, môi trường làm việc (identifying necessary resources and work environment), tìm hiểu yêu cầu hệ thống (requirements) từ đó đưa ra các tài liệu đặc tả (Bussiness Requirement Specifications và System Requirement specifications) để phục vụ cho việc trao đổi giữa khách hàng và phân tích hệ thống sau này.

Phân tích rủi ro – Risk analysis phase: Một quá trình phân tích sẽ được thực hiện để xác định rủi ro và đưa ra các giải pháp thay thế. Một prototype sẽ được tạo ra ở cuối giai đoạn phân tích rủi ro. Nếu có bất kỳ rủi ro nào được tìm thấy trong quá trình này thì các giải pháp thay thế sẽ được đề xuất và thực hiện.

Thực thi kỹ thuật – Engineering phase: Đây là giai đoạn mà dự án được các dev tiến hành code, các tester tiến hành test và deploying software trên trang web của khách hàng.

Đánh giá – Evaluation phase: Khách hàng sẽ tham gia vào giai đoạn này để đánh giá công việc, sản phẩm và đảm bảo rằng sản phẩm đáp ứng tất cả các yêu cầu đã đặt ra trước đó. Nếu có bất kỳ yêu cầu thay đổi nào từ khách hàng, các giai đoạn sẽ được lặp lại. Đây là giai đoạn quan trọng vì cần có sự phản hồi của khách hàng về sản phẩm trước khi sản phẩm được release.

Nhận xét:

**** Ưu điểm:***

- Sau mỗi lần tăng vòng thì có thể chuyển giao kết quả thực hiện được cho khách hành nên các chức năng của hệ thống có thể nhìn thấy sớm hơn.

- Các vòng trước đóng vai trò là mẫu thử để giúp tìm hiểu thêm các yêu cầu ở những vòng tiếp theo.

- Những chức năng của hệ thống có thứ tự ưu tiên càng cao thì sẽ được kiểm thử càng kỹ.

- Mô hình xoắn ốc là một cách tiếp cận hiện thực để phát triển các phần mềm và các hệ thống lớn. Vì PM được phát triển theo sự tiến hóa, khách hàng và nhà phát triển hiểu nhau tốt hơn và rủi ro sẽ được giảm thiểu sau mỗi lần tiến hóa.

- Các ước lượng (budget, schedule, etc.) trở nên thực tế hơn như là một quy trình làm việc, bởi vì những vấn đề quan trọng đã được phát hiện sớm hơn. Có sự tham gia sớm của deverlopers Quản lý rủi ro và phát triển hệ thống theo phase.

**** Nhược điểm:***

- Chi phí cao và thời gian dài để có sản phẩm cuối cùng.

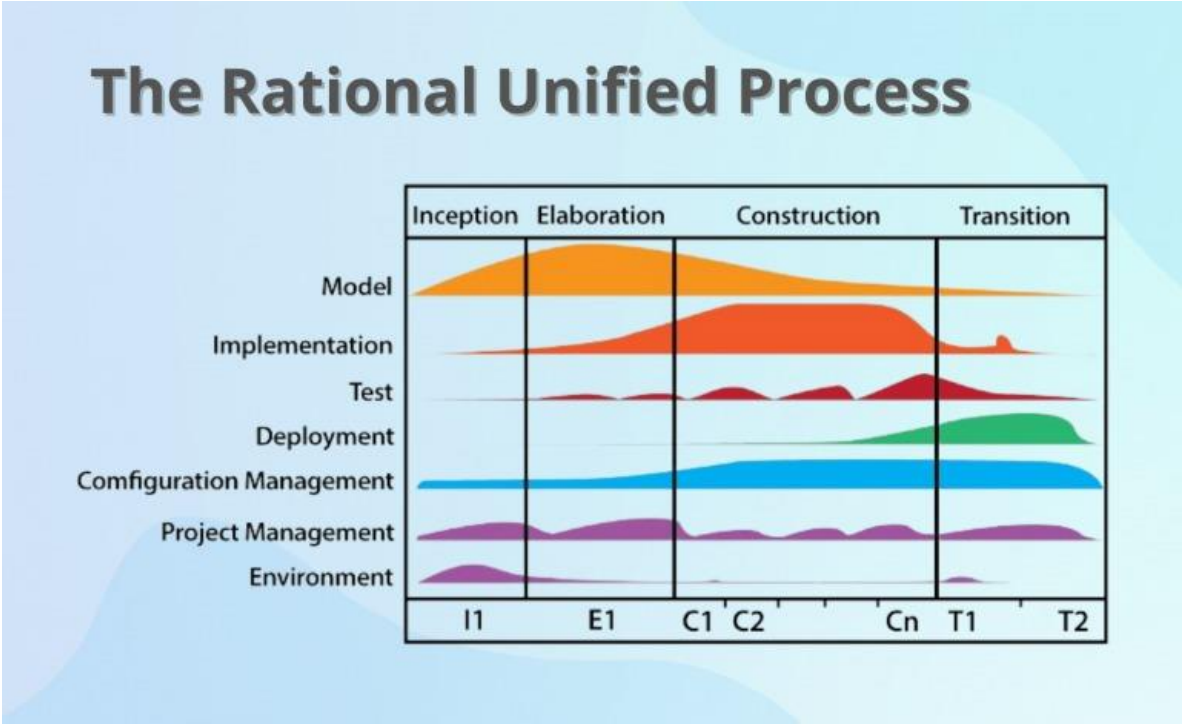
- Phải có kỹ năng tốt để đánh giá rủi ro và giả định.

- Mô hình này chưa thực sự quen thuộc với các nhà phát triển như trong mô hình tuyến tính.

- Hạn chế lớn nhất của mô hình này là khả năng thuyết phục khách hàng về giảm thiểu tính rủi ro trong quá trình phát triển.

Mô hình xoắn ốc thường được sử dụng cho các ứng dụng lớn và các hệ thống được xây dựng theo các giai đoạn nhỏ hoặc theo các phân đoạn

2.2.5 Tiến trình hợp nhất – RUP



Hình 2.8: Tiến trình hợp nhất – RUP

Rational Unified Process (RUP) - Tiến trình hợp nhất được phát triển bởi hãng IBM là một phương pháp phát triển ứng dụng phần mềm bao gồm một số công cụ hỗ trợ mã hóa sản phẩm cuối cùng và các hoạt động đi kèm với nó. Tiến trình này yêu cầu việc phát triển ứng dụng một cách chặt chẽ và nghiêm ngặt với việc đưa ra các mẫu được thực hiện nhanh chóng qua các cuộc làm việc vớI khách hàng và nhóm dự án, việc lập kế hoạch và đưa ra các chức năng hệ thống một cách tích cực. Kết quả sẽ đưa ra một ứng dụng đáp ứng các yêu cầu của người sử dụng và giúp cho quá trình lên kế hoạch và thực thi nhanh chóng. RUP là một phương pháp hướng tới đối tượng để quản lý dự án và phát triển phần mềm chất lượng cao.

RUP là một tập hợp các phương pháp có thể điều chỉnh theo môi trường và nhu cầu của từng công ty, chứ không phải là một hệ thống với các quy trình cứng nhắc.

Các giai đoạn của RUP

- Khởi tạo**:** Ý tưởng được hình thành
- Thiết kế**:** Các trường hợp sử dụng và kiến trúc được thiết kế
- Xây dựng**:** Các hoạt động từ thiết kế đến thành phẩm
- Chuyển đổi**:** Các hoạt động tiếp theo để đảm bảo sự hài lòng của khách hàng

Khi nào thì sử dụng mô hình RUP

✓ Có sự thay đổi liên tục trong các yêu cầu.

- ✓ Khi bạn có thông tin và dữ liệu chính xác.
- ✓ Khi cần tích hợp nhất định trong suốt quá trình phát triển.

2.2.6 Mô hình phát triển ứng dụng nhanh (RAD – Rapid Application Development)

Mô hình RAD được biểu diễn như hình dưới:

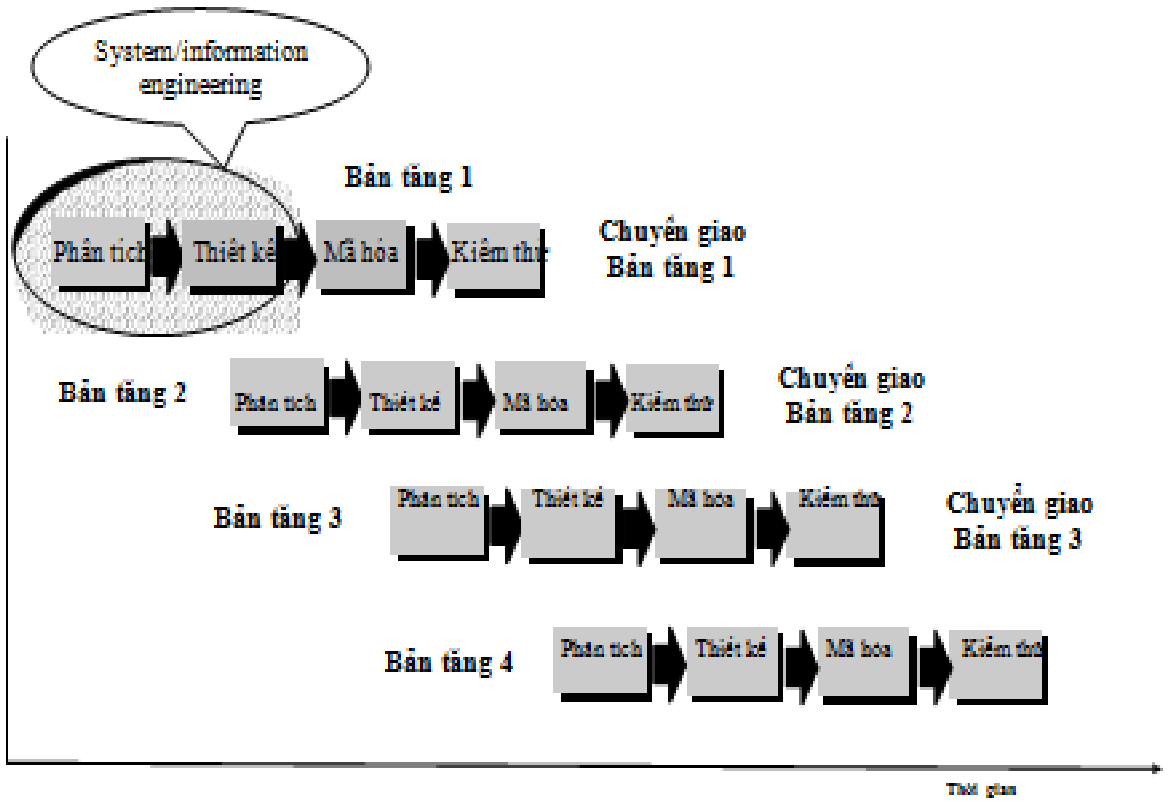
Hình 2.9: Mô hình RAD

Đặc điểm của mô hình RAD:

- ✓ Hợp với những hệ thống có khả năng mô đun hóa cao: Hướng thành phần, tái sử dụng, sử dụng công cụ tự động.
- ✓ Thời gian phát triển sản phẩm nhanh: 60 – 90 ngày
- ✓ Không phù hợp với các sản phẩm: khó phân chia thành các thành phần, đòi hỏi hiệu năng cao.

2.2.7 Mô hình tăng trưởng (incremental model)

Mô hình tăng trưởng được biểu diễn như hình dưới:



Hình 2.10: Mô hình tăng trưởng

Đặc điểm của mô hình tăng trưởng

- ✓ Chuyển giao dần từng thành phần của hệ thống
  - Sản phẩm chia thành từng lần tăng theo yêu cầu chức năng
  - Yêu cầu người dùng ưu tiên theo thứ tự lần tăng.
- ✓ Có thể áp dụng cho các sản phẩm dùng trong thời gian ngắn
  - Đáp ứng nhanh yêu cầu của khách hàng
  - Chiếm lĩnh thị trường
  - Khác với bản mẫu.
- ✓ Công ty phát triển phải có tiềm lực cao như Công nghệ, tài sản phần mềm

2.2.8 Mô hình Agile – Agile Model

Agile là một tập hợp các nguyên lý dành cho phát triển phần mềm, trong đó khuyến khích việc lập kế hoạch thích ứng, phát triển tăng dần, chuyển giao sớm, và cải tiến liên tục. Agile cũng chủ trương thích ứng nhanh chóng với các thay đổi. Agile không định nghĩa một phương pháp cụ thể để đạt được những điều này, nhưng lại có nhiều phương pháp phát triển phần mềm khác nhau thỏa mãn và hướng theo các tiêu chí đó.

Mục đích của các phương pháp Agile là giúp các doanh nghiệp đạt được sự linh hoạt (Agility), từ đó nâng cao sức cạnh tranh và phát triển bền vững. Các phương pháp Agile đã thay đổi diện mạo thế giới không chỉ trong Phát triển phần mềm mà còn đang thể hiện giá trị trong các lĩnh vực khác như Marketting (Agile Marketting), giáo dục (EduScrum, Lean Edu, v.v.), thiết kế (Lean UX, Design Thinking), khởi nghiệp (Lean Startup) và Phần cứng.

Nhận xét:

*** Ưu điểm:** Giảm thời gian cần thiết để tận dụng một số tính năng của hệ thống, kết quả cuối cùng là phần mềm chất lượng cao trong thời gian ít nhất có thể và sự hài lòng của khách hàng.

* **Nhược điểm:** Phụ thuộc vào kỹ năng của người phát triển phần mềm.

Mô hình Agile có thể được sử dụng với bất kỳ loại hình dự án nào, nhưng nó cần sự tham gia và tính tương tác của khách hàng. Ngoài ra, nó có thể được sử dụng khi khách hàng yêu cầu chức năng sẵn sàng trong khoảng thời gian ngắn (3 tuần).

2.2.9 Mô hình Scrum

Scrum là một quy trình phát triển phần mềm thuộc họ agile. Là một quy trình phát triển phần mềm theo mô hình linh hoạt (agile). Với nguyên tắc chủ đạo là chia nhỏ phần mềm cần sản xuất ra thành các phần nhỏ để phát triển (các phần nhỏ này phải đọc lập và Release được), lấy ý kiến khách hàng và thay đổi cho phù hợp ngay trong quá trình phát triển để đảm bảo sản phẩm release đáp ứng những gì khách hàng mong muốn. Scrum chia dự án thành các vòng lặp phát triển gọi là các sprint. Mỗi sprint thường mất 2- 4 tuần (30 ngày) để hoàn thành. Nó rất phù hợp cho những dự án có nhiều sự thay đổi và yêu cầu tốc độ cao.

Nhận xét:

*** Ưu điểm:** Một người có thể làm nhiều việc ví dụ như dev có thể test Phát hiện lỗi sớm hơn rất nhiều so với các phương pháp truyền thống Khách hàng nhanh chóng thấy được sản phẩm qua đó đưa ra phản hồi sớm. Có khả năng áp dụng được cho những dự án mà yêu cầu khách hàng không rõ ràng ngay từ đầu.

*** Nhược điểm:** Trình độ của nhóm là có một kỹ năng nhất định Phải có sự hiểu biết về mô hình aglie . Khó khăn trong việc xác định ngân sách và thời gian. Luôn nghe ý kiến phản hồi từ khách hàng và thay đổi theo nên thời gian sẽ kéo dài khi có quá nhiều yêu cầu thay đổi từ khách hàng. Vai trò của PO (Product Owner) rất quan trọng, PO là người định hướng sản phẩm; Nếu PO làm không tốt sẽ ảnh hưởng đến kết quả chung.

Câu hỏi:

Câu 1. Áp dụng mô hình nào là thích hợp khi các yêu cầu đã được tìm hiểu rõ ràng và ít thay đổi

a) Mô hình làm bản mẫu
b) Mô hình thác nước
c) Mô hình xoắn ốc
d) Mô hình phát triển dựa trên thành phần

Câu 2. Giai đoạn nào sau đây không nằm trong tiến trình RUP ?

a) Khởi đầu
b) Chi tiết
c) Xây dựng
d) Thẩm định

Câu 3. Phần mềm xây dựng theo mô hình mẫu thử có những đặc điểm gì?

a) Giao diện ít thân thiện
b) Mang tính khách quan, đáp ứng yêu cầu người dùng và thường được người dùng chấp nhận
c) Phức tạp, khó dùng
d) Dễ dàng đánh giá tính hiệu quả của thuật toán

Câu 4. Mô hình phát triển ứng dụng nhanh (RAD) là…

a) Mô hình đặt trọng tâm vào phân tích rủi ro
b) Mô hình tăng dần với chu kỳ phát triển ngắn
c) Một mô hình dựa trên nguyên mẫu.
d) Cả a, b, c

Câu 5. Quy trình phát triển phần mềm bao gồm những hoạt động cơ bản nào sau đây?

a) Phân tích thành phần, điều chỉnh yêu cầu, thiết kế hệ thống tái sử dụng, phát triển tích hợp
b) Đặc tả yêu cầu, xây dựng hệ thống, kiểm thử, cải tiến.
c) Đặc tả yêu cầu, phần tích thành phần, thiết kế hệ thống tái sử dụng, cài đặt
d) Tạo bản mẫu, lấy ý kiến, cải tiến

Bài 4: Lập kế hoạch quản lý dự án phần mềm (Số tiết: 03 tiết)

2.3 Lập kế hoạch quản lý dự án phần mềm

Một thực tế là những dự án CNTT có quy mô lớn thường vượt qua ngân sách trung bình được đề ra ban đầu. Có một số nhân tố ảnh hưởng đến thực tế này. Trước đây chúng ta đã xác định được là do các yêu cầu về chức năng, yêu cầu kỹ thuật được xây dựng không đầy đủ.

Nếu như quá trình lập kế hoạch không được thực hiện đúng cách thì ảnh hưởng của nó có thể sẽ là một thảm họa. Ngược lại, nếu quá trình lập kế hoạch dự án được thực hiện tốt, thì những lỗi trong yêu cầu kĩ thuật có thế được xác định sớm và dự án sẽ có được một nền tảng vững chắc. Trong bài học này chúng ta sẽ học cách đặt nền tảng cho một dự án.

Bản chất động và sự phức tạp của các dự án công nghệ thông tin đòi hỏi một người quản lý dự án xuất sắc phải thiết lập được một nền tảng vững chắc. Thiết lập được nền tảng như vậy đòi hỏi phải có sự hiểu biết vững chắc về phân tích cấu trúc chi tiết công việc (Work Breakdown Structure - WBS) và cách sử dụng các sơ đồ mạng (Network Diagram), phương pháp đường tới hạn (Critical Path Method), lịch trình dự án (Project Schedules) và ngân sách dự án. Không nắm vững được những vấn đề này thường dẫn tới việc dự án hoạt động với hiệu quả không ổn định thậm chí có thể thất bại. Ngoài ra, nếu bạn nắm vững những công cụ này, bạn sẽ dần thấy mình có một cơ sở vững chắc bất kể bạn đang làm cho dự án CNTT nào.

Kỳ vọng cho việc lập kế hoạch dự án

Lập kế hoạch là một kỹ năng mà nhiều người nghĩ rằng mình đã có kinh nghiệm, tuy nhiên rất ít người thực sự có được kỹ năng này. Xây dựng một bản kế hoạch dự án toàn diện là một trong những công việc khó khăn nhất trong suốt vòng đời của dự án.

Đặt những kỳ vọng có tính thực tế cho quy trình lập kế hoạch và thảo luận về các phương pháp truyền tải những kỳ vọng này trước khi bắt đầu công việc lập kế hoạch dự án.

Là một người quản lý dự án CNTT, bạn được đánh giá chủ yếu trên cơ sở thành công hay thất bại của các dự án mà bạn đang tham gia. Và bạn được giao thời gian hoặc nguồn lực để lập kế hoạch. Do các dự án CNTT rất phức tạp và luôn biến động nên đòi hỏi quá trình lập kế hoạch tổng thể phải thành công.

Nếu muốn dự án hoàn thành đúng thời hạn, đúng ngân sách và đúng yêu cầu kỹ thuật thì phải đảm bảo ban quản lý dự án có những kỳ vọng hợp lý về thời gian và tài nguyên để hoàn thành công việc.

2.3.1 Định nghĩa lập kế hoạch

Lập kế hoạch là một phương thức tiếp cận có hệ thống, cách nhìn chỉnh thể, toàn diện dự án nhằm xác định các phương pháp, tài nguyên và các công việc cần thiết để đạt được mục tiêu.

Mục tiêu của các Dự án CNTT là hoàn thành đúng thời hạn, đúng ngân sách, cung cấp hiệu quả hoạt động mong muốn về mặt kỹ thuật được chấp nhận của nhà tài trợ/khách hàng.

Quá trình lập kế hoạch xác định cần phải làm gì? Làm như thế nào? Ai là người thực hiện và thực hiện khi nào? Mục đích chính của quá trình lập kế hoạch vạch ra phương hướng, quan trọng hơn để tránh các vấn đề rắc rối, rủi ro. Lượng thời gian và chi phí dành cho quá trình lập kế hoạch cần phải tỉ lệ trực tiếp với chi phí tiềm năng từ các lỗi gặp phải do thiếu quá trình lập kế hoạch. Mục tiêu là chi đủ tài nguyên trong quá trình lập kế hoạch nhằm thúc đẩy thực hiện thành công dự án chứ không phải là chi phí quá nhiều tài nguyên đến mức không còn đủ cho quá trình thực hiện dự án. Lập kế hoạch dự án là một quá trình lặp diễn ra bất cứ khi nào có những thay đổi quan trọng trong dự án.

Ví dụ: Khi các nhà tài trợ giao cho thực hiện một dự án phát triển Web, Giám đốc dự án Phong đã rất phấn khởi nhưng anh cũng biết rằng có rất nhiều việc đang ở phía trước bởi vì anh vẫn còn chưa quen với các ứng dụng theo mô hình chủ khách 3 tầng. Anh đã tìm thấy một tài nguyên rất quý đó là một SME về Web làm cho một công ty tư vấn có mối quan hệ rất tốt với công ty anh và anh đã dùng người này vào việc xây dựng các yêu cầu kỹ thuật. Anh đã bỏ ra một số tiền trong ước tính về chi phí để có được sự giúp đỡ từ bên ngoài. Bây giờ là lúc lên lịch cho chuyên gia tư vấn giúp anh tách cấu trúc chi tiết để anh có thể định nghĩa các gói công việc, lập hồ sơ mạng, lên lịch trình dự án và ngân sách chính xác.

2.3.2 Kế hoạch dự án công nghệ thông tin

Cái gì tạo nên một bản kế hoạch dự án? Một bản kế hoạch dự án thực sự bao gồm những gì? Hiểu biết rõ ràng một bản kế hoạch dự án phải bao gồm những gì có thể là vô giá trong việc thiết lập kỳ vọng đối với ban quản lý dự án.

Định nghĩa:

Kế hoạch dự án CNTT (IT Project Plan) là một tài liệu dự án chính thức do người quản lý dự án, nhà tài trợ, các đối tượng liên quan đến dự án và các thành viên đội dự án xây dựng với mục đích giám sát việc thực hiện dự án. Kích thước và mức độ chi tiết của bản kế hoạch dự án phải tương ứng với loại dự án. Một bản kế hoạch dự án tối thiểu phải có những thành phần quan trọng sau:

- Bản đề xuất dự án (Project Proposal)

- Tôn chỉ dự án/ Bản công bố dự án (Project Charter)
- Tuyên bố về phạm vi (Scope Statement)
- Cấu trúc chi tiết công việc (Work Breakdown Structure)
- Ma trận trách nhiệm (Responsibility Matrix) đối với kết quả chuyển giao chính.
- Ma trận phân bố tài nguyên cho việc thống kê các kỹ năng, nguyên vật liệu và cơ sở vật chất cần thiết.
- Ước tính chi phí và thời gian cho dự án.
- Lịch trình của dự án, bao gồm cả ngày tháng cho tất cả các mốc quan trọng.
- Ngân sách của dự án, bao gồm cả chi phí cơ sở (Cost Baselines)
- Kế hoạch quản lý rủi ro.
- Kế hoạch truyền thông. (Communication Plan)

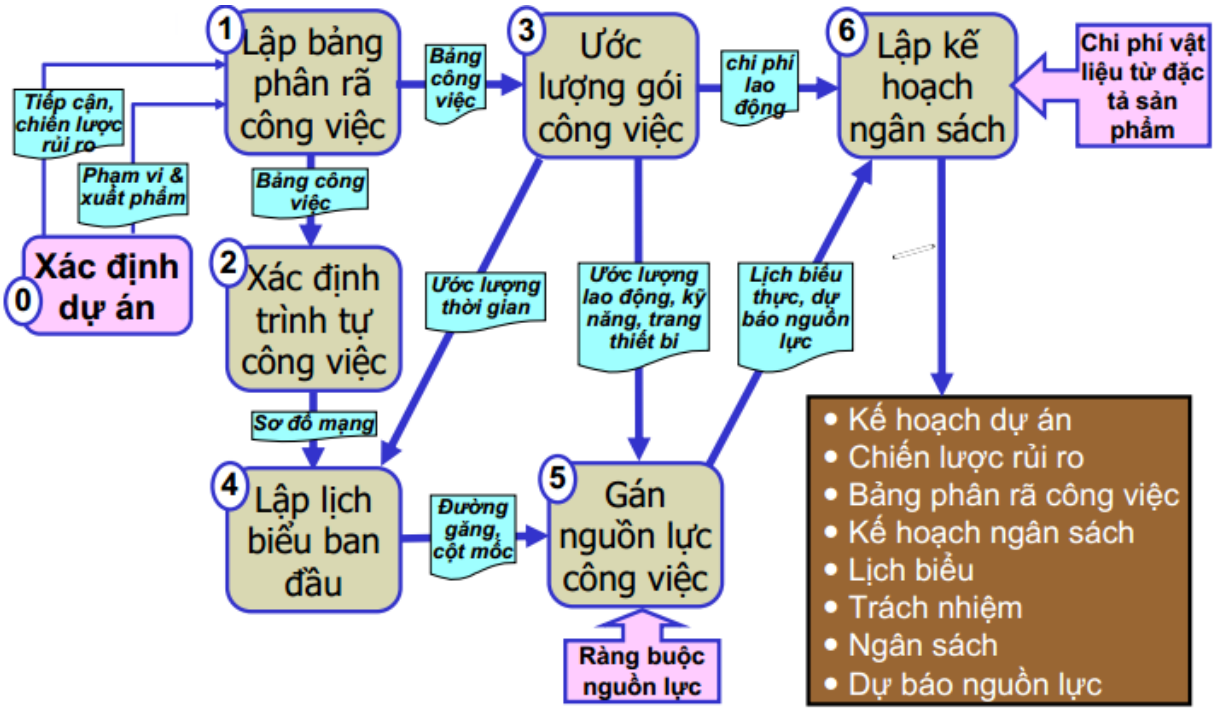
Những Dự án CNTT có quy mô lớn và độ phức tạp cao đòi hỏi phải có kế hoạch chi tiết hơn cho những lĩnh vực cụ thể. Những kế hoạch quản lý chi tiết này bao gồm:

- Kế hoạch quản lý phạm vi
- Kế hoạch quản lý lịch trình
- Kế hoạch kiểm thử hay quản lý chất lượng
- Kế hoạch quản lý nhân sự
- Kế hoạch quản lý mua sắm.
- Kế hoạch phản ứng trước rủi ro hoặc đối phó với những bất ngờ trong dự án.

Không phải mọi kế hoạch dự án đều đòi hỏi phải có những yếu tố này mới đạt tính hiệu quả. Đối với những dự án lớn và phức tạp nên có tất cả những yếu tố này.

2.3.3 Triển khai kế hoạch dự án

Sau khi xác định dự án sẽ đạt tới cái gì? Thì bước tiếp theo là xác định cách nó sẽ hoàn thành các mục đích và mục tiêu đó như thế nào. Cách thức để hoàn thành mục đích và mục tiêu là cần thực hiện:



Hình 2.11: Tiến trình triển khai kế hoạch dự án

Cấu trúc chi tiết công việc (WBS): WBS là bản thiết kế phân cấp các sản phẩm, sản phẩm phụ, nhiệm vụ và nhiệm vụ con cần để hoàn thành dự án. Một WBS tốt cung cấp cơ sở cho việc xây dựng các ước lượng thời gian và chi phí có ý nghĩa cũng như lịch biểu khả thi.

Ước lượng thời gian: Người quản lý dự án có thể áp dụng các ước lượng thời gian theo các nhiệm vụ và nhiệm vụ con được nhận diện trong WBS. Có một số kĩ thuật ước lượng có thể được áp dụng tùy theo mức độ tin cậy ước lượng.

Lịch biểu: Người quản lý dự án có thể phân tích từ WBS và các ước lượng thời gian để xây dựng lịch biểu. Trước hết nhận diện mối quan hệ logic giữa các nhiệm vụ rồi áp dụng các ước lượng thời gian cho các nhiệm vụ đó, tiếp theo tính ngày tháng cho từng nhiệm vụ và lưu ý tới các ràng buộc của nó đối với dự án. Qua việc tính toán lịch biểu, người quản lý dự án nhận diện các nhiệm vụ cần cho việc hoàn thành dự án đúng hạn.

Việc xây dựng **Lịch biểu đích** và **Lịch biểu hiện thời** là tính năng quan trọng khác. Lịch biểu đích hay vạch ranh giới, là lịch biểu bạn đồng ý theo đuổi. Lịch biểu hiện thời là sự tổ hợp của lịch biểu đích và hiện trạng so với các khoản mục được chứa trong đó. Lịch biểu hiện thời cũng dự kiến khi nào các hoạt động hiện thời và tương lai sẽ bắt đầu hay kết thúc, hay cả hai nếu dự án tiếp tục như nhịp độ hiện tại của nó. Người quản lý dự án theo đuổi bất kỳ cái gì cần thiết để so sánh lịch biểu hiện thời với lịch biểu đích. Ngẫu nhiên, phần lớn các dự án đều cho phép bạn sửa đổi lịch biểu đích bất kỳ khi nào cần thiết.

Phân bổ tài nguyên: Các dự án đều tiêu thụ tài nguyên như con người, vật tư, trang thiết bị và không gian làm việc. Người quản lý dự án phải phân bố tài nguyên cho các nhiệm vụ để hoàn thành chúng. Sau khi dùng các tài nguyên, người quản lý dự án có thể xác định liệu tài nguyên có đủ để hoàn thành việc chuyển giao sản phẩm hay không?

Tính chi phí: Sau khi tạo ra cấu trúc phân việc, xác định các ước lượng thời gian; Xây dựng lịch biểu; Và cấp phát tài nguyên, người quản lý dự án có thể tính toán chi phí để thực hiện từng nhiệm vụ và cho toàn bộ dự án. Chi phí được ước lượng cuối cùng trở thành ngân sách. Trong khi thực hiện dự án, người quản lý theo dõi hiệu năng chi phí so với ngân sách.

Kiểm soát rủi ro: Không dự án nào hoạt động độc lập, mà phải nằm trong một môi trường có tổ chức. Nên luôn có những rủi ro, mối đe dọa, ảnh hưởng tới sự thực hiện và sự thành công của dự án. Người quản lý dự án phải xác định những rủi ro này và xây dựng một kế hoạch hiện thực để giảm thiểu tác động.

2.3.4 Thực thi kế hoạch

Có thể nói lập kế hoạch dự án tốt là chìa khóa mở ra thành công cho dự án, thì quá trình thực hiện dự án là cơ hội để người quản lý dự án thể hiện vai trò quản lý của mình. Các dự án CNTT thường rất phức tạp, nhưng không phải vì thế mà khó kiểm soát được. Người quản lý dự án phải:

- Chủ động giám sát dự án
- Vạch ra chiến lược theo dõi sự ủng hộ của nhà tài trợ và cổ đông
- Vạch ra chiến lược cho các kênh truyền thông
- Vạch ra chiến lược cho các phương pháp và tiêu chuẩn
- Vạch ra chiến lược giám sát việc tuân thủ các quy tắc đề ra
- Xác lập một hệ thống thông tin để theo dõi
- Xác định được các dấu hiệu rủi ro

2.4 Case Study: Lập kế hoạch dự án phần mềm

a) Mục tiêu

- Phát biểu được bài toán (case study)

- Lập kế hoạch quản lý dự án

b) Yêu cầu

1. Giới thiệu thông tin nhóm thực hiện

2. Đặt vấn đề, Phát biểu được bài toán

3. Khảo sát hiện trạng

4. Đề xuất hệ thống thực hiện

5. Lập kế hoạch quản lý dự án

c) Hướng dẫn, gợi ý: [33 – Chapter 1,2]

Câu hỏi

Câu 1. Bước đầu tiên trong vòng đời phát triển phần mềm (Software Development Life Cycle) là?

a) Xác định các nhu cầu và ràng buộc
b) Viết phần mềm
c) Vận hành hệ thống để loại bỏ các khiếm khuyết
d) Nâng cao sản phẩm sau khi đã triển khai

Câu 2. Mô hình bản mẫu (prototyping model) của phát triển phần mềm là …

a) Một cách tiếp cận hợp lý khi yêu cầu được định nghĩa rõ ràng
b) Một cách tiếp cận hữu ích khi khách hàng không thể định nghĩa yêu cầu rõ ràng
c) Cách tiếp cận tốt nhất cho những dự án có đội phát triển lớn
d) Tất cả các phương án trên đều sai

Câu 3. Kỹ thuật nào sau đây là xây dựng phần mềm từ các thành phần đã được thiết kế trong lĩnh vực công nghệ khác nhau?

a) Extreme programming
b) Evolutionary prototyping
c) Component architecture
d) Open-source development

Câu 4. Tính khả thi của dự án phần mềm dựa vào các yếu tố nào?

a) Nghiệp vụ và tiếp thị
b) Phạm vi và thị trường
c) Công nghệ, chi phí, thời gian và tài nguyên
d) Năng lực của nhà phát triển

Câu 5. Điều nào là đặc trưng của một thiết kế phần mềm tốt?

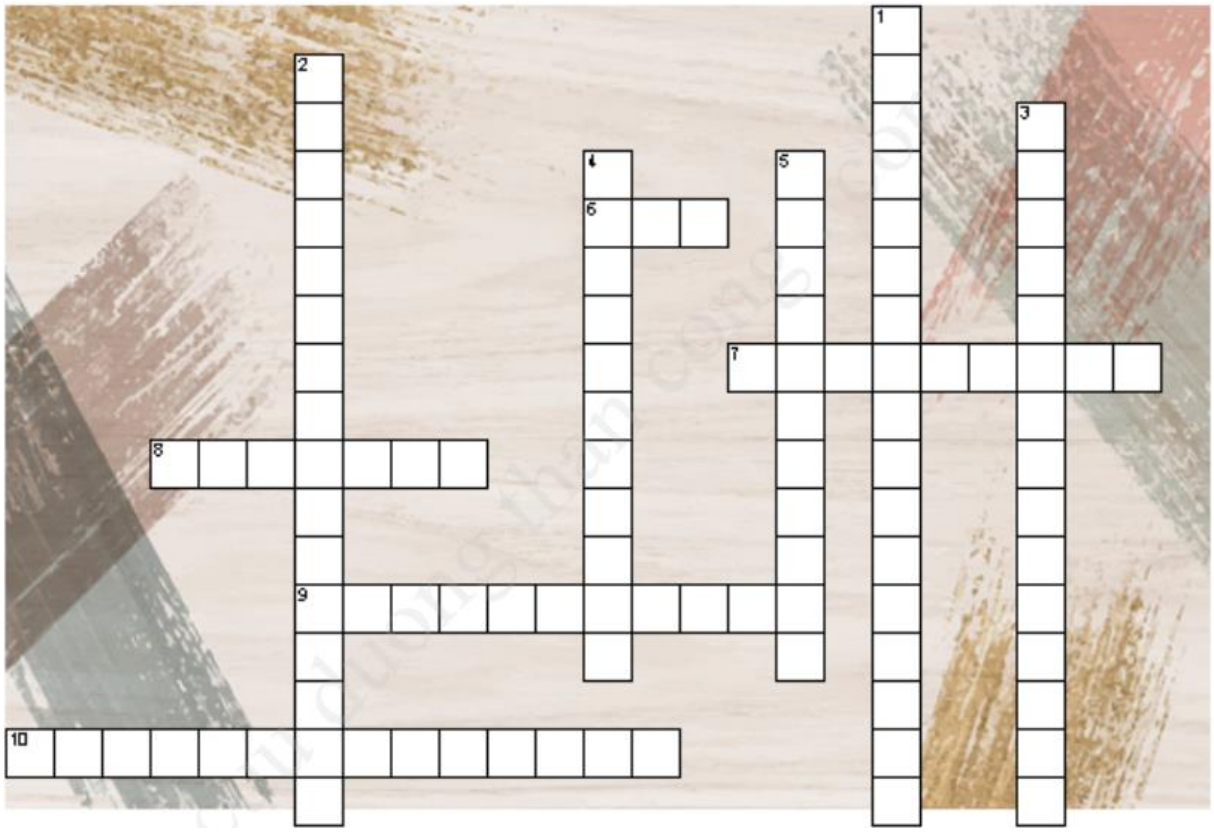
a) Thể hiện kết nối mạnh mẽ giữa các mô-đun của nó
b) Thực hiện tất cả các yêu cầu trong mô hình phân tích
c) Bao gồm các trường hợp thử nghiệm cho tất cả các thành phần
d) Cung cấp một bức tranh hoàn chỉnh của phần mềm

Bài tập cuối chương

Bài 2.1. Hãy so sánh các mô hình phát triển phần mềm: Thác nước, Chữ V, Mẫu thử, Xoắn ốc, Agile

	Đặc điểm chính	**Ưu điểm**	**Nhược điểm**	**Trường hợp áp dụng phù hợp**
Mô hình thác nước				
Mô hình chữ V				
Mô hình mẫu thử				
Mô hình xoắn ốc				
Mô hình Agile				

Bài 2.2. Hãy giải ô chữ dưới đây với các gợi ý kèm theo.



Các gợi ý cho ô chữ:

1. Thời kỳ tính từ khi phần mềm được sinh ra cho đến khi chết đi

2. Sự kết hợp mô hình tuần tự và ý tưởng lặp lại của chế bản mẫu

3. Tập hợp các hành động, phương thức, thực hành, thay đổi mà người ta dùng để duy trì và phát triển phần mềm và các thành phần liên quan

4. Mô hình có thể sử dụng như "hệ sơ khai" để thu thập yêu cầu

5. Rủi ro được xem xét kỹ ở mỗi lần lặp tiến trình

6. Mô hình phát triển tăng dần từng bước, chia thành các team thực hiện đầy đủ các pha. Mỗi chu trình phát triển rất ngắn từ 60-90 ngày.

7. Các yêu cầu có tính ổn định cao. Các pha của tiến trình phát triển thực hiện tuần tự từng bước, không quay trở lại pha trước đó.

8. Giai đoạn kiểm tra logic bên trong và chức năng bên ngoài để đảm bảo kết quả đầu ra chính xác.

9. Giai đoạn đáp ứng những thay đổi, nâng cấp phần mềm đã phát triển do sự thay đổi của môi trường, nhu cầu.

10. Có ưu điểm là khả năng tái sử dụng phần mềm thông qua các hoạt động: Phân tích yêu cầu, thiết kế với sử dụng lại, phát triển và tích hợp, thẩm định.

Bài 2.3. Hãy so sánh mô hình thác nước (Waterfall modell) và Scrum

	Thác nước	**Scrum**
Đặc điểm chính		
Đặc điểm các pha phát triển		
Kích thước nhóm phát triển (Team size)		
Phong cách quản lý (Management style)		
Quan điểm về sự thay đổi trong dự án		
Vấn đề xây dựng tài liệu		
Đối phó với các rủi ro (risk)		
Trường hợp áp dụng phù hợp		

Bài 2.4. Nếu không áp dụng các mô hình vòng đời phần mềm thì có phát triển được phần mềm không? Tại sao?

Bài 2.5. Công nghệ phần mềm phân tán là gì? Tại sao phải phát triển phần mềm phân tán? Lấy ví dụ minh họa về các hệ thống phần mềm phân tán.

Bài 2.6. Vòng đời phần mềm? Hãy mô tả các giai đoạn trong vòng đời phát triển phần mềm.

Bài 2.7. Tiến hoá phần mềm là gì? Tại sao cần tiến hoá phần mềm?

Bài 2.8. Việc xác định mô hình quy trình phần mềm khi thực hiện dự án có ý nghĩa gì?

Bài 2.9. Thế nào là một phần mềm tốt?

Bài 2.10. Sự khác biệt giữa mô hình Thác nước và mô hình Agile?

Chương III: KỸ NGHỆ YÊU CẦU - RE

Nội dung chính của chương

3.1 Tổng quan về kỹ nghệ yêu cầu

3.2 Yêu cầu phần mềm

3.3 Phát triển tập yêu cầu

3.4 Quản lý yêu cầu

Mục tiêu cần đạt được của chương

- Biết đặc tả mục tiêu, yêu cầu của dự án
- Biết thu thập yêu cầu dựa trên các phương pháp kỹ thuật và công cụ thu thập phân loại yêu cầu
- Áp dụng được các kiến thức phân tích đặc tả yêu cầu phần mềm để giải quyết các vấn đề trong quy trình phát triển phần mềm

Bài 5: Giới thiệu về Kỹ nghệ yêu cầu (Số tiết: 03 tiết)

3.1 Tổng quan về kỹ nghệ yêu cầu (RE) [2]

RE là quy trình xác định mục tiêu dự án; định nghĩa; lập tài liệu yêu cầu; và quản lý các yêu cầu trong suốt tiến trình dự án. Nó cung cấp cơ chế thích hợp để phân tích viên hiểu được các mong đợi của khách hàng và phân tích nhu cầu của họ; đánh giá tính khả thi; thương lượng giải pháp hợp lý với khách hàng; đặc tả giải pháp đề xuất cho việc giải quyết các vấn đề của khách hàng một cách rõ ràng; thẩm định các đặc tả và quản lý các yêu cầu khi chúng được chuyển dịch vào trong hệ thống làm việc.

Do đó, kỹ nghệ yêu cầu là việc áp dụng một cách có nguyên tắc các nguyên lý, các phương pháp, công cụ và ký hiệu đã được phê chuẩn để mô tả hành vi của hệ thống cần xây dựng & các ràng buộc được kết hợp với dự án SE.

Tiến trình RE thường chia làm 2 giai đoạn *(1) Phát triển yêu cầu; (2) Quản lý yêu cầu.*

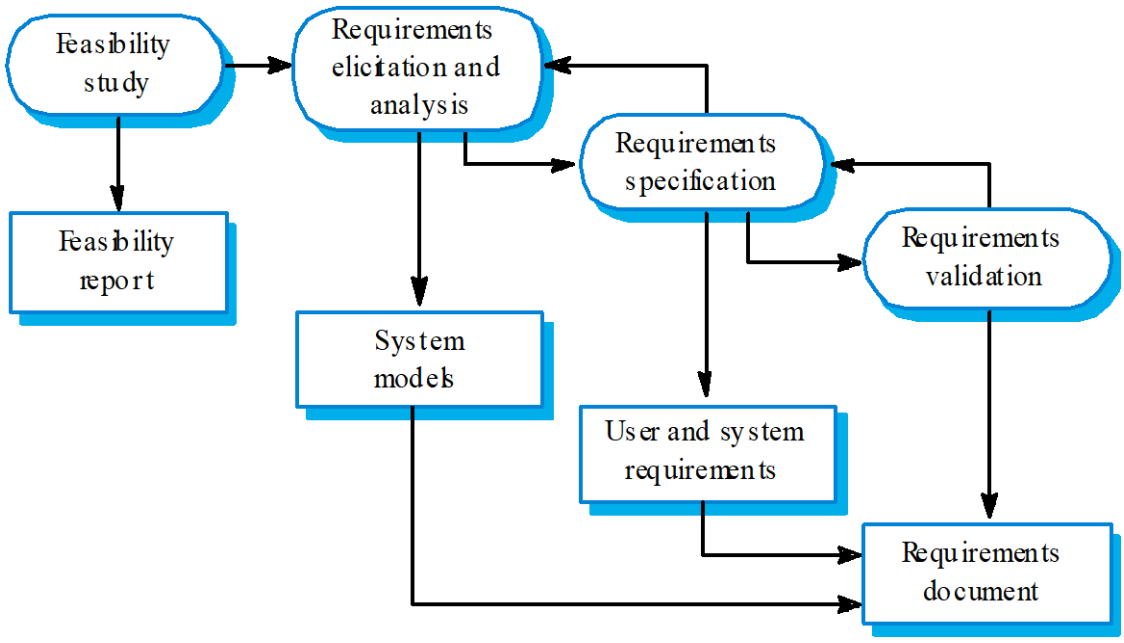
- **Phát triển yêu cầu**:

  Để phát triển tập yêu cầu, chúng ta thường trải qua 4 bước:

  1) Thu thập yêu cầu
  2) Phân tích yêu cầu
  3) Đặc tả yêu cầu
  4) Thẩm định yêu cầu

Các hoạt đông này được tiến hành một cách lặp lại nhằm phát hiện vấn đề về yêu cầu, bổ sung các thiếu xót và giải quyết các xung đột liên quan đến yêu cầu. Ngoài 4 hoạt động trên, nghiên cứu tính khả thi của dự án cũng được xem là một hoạt động thuộc quy

trình kỹ nghệ yêu cầu. Quy trình phát triển tập yêu cầu đảm bảo chất lượng được chỉ ra trong Hình 3.1



Hình 3.1: Quy trình phát triển tập yêu cầu

❖ **Quản lý yêu cầu**:

Quản lý yêu cầu là các hoạt động nhằm thiết lập và bảo trì tập các yêu cầu đã phát triển được trong suốt thời gian sống của phần mềm. Quy trình này bao gồm:

(1) Giám sát những thay đổi về yêu cầu sau khi chúng được cố định và ký kết. Các thay đổi về yêu cầu sau đó nếu được đề xuất cần tiến hành theo một quy trình nhất định.

(2) Đảm bảo các yêu cầu thống nhất với kế hoạch triển khai và các sản phẩm công việc cần tạo.

(3) Thương lượng các yêu cầu mới với khách hàng dựa trên sự ảnh hưởng của chúng đối với hệ thống.

3.2 Yêu cầu phần mềm

3.2.1 Khái niệm

Yêu cầu là một phát biểu/mô tả/đặc tả về dịch vụ mà hệ thống cung cấp, hoặc mô tả về một ràng buộc/điều kiện/phụ thuộc mà hệ thống phải thoả mãn; hoặc cũng có thể là một mục tiêu mà hệ thống phần mềm phải đạt được [1] [4].

Yêu cầu có thể được đặc tả/mô tả bằng các công cụ khác nhau như ngôn ngữ tự nhiên, ngôn ngữ tự nhiên có cấu trúc, ngôn ngữ mô hình, ngôn ngữ hình thức...

Ví dụ : Xét phần mềm quản lý hồ sơ sinh viên của một trường đại học. Dưới đây là một yêu cầu được phát biểu bằng ngôn ngữ tự nhiên về một dịch vụ mà phần mềm phải cung cấp:

REQ1: "Phần mềm phải lưu trữ được mọi thông tin cần quản lý của sinh viên từ khi nhập học cho đến khi tốt nghiệp ra trường được 10 năm".

❖ **Tầm quan trọng của yêu cầu**

Các yêu cầu phần mềm giữ vị trí quan trọng đối với dự án phần mềm. Cụ thể:

- Làm cơ sở cho việc mời thầu: đối với các dự án triển khai ý tưởng tự đề xuất; do đó chúng cần được phát biểu sao cho dễ đọc, dễ hiểu, có giải thích rõ ràng (viết cho khách hàng, đối tác);
- Làm cơ sở ký kết hợp đồng thầu => Cần phát biểu đầy đủ, chi tiết, có khả năng triển khai và nghiệm thu (viết cho khách hàng);
- Làm tài liệu đầu vào cho thiết kế và triển khai => Cần phát biểu đủ chi tiết, rõ ràng, các thể dùng các kiến thức chuyên ngành để đặc tả (viết cho developers);
- Làm cơ sở đánh giá, nghiệm thu dự án: viết cho testers and QA: phải đảm bảo mọi yêu cầu có khả năng nghiệm thu.

Như vậy, một yêu cầu có thể được phân tích làm mịn từ một phát biểu ở mức độ trừu tượng cao (bằng ngôn ngữ tự nhiên) → một đặc tả toán học chi tiết (đặc tả hình thức).

Yêu cầu cần phát biểu ở mức độ trừu tượng cao vì (1) chúng là cơ sở cho việc mời thầu – do đó nó phải không chứa các chi tiết kỹ thuật, phải dễ đọc, dễ hiểu đối với tất cả các đối tượng người đọc có trình độ khác nhau (dành cho khách hàng, người dùng và đối tác) ; (2) là cơ sở để ký kết hợp đồng đấu thầu – do đó phải được định nghĩa chi tiết (đủ chi tiết để ký kết hợp động và nghiệm thu). Tuy nhiên, khi bắt tay vào triển khai hệ thống, các yêu cầu, đặc biệt là các yêu cầu chức năng cần được đặc tả chi tiết hơn, hình thức hơn, có thể chứa các chi tiết kỹ thuật (đặc tả bán hình thức) để làm đầu vào cho giai đoạn thiết kế và triển khai hệ thống một cách dễ dàng (đối tượng người đọc là developers) ; hoặc làm cơ sở để tạo sinh thiết kế/code tự động (đặc tả hình thức). Do đó, khi đặc tả/phát biểu các yêu cầu, chúng phải được phát biểu tối thiểu ở 2 mức độ trừu tượng:

- *Yêu cầu người dùng (user requirememts);*
- *Yêu cầu hệ thống (System requirements).*

Từ một yêu cầu người dùng, có thể phân tích và ánh xạ thành nhiều yêu cầu hệ thống chi tiết hơn. Ví dụ:

User requirement

1. The software shall provide a means of representing and accessing external files created by other tools.

System requirements

1.1 The user should be provided with facilities to define the type of external files.
1.2 Each external file type may have an associated tool which may be applied to the file.
1.3 Each external file type may be represented as a specific icon on the user's display.
1.4 Facilities should be provided for the icon representing an external file type of be defined by the user.
...

Yêu cầu người dùng là những yêu cầu được phát biểu bằng ngôn ngữ tự nhiên bởi người sử dụng, thể hiện ở mức độ trừu tượng những công việc mà người sử dụng muốn hệ thống phải đáp ứng. Tài liệu mô tả yêu cầu của người dùng nên nói rõ những yêu cầu chức năng và phi chức năng để người sử dụng có thể hiểu được chúng mà không cần phải có những kiến thức về công nghệ một cách chi tiết, được định nghĩa bằng cách sử dụng ngôn ngữ tự nhiên, bảng hoặc biểu đồ đơn giản. Tuy nhiên, chúng ta sẽ gặp phải một số khó khăn khi sử dụng ngôn ngữ tự nhiên:

- *Không rõ ràng*: Tính chính xác rất khó đạt được nếu tài liệu khó đọc.

- *Yêu cầu lộn xộn*: các yêu cầu chức năng và phi chức năng không rõ ràng.

- *Lẫn lộn giữa các yêu cầu*: các yêu cầu khác nhau có thể được diễn tả cùng với nhau.

Do đó, để viết yêu cầu của người dùng ta nên áp dụng một số quy tắc sau:

- Đưa ra một định dạng chuẩn và áp dụng nó cho tất cả các yêu cầu.

- Bắt buộc sử dụng ngôn ngữ một cách thống nhất.

- Đánh dấu những phần quan trọng trong các yêu cầu.

- Tránh sử dụng những từ ngữ mang tính chuyên môn, kỹ thuật.

3.2.2 Phân loại yêu cầu hệ thống

Các yêu cầu của hệ thống thường được chia thành ba loại:

- *Yêu cầu chức năng (Functional Requirement);*
- *Yêu cầu phi chức năng (Non – functional Requirement);*
- *Yêu cầu miền ứng dụng (Domain Requirement).*

❖ **Yêu cầu chức năng**

Yêu cầu chức năng là các phát biểu phản ánh trực tiếp các chức năng/dịch vụ mà hệ thống sẽ cung cấp. Ví dụ: một số yêu cầu chức năng của hệ thống LYBSYS

- *REQ1: "Người sử dụng có thể **tìm kiếm** tài liệu theo các thông tin được quản lý trong Cơ sở dữ liệu của hệ thống"*
- *REQ2: "Hệ thống sẽ cung cấp các chức năng để người sử dụng **đọc/xem** tài liệu, **download, in** tài liệu,…"*
- *REQ3: "Tất cả những hoá đơn mà người dùng **đăng ký** để **in/sao chép** tài liệu có một mã duy nhất."*

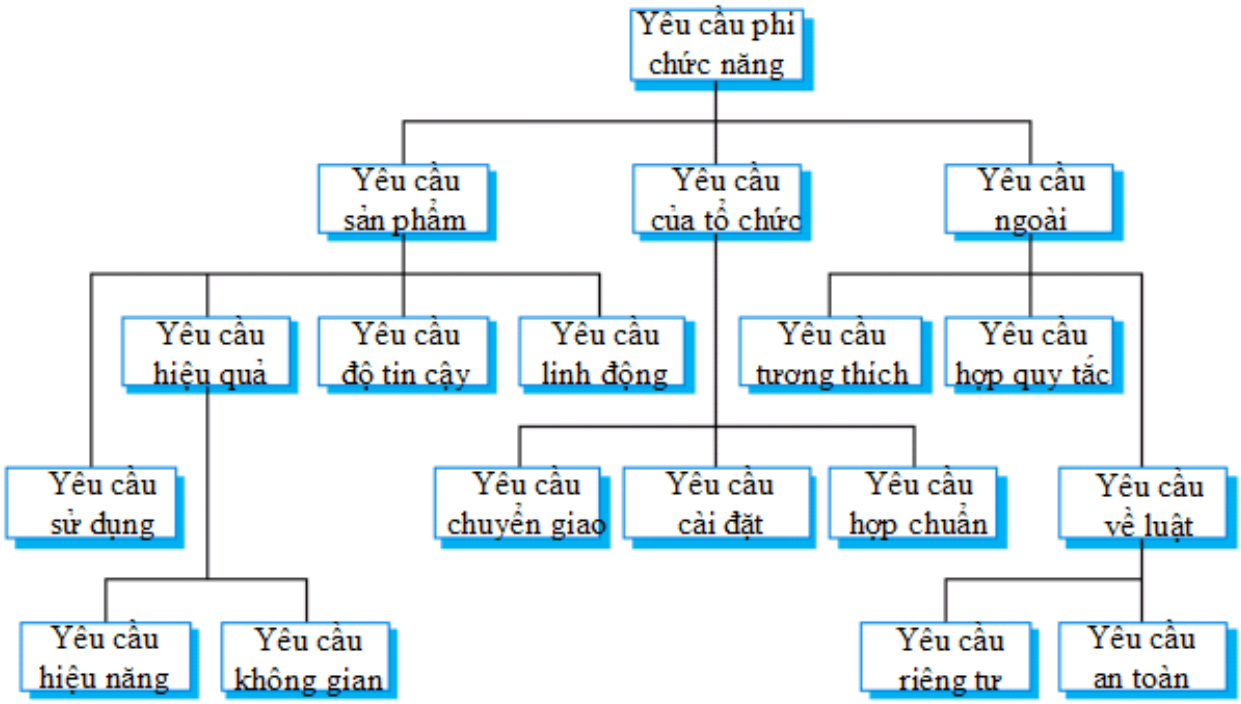
=> Chúng phản ánh trực tiếp các chức năng mà LYBSYS cần cung cấp gồm: ***Tìm kiếm tài liệu; hiển thị nội dung tài liệu; download tài liệu, in tài liệu; đặng ký người dùng, in tài liệu, copy tài liệu.***

❖ **Yêu cầu phi chức năng**

Yêu cầu phi chức năng không đề cập trực tiếp tới các chức năng cụ thể của hệ thống. Chúng thường mô tả các thuộc tính chất lượng của hệ thống như: độ tin cậy, thời gian phản hồi, yêu cầu về không gian lưu trữ; hoặc mô tả các ràng buộc của hệ thống như: khả năng của thiết bị vào/ra, giao diện tương tác, kinh phí, công nghệ sử dụng, … hoặc mô tả các mục tiêu cần đạt được của hệ thống; một số yêu cầu phi chức năng còn có liên quan đến quy trình xây dựng, triển khai hệ thống, ví dụ: các chuẩn được sử dụng, các công cụ CASE, ngôn ngữ lập trình …

Các yêu cầu phi chức năng có thể nẩy sinh từ: (1) các yêu cầu về chất lượng sản phẩm; (2) các yêu cầu từ phía tổ chức người dùng sản phẩm; (3) từ môi trường vận hành sản phẩm. Do đó, căn cứ vào nguồn gốc phát sinh, chúng ta có thể phân loại các yêu cầu phi chức năng thành 3 loại (chi tiết xem Hình 3.2) như sau:

1) Các yêu cầu về sản phẩm: hiệu năng, khả năng sử dụng, độ tin cậy …
2) Các yêu cầu về tổ chức: các yêu cầu này nẩy sinh từ các chính sách và quy tắc của khách hàng hoặc tổ chức sử dụng hệ thống.
3) Các yêu cầu ngoài: được xác định từ môi trường vận hành sản phẩm, từ các tác nhân ngoài tương tác với hệ thống.



Hình 3.2: Phân loại các yêu cầu phi chức năng

Ví dụ: Một số yêu cầu phi chức năng của hệ thống LIBSYS

- *REQ 4: Yêu cầu về sản phẩm: "LIBSYS phải được cài đặt bằng HTML mà không có frame hoặc Java applets".*
- *REQ5: Yêu cầu về mặt tổ chức: "Quy trình xây dựng hệ thống và các tài liệu chuyển giao phải thoả mãn các quy tắc đã được định nghĩa trong XYZCo-SP-STAN-95".*
- *REQ6: Yêu cầu ngoài: "Hệ thống không được để lộ các thông tin cá nhân của khách hàng".*

Các yêu cầu phi chức năng thường khó thẩm định để nghiệm thu. Do đó, để đảm bảo có thể nghiệm thu được, chúng nên được phát biểu một cách có định lượng (có đơn vị đo, đếm). Bảng 3.1 là danh sách các phép đo tham khảo. Ví dụ:

- Yêu cầu ban đầu: *"Phần mềm phải dễ dàng sử dụng";*
- Phát biểu lại thành có định lượng: "*Người dùng có thể sử dụng mọi chức năng của hệ thống sau 2 tiếng huấn luyện và số lỗi mắc phải không quá 2 lỗi/1 ngày.*"

Như vậy tính dễ sử dụng được xác định thông qua 2 phép đo: *(1) thời gian huấn luyện; (2) số lỗi người dùng mắc phải sau khi được tập huấn sử dụng phần mềm.*

Bảng 3.1: Các đơn vị đo chất lượng phần mềm

Property	Measure
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	M Bytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

❖ **Yêu cầu miền**

Yêu cầu miền là các yêu cầu được xác định từ miền ứng dụng mà hệ thống hỗ trợ, thường chứa các thuật ngữ chuyên ngành, mang tính kỹ thuật thuộc lĩnh vực ứng dụng do đó thường khó hiểu đối với đội dự án. Các yêu cầu này có thể là yêu cầu chức năng và phi chức năng và chúng được phân loại riêng để developers cần thận trọng và phải hiểu kỹ về chúng; cũng như thu thập đầy đủ các yêu cầu này từ phía khách hàng và đối tác để tránh bị bỏ sót dẫn đến hệ thống không vận hành được.

Ví dụ: Một số yêu cầu miền ứng dụng của LIBSYS:

- *REQ7: "Giao diện người dùng chuẩn cho mọi cơ sở dữ liệu cần dựa trên chuẩn Z39.50".*
- *REQ8: "Vì các hạn chế về bản quyền, một số tài liệu chỉ được in trên máy chủ hệ thống và được chuyển đến người dùng.*

Hoặc ví dụ về yêu cầu miền của hệ thống lái tầu tự động:

- *REQ1: "Hệ thống điều khiển tàu phải tính đến các đặc tính phanh tầu trong các điều kiện thời tiết khác nhau"*
- *REQ1: "Sự giảm tốc độ của tầu được tính theo công thức:* $D_{train} = D_{control} + D_{gradient}$ *. Trong đó:* $D_{gradient} = 9.81ms^2 * compensated\ gradient/alpha$ *và giá trị* $9.81ms^2/alpha$ *áp dụng cho các kiểu tầu khác nhau."*

3.3 Phát triển tập yêu cầu [2]

Như đã đề cập tại mục 3.1, để phát triển tập yêu cầu, chúng ta thường trải qua quy trình gồm 4 bước như mô tả trong Hình 3.1. Kết quả thu được là tài liệu đặc tả yêu

cầu. Vì yêu cầu tối thiểu cần đặc tả ở hai mức: mức người dùng và mức hệ thống, do đó cần xây dựng 2 loại tài liệu đặc tả yêu cầu gồm:

- **Tài liệu đặc tả yêu cầu người dùng** – Làm hợp đồng ký kết với khách hàng và làm cơ sở để nghiệm thu dự án. Tài liệu này đôi khi gọi là tài liệu tầm nhìn của dự án – Project Vision Document;
- **Các tài liệu đặc tả các yêu cầu hệ thống** – làm cơ sở để thiết kế và triển khai hệ thống. Tài liệu này đôi khi gọi là tài liệu đặc tả yêu cầu phần mềm (SRS), hoặc các tài liệu đặc tả UC; tài liệu đặc tả bổ sung của dự án (SUPL specification documentation).

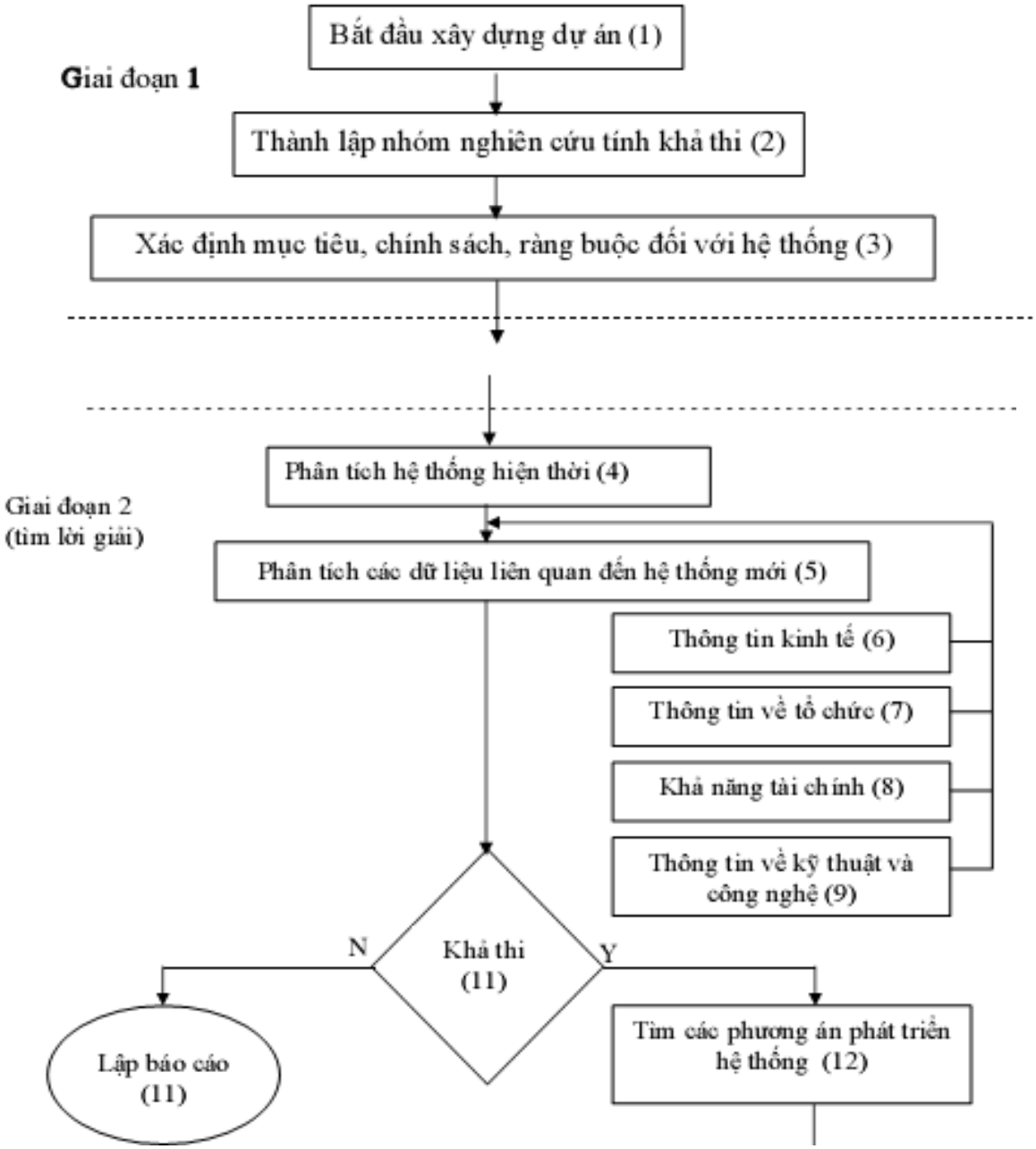
Lưu ý: Đối với các dự án nội bộ, có thể chỉ cần tài liệu SRS. Nếu chỉ có tài liệu SRS thì phải đảm bảo trong nội dung chứa các yêu cầu người dùng.

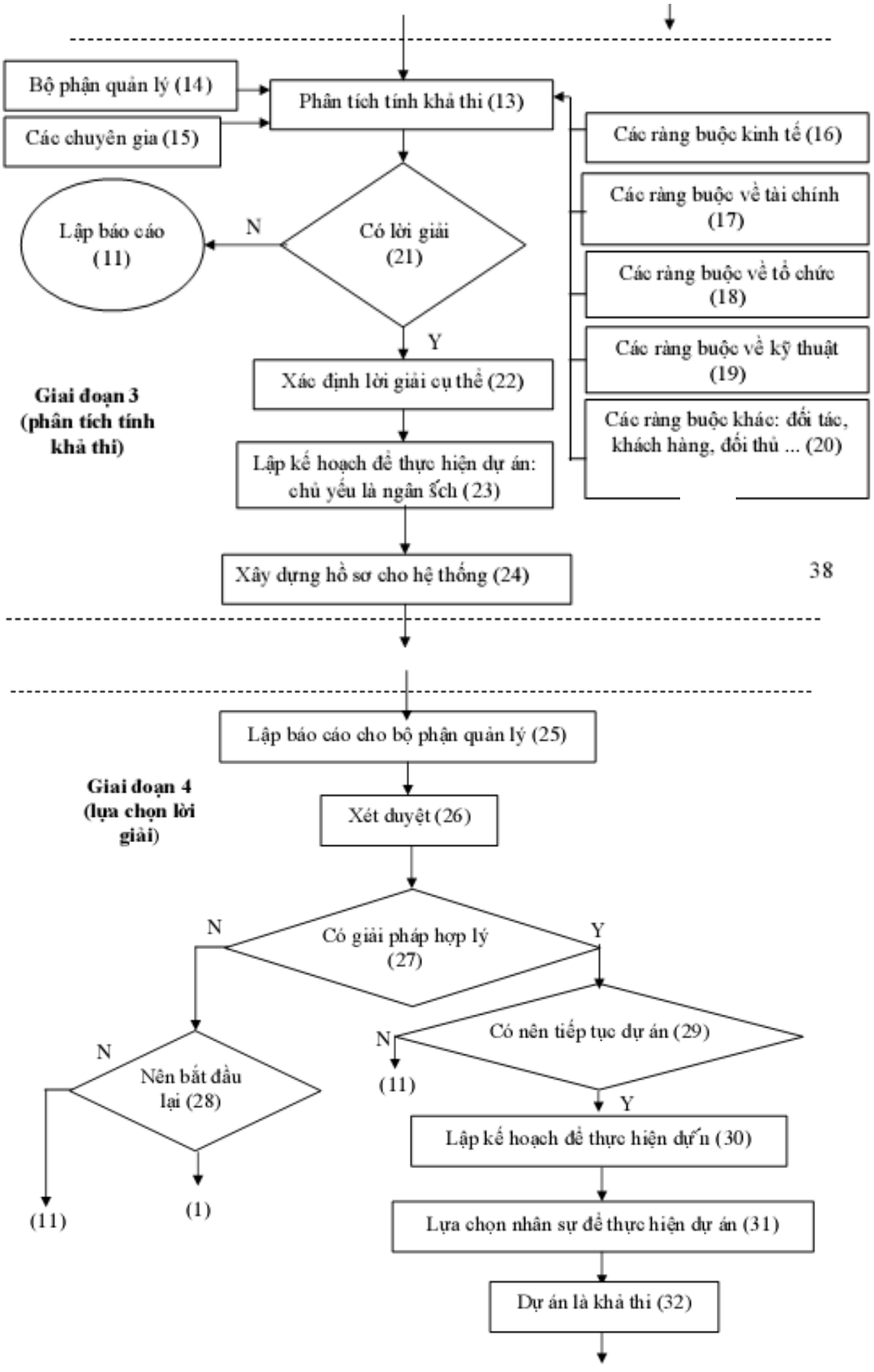
3.3.1 Nghiên cứu tính khả thi dự án

Đối với các dự án phát triển hệ thống phần mềm mới, quy trình xác định yêu cầu thường bắt đầu bằng việc phân tích khả thi. Mục đích của phân tích tính khả thi của dự án là trả lời câu hỏi: "Dự án có khả thi về mặt công nghệ, về kinh tế không?". Hoạt động này cần tiến hành chủ yếu dựa trên kinh nghiệm, và cần thực hiện một cách nhanh chóng và không quá tốn kém. Thông tin đầu vào để phân tích khả thi là các yêu cầu nghiệp vụ, mô tả sơ bộ về hệ thống, cách thức hệ thống hỗ trợ các yêu cầu nghiệp vụ. Kết quả của việc phân tích khả thi là một báo cáo để quyết định có nên xây dựng hệ thống đề xuất hay không.

Thuật toán nghiên cứu tính khả thi của một số dự án SE thường trải qua 4 giai đoạn (chi tiết xem Hình 3.3):

1. Tổ chức nhóm nghiên cứu tính khả thi: giai đoạn 1
2. Tìm kiếm lời giải: giai đoạn 2
3. Phân tích tính khả thi: giai đoạn 3
4. Lựa chọn lời giải: giai đoạn 4


Giai đoạn 1
Bắt đầu xây dựng dự án (1)
Thành lập nhóm nghiên cứu tính khả thi (2)
Xác định mục tiêu, chính sách, ràng buộc đối với hệ thống (3)
Giai đoạn 2
(tìm lời giải)
Phân tích hệ thống hiện thời (4)
Phân tích các dữ liệu liên quan đến hệ thống mới (5)
Thông tin kinh tế (6)
Thông tin về tổ chức (7)
Khả năng tài chính (8)
Thông tin về kỹ thuật và công nghệ (9)
Khả thi (11)
N
Y
Lập báo cáo (11)
Tìm các phương án phát triển hệ thống (12)



Hình 3.3: Thuật toán nghiên cứu tính khả thi của dự án phần mềm

3.3.2 Thu thập, phân tích xác định yêu cầu

❖ **Thu thập yêu cầu**

Giai đoạn này phân tích viên và khách hàng cùng hợp tác để xác định miền ứng dụng, các dịch vụ mà hệ thống cung cấp, hiệu năng của hệ thống, các ràng buộc về hệ thống cần triển khai, ví dụ: môi trường vận hành hệ thống, về chất lượng sản phẩm, …

Ở đây, xuất hiện một khái niệm mới là Stakeholder. Stakeholder tạm dịch là các bên liên quan, hoặc những cá nhân/tổ chức liên quan đến dự án xây dựng hệ thống, hoặc những tổ chức/cá nhân sẽ cung cấp yêu cầu cho dự án, hoặc bị tác động bởi dự án. Họ có thể là những người sử dụng cuối; khách hàng; đối tác đầu tư; các đối thủ cạnh trang; các tác nhân thị trường; các kỹ sư phần mềm; người quản lý; hoặc các chuyên gia lĩnh vực; ….

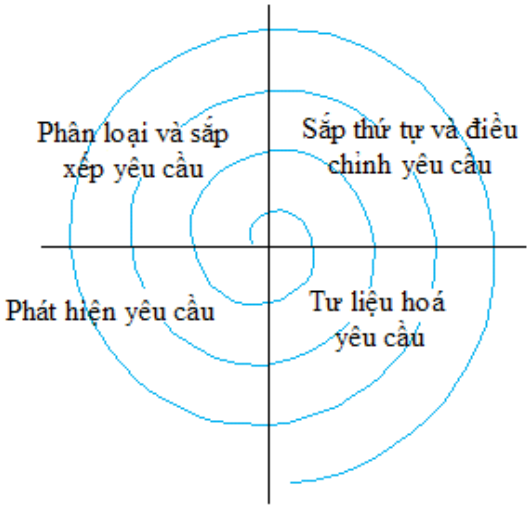
Ví dụ, trong hệ thống ATM gồm các Stakeholder sau: khách hàng của ngân hàng, đại diện của các ngân hàng khác, người quản lý ngân hàng, nhân viên ngân hàng, người quản trị cơ sở dữ liệu của ngân hàng, nhân viên quản lý bảo mật, phòng marketing, kỹ sư bảo trì phần cứng và phần mềm, người điều hành ngân hàng.

Do đó, để thu thập các yêu cầu về hệ thống cần xây dựng, trước hết phân tích viên cần xác định được các stakeholder của dự án. Khách hàng và người dùng cuối của hệ thống là hai loại Stakholder không thể thiếu trong hầu hết các dự án phần mềm. Sau đó sẽ lập kế hoạch để thu thập yêu cầu từ họ.

Trong quá trình thu thập, phát hiện và tìm hiểu yêu cầu của stakeholder, chúng ta thường gặp khó khăn như:

- Stakeholder không biết những gì mà họ thật sự mong muốn.
- Stakeholder mô tả các yêu cầu theo thuật ngữ của họ.
- Những stakeholder khác nhau có thể có các yêu cầu xung đột nhau
- Những yếu tố tổ chức và quyền lực có thể ảnh hưởng tới các yêu cầu hệ thống.
- Yêu cầu được phát biểu từ stakeholder một cách không chắc chắn, có thể thay đổi nhanh chóng trong suốt quá trình thu thập, phân tích yêu cầu.

Mô hình xoắn ốc (Hình 3.4) được khuyến cáo được sử dụng để khắc phục các vấn đề này. Hay nói cách khác, các hoạt động trong giai đoạn này nên tiến hành một cách lặp lại, mỗi lần lặp là một lần chúng ta hoàn thiện/tinh chế các kết quả của lần lặp trước đó và tiến triển bổ sung thêm các yêu cầu mới nếu phát sinh cho lần lặp hiện thời.



Hình 3.4: Quy trình thu thập và phân tích yêu cầu

Trong quy trình xoắn ốc, mỗi vòng lặp xoắn ốc cần tiến hành các công việc:

- ✓ Phát hiện yêu cầu: thu thập yêu cầu từ stakeholder; các yêu cầu miền nên làm rõ và phát hiện ở bước này.
- ✓ Phân loại, sắp xếp yêu cầu: xác định, phân loại, gom nhóm/sắp xếp các yêu cầu thành các nhóm có liên quan và tổ chức chúng thành các nhóm gắn kết.
- ✓ Gán thứ tự ưu tiên và điều chỉnh các yêu cầu xung đột: càng nhiều stakeholder thì khả năng xung đột giữa các yêu cầu của họ càng cao. Hoạt động này nhằm đánh thứ tự ưu tiên của các yêu cầu, phát hiện và giải quyết xung đột giữa các yêu cầu.
- ✓ Tư liệu hóa yêu cầu: viết tài liệu mô tả các yêu cầu đã xác định, chuyển cho stakeholder liên quan đánh giá, hiệu chỉnh và tiếp tục vòng xoắn ốc tiếp theo nếu còn có những vấn đề phát sinh liên quan đến yêu cầu.

Các kỹ thuật/phương pháp có thể sử dụng để thu thập yêu cầu từ Stakeholder:

1. *Phỏng vấn - Interviews*
2. *Bảng câu hỏi thăm dò – Questionnaires*
3. *Hội thảo – Workshops*
4. *Thẻ sự kiện - Storyboarding*
5. *Phân vai - Role playing*
6. *Phiên làm việc tập trung - Brainstorming sessions*
7. *Mẫu thử - Prototyping*
8. *Trường hợp sử dụng & Kịch bản sử dụng- Use cases & scenarios.*
9. *Phân tích các tài liệu - Analysis of existing documents*
10. *Quan sát & Minh họa nhiệm vụ - Observation & Task demonstration*
11. *Phân tích các hệ thống đang tồn tại - Analysis of existing systems*
12. *Khung nhìn - Viewpoint*

Mỗi kỹ thuật thu thập yêu cầu đều có các ưu điểm và nhược điểm riêng và phù hợp với ngữ cảnh thu thập yêu cầu nhất định. Ví dụ: để thu thập các yêu cầu về giao diện chương trình, thẻ sự kiện là kỹ thuật tối ưu; để thương lượng, giải quyết các yêu cầu xung đột, phiên làm việc tập trung là lựa chọn tốt nhất; với dự án lớn, phức tạp, có nhiều đối tác cần gặp gỡ và trao đổi trực tiếp, yêu cầu mâu thuẫn, … thì hội thảo có thể là một lựa chọn cần cân nhắc để tiết kiệm thời gian…

Việc lựa chọn kỹ thuật thu thập yêu cầu phù hợp với Stakeholder sẽ mang lại nhiều lợi ích và thuận lợi cho phân tích viên hệ thống.

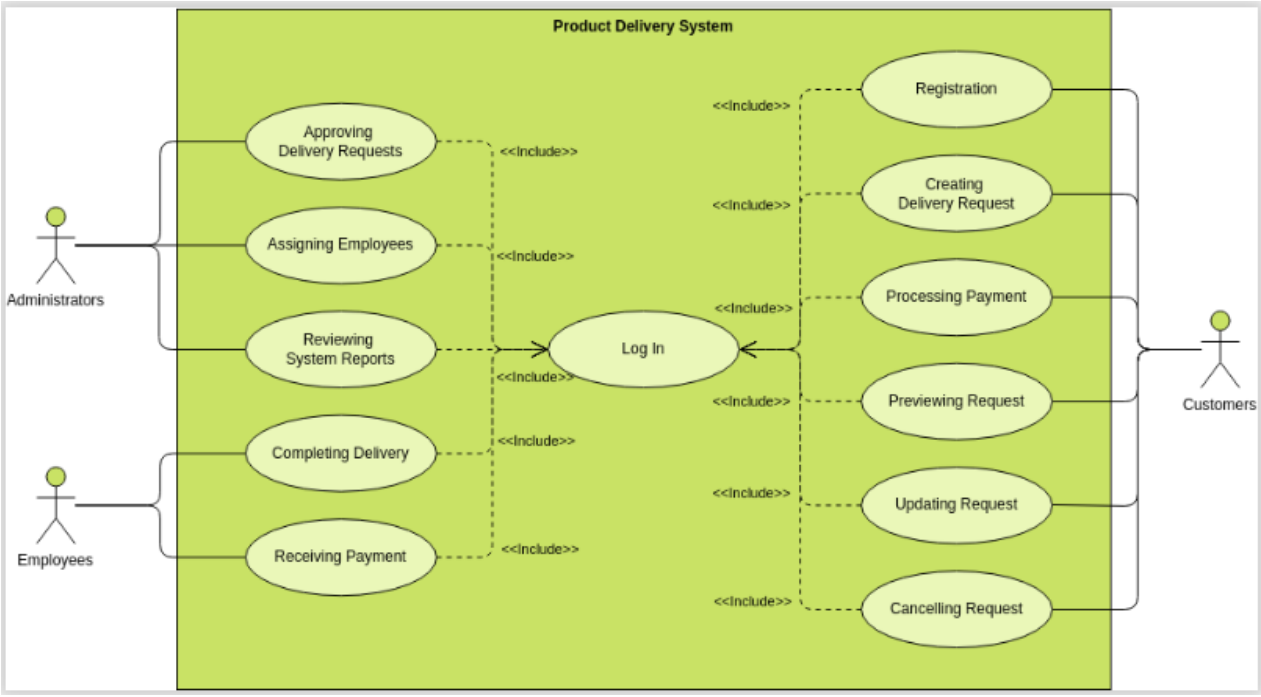
❖ **Phân tích xác định yêu cầu**

Nhiệm vụ của giai đoạn này là phải phân tích, xác định được các yêu cầu làm nổi bật được các hành vi, chức năng cần thực hiện của hệ thống dự kiến; mô tả chi tiết các yêu cầu của bài toán cần giải quyết là gì thông qua việc trả lời các câu hỏi:

- *Quy trình (bao gồm những công việc sẽ tương ứng với các chức năng hệ thống hỗ trợ) hệ thống hỗ trợ là gì?*
- *Đầu vào của mỗi chức năng hệ thống là gì? Và đầu ra tương ứng? Các ràng buộc trong xử lý đầu vào cho ra đầu ra là gì?*
- *Các yêu cầu phi chức năng?*

Đầu vào của giai đoạn này là kết quả khảo sát, thu thập được từ Stakeholder ở pha trước & đầu ra của giai đoạn chính là tập hợp các yêu cầu/bài toán đã phân tích xác định được. Việc phân tích yêu cầu có thể liên quan với việc phải tạo dựng một hay nhiều mô hình hệ thống khác nhau để phân tích viên có sự hiểu biết rõ ràng hơn về hệ thống dự kiến sẽ xây dựng.

Ví dụ: Hình 3.5 mô tả các chức năng của hệ thống phân phối sản phẩm thông qua mô hình use case. Qua mô hình này giúp chúng ta xác định được các chức năng/use case cụ thể cần triển khai của hệ thống là gì? Mối quan hệ giữa các chức năng? Ai được cấp quyền truy cập đến chức năng tương ứng? hệ thống tương tác với ứng dụng nào có sẵn trên môi trường vận hành, …

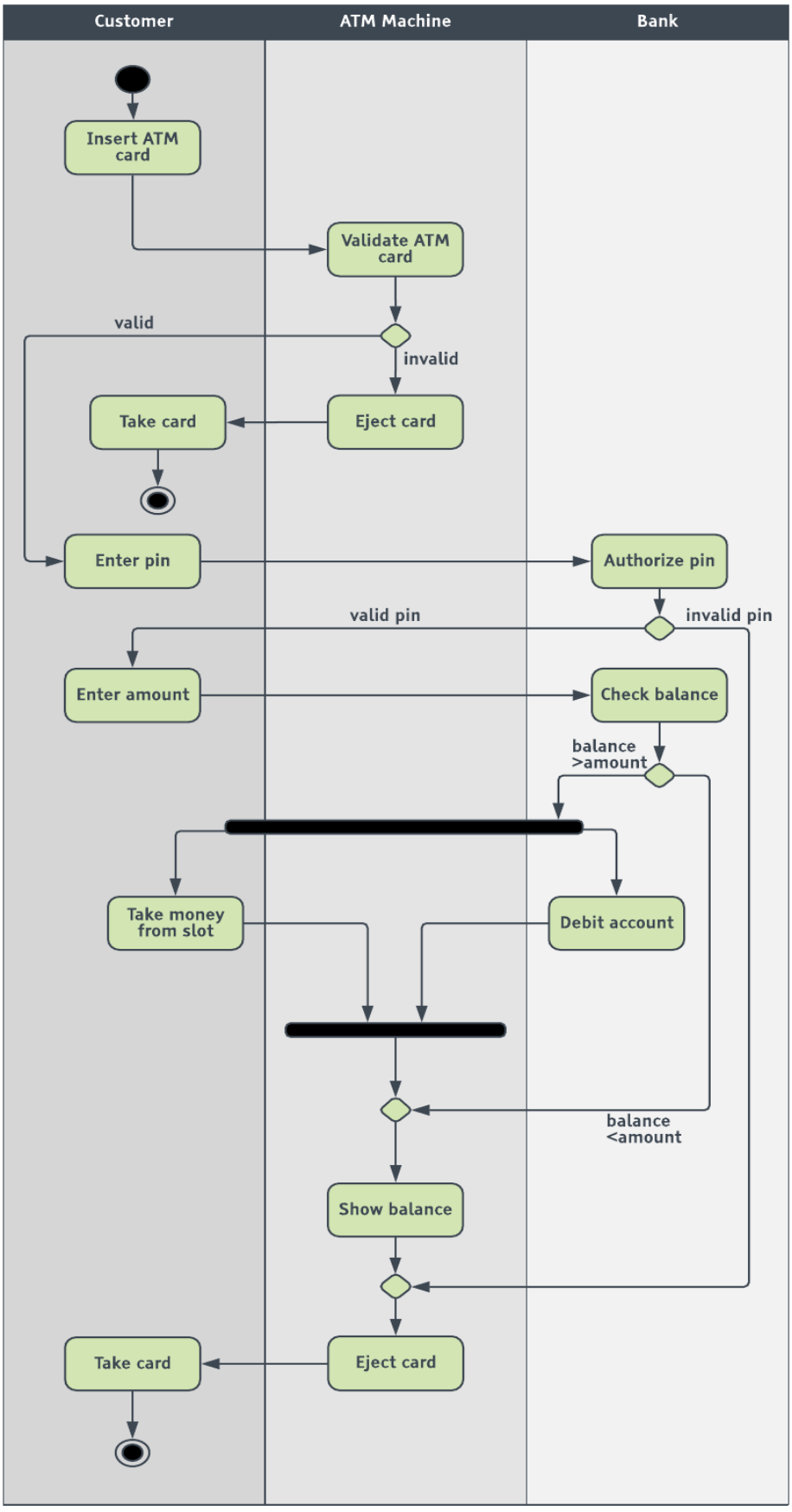


Hình 3.5: Mô hình UC của hệ thống phân phối sản phẩm

3.3.3 Đặc tả yêu cầu

Xây dựng tài liệu đặc tả yêu cầu đóng vai trò quan trọng vì: *"Không ai biết chắc chắn mình phải làm gì khi chưa có tài liệu đặc tả!"*.

Tài liệu đặc tả yêu cầu cần mô tả những yêu cầu chính thức về những gì cần phải thực hiện bởi đội dự án. Nó nên chứa tất cả các định nghĩa về yêu cầu ở hai mức: yêu cầu người dùng (thường phát biểu bằng ngôn ngữ tự nhiên, ngôn ngữ tự nhiên có cấu trúc – đặc tả phi hình thức) & các yêu cầu hệ thống (đặc tả bán hình thức, sử dụng các mô hình, mẫu biểu, các biểu diễn có cấu trúc, …). Hình 3.6 là ví dụ về biểu đồ hoạt động đặc tả chức năng rút tiền quả thẻ từ ATM:



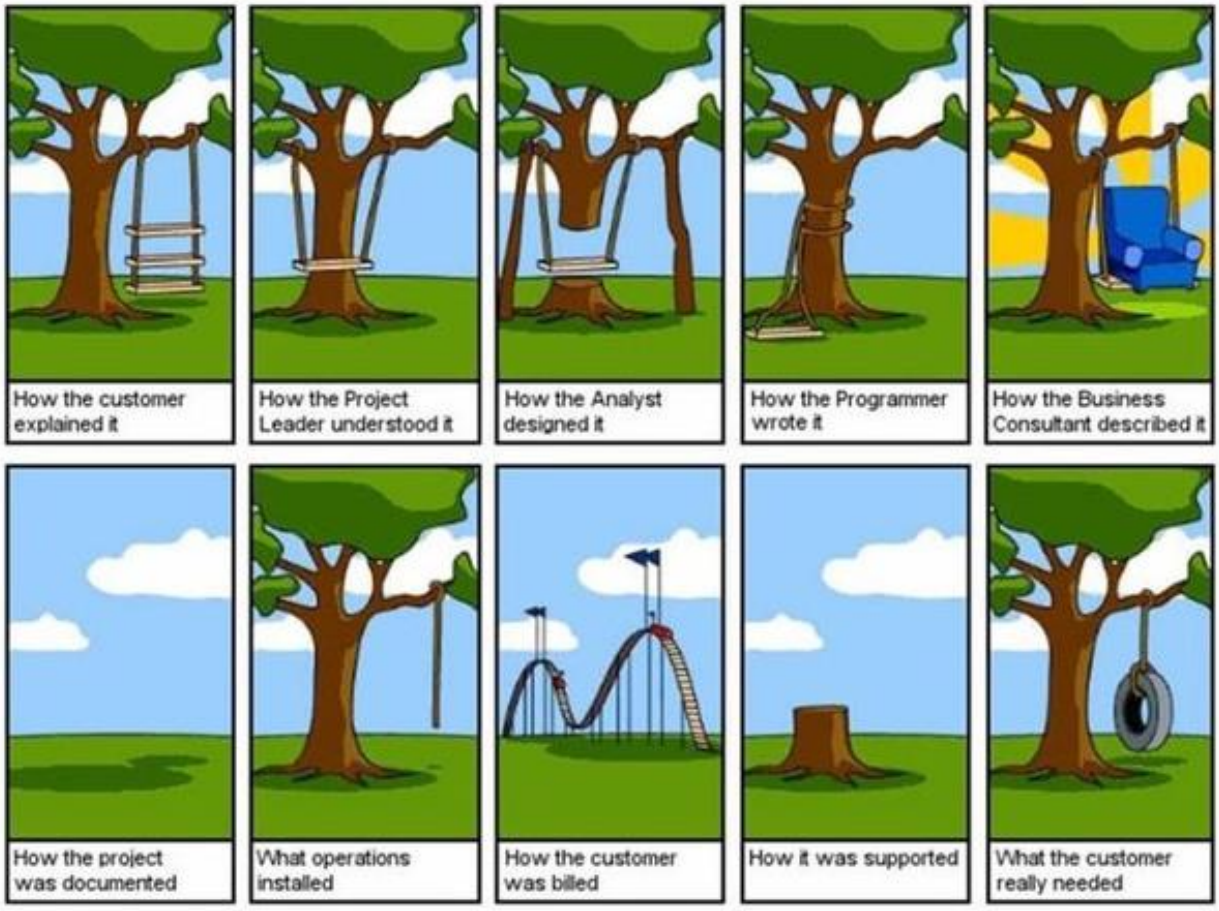
Hình 3.5: Biểu đồ hoạt động: UC rút tiền qua thẻ từ ATM

Về nguyên tắc, các yêu cầu được mô tả trong tài liệu phải rõ ràng, cụ thể, đầy đủ, chính xác và thống nhất để làm cơ sở phê chuẩn dự án, nghiệm thu, và đánh giá phần mềm trước khi chuyển giao đến người dùng. Việc mô tả sơ sài, mơ hồ các yêu cầu phần mềm có thể dẫn đến các hiểu sai, hiểu nhầm giữa những tổ chức, cá nhân liên quan, đặc biệt là đội phát triển dự án và đối tác khách hàng là hết sức nguy hiểm. Việc sửa chữa những lỗi nhầm này thực tế cho thấy phải mất rất nhiều công sức, tiền bạc và thời gian.

Hình 3.6 là một bức tranh quen thuộc trong SE, nhấn mạnh về sự hiểu sai, hiểu nhầm giữa các bên liên quan về cùng một hệ thống:

(1) Hệ thống được giải thích/mô tả bởi khách hàng (customer)
(2) Hệ thống được hiểu dưới góc độ của người lãnh đạo dự án (leader)
(3) Hệ thống được hiểu và mô tả dưới góc nhìn của phân tích viên (analyser)
(4) Hệ thống được hiểu và xây dựng bởi lập trình viên (programmer)
(5) Hệ thống dưới góc nhìn của nhà tư vấn nghiệp vụ (business consultant)
(6) Hệ thống được tư liệu hoá/đặc tả (~ không hình dung ra được)
(7) Hệ thống được vận hành thực sự ra sao (không vận hành được, không đáp ứng mục tiêu, giải quyết các khó khăn, thách thức của khách hàng)
(8) Kinh phí mà khách hàng đã chi cho việc triển khai dự án (lớn gấp nhiều lần so với giá trị mang lại)
(9) Hệ thống hỗ trợ cho khách hàng thực tế ra sao
(10) Hệ thống mà khách hàng thực sự mong muốn thực chất là gì.

Bức trang này là một sự cảnh báo mà tổ chức phát triển phần mềm cần hết sức lưu tâm để tránh các sai lầm đã được đúc kết do vấn đề giao tiếp, truyền thông không hiệu quả, đặc biệt là ở giai đoạn thu thập, phân tích và xác định yêu cầu từ các stakeholder/khách hàng.



Hình 3.6: Các vấn đề truyền thông không hiệu quả trong SE

(nguồn: htttp://www.furrygoat.com/)

Tài liệu đặc tả yêu cầu được xem như một phần của bản hợp đồng giữa tổ chức phát triển phần mềm và khách hàng. Do đó, nó cần đảm bảo chất lượng và tuân theo định dạng nhất định. Ví dụ: định dạng tài liệu SRS theo chuẩn IEEE 830 – 1993 [14]

1. Giới thiệu
- *1.1. Mục đích của tài liệu*
- *1.2. Phạm vi của sản phẩm*
- *1.3. Các định nghĩa, từ viết tắt*
- *1.4. Tài liệu tham khảo*
- *1.5. Tổng quan về tài liệu yêu cầu (mô tả cấu trúc tài liệu)*

2. Mô tả chung
- *2.1. Tổng quan về sản phẩm*
- *2.2. Các chức năng của sản phẩm*
- *2.3. Đối tương người dùng*
- *2.4. Các ràng buộc tổng thể*
- *2.5. Giả thiết và các phụ thuộc*

3. Đặc tả yêu cầu (yêu cầu chi tiết)
- *3.1 Yêu cầu chức năng*
  - *3.1.1 Yêu cầu chức năng 1*
    - *3.1.1.1 Giới thiệu*
    - *3.1.1.2 Dữ liệu vào*
    - *3.1.1.3 xử lý*
    - *3.1.1.4 Kết quả*
  - *3.1.2 Yêu cầu chức năng 2*
  - *...*
  - *3.1.n Yêu cầu chức năng n*
- *3.2 Các yêu cầu phi chức năng*
  - *3.2.1 Các thuộc tính của hệ thống*
  - *3.2.2 Các ràng buộc của hệ thống*
  - *3.2.3 Các yêu cầu khác.*

4. Phụ lục

5. Chỉ mục.

Các đặc tính quy định một tài liệu SRS tốt theo chuẩn IEEE 830-1993 gồm:

- ✓ Đúng đắn
- ✓ Không mập mờ
- ✓ Đầy đủ
- ✓ Thống nhất
- ✓ Phân hạng các yêu cầu quan trọng/ổn định
- ✓ Có khả năng thẩm định
- ✓ Có khả năng sửa đổi
- ✓ Có khả năng theo dõi.

3.3.4 Thẩm định, đánh giá yêu cầu

Các yêu cầu phần mềm có vai trò quan trọng trong việc định hình hệ thống ở giai đoạn đầu tiên trong tiến trình dự án. Lỗi liên quan đến yêu cầu có thể lây lan đến các giai đoạn tiếp theo của dự án và có thể gây ra những hậu quả nghiêm trọng nếu không được phát hiện sớm và xử lý kịp thời.

Do đó, thẩm định/đánh giá yêu cầu là không thể thiết trong quy trình RE. Trong quá trình đánh giá yêu cầu, chúng ta phải kiểm tra các yêu cầu ở những khía cạnh sau:

- ❖ **Các tiêu chí thẩm định yêu cầu riêng lẻ:**
  - Không mập mờ (Unambigious)
  - Có thể kiểm thử (Testable, verifiable)
  - Rõ ràng (clear: ngắn gọn - concise, súc tích-terse, đơn giản-simple, chính xác-precise)
  - Đúng đắn (correct)
  - Có thể hiểu (understandable)
  - Khả thi (Feasible: hiện thực-realistic, có thể thực hiện-possible)
  - Độc lập (Independent)
  - Nguyên tử (atomic)
  - Cần thiết (necessary)
  - Độc lập với cài đặt (Inplementation-free: Trừu tượng:abstract)
- ❖ **Các tiêu chí thẩm định tập yêu cầu:**
  - Thống nhất/Nhất quán (Consistent)
  - Không dư thừa (Nonredundant)
  - Đầy đủ (Complete)

Câu hỏi

Câu 1. Thuật toán nghiên cứu tính khả thi của dự án CNTT gồm mấy giai đoạn?

a) 3
b) 4
c) 5
d) 6

Câu 2. Do thường gặp khó khăn trong việc phát hiện và phân tích yêu cầu từ Stakeholder, nên sẽ sử dụng mô hình nào để phát hiện và phân tích yêu cầu?

a) Mô hình làm bản mẫu
b) Mô hình xoắn ốc
c) Mô hình thác nước
d) Mô hình RAD

Câu 3. Kết quả của việc nghiên cứu tính khả thi dự án là?

a) Là một báo cáo để quyết định có nên xây dựng hệ thống đề xuất hay không
b) Là một báo cáo đưa ra thuật toán giải quyết vấn đề
c) Là một báo cáo gồm 4 giai đoạn
d) Là một báo cáo để lựa chọn nhân sự

Câu 4. Quy trình tái kỹ nghệ bao gồm các hoạt động?

a) Dịch mã nguồn, Kỹ nghệ ngược
b) Cải thiện cấu trúc chương trình
c) Mô đun hóa chương trình, tái kỹ nghệ dữ liệu
d) Cả a,b,c

Câu 5. Hoạt động quản lý các yêu cầu phầm mềm nhằm…

a) Lưu trữ, cập nhật, lưu vết và tìm kiếm các yêu cầu dễ dàng hơn
b) Phân tích và xác định các yêu cầu
c) Quản lý sự thay đổi của các yêu cầu phần mềm
d) Trả lời câu hỏi What

Bài 6: Quản lý yêu cầu (Số tiết: 03 tiết)

3.4 Quản lý yêu cầu

Trong một báo cáo phân tích về các thất bại dự án phần mềm (2005) đã xác định ~ 71% dự án phần mềm thất bại, nguyên nhân chính do phân tích và quản lý các yêu cầu một cách nghèo nàn.

Các vấn đề trong quản lý yêu cầu thường được coi là nguyên nhân chính dẫn đến thất bại của dự án. Vì nếu chúng không được xác định đầy đủ có thể dẫn đến sai lệch phạm vi, trì hoãn dự án, vượt chi phí; chất lượng sản phẩm kém không đáp ứng nhu cầu khách hàng.

Quản lý yêu cầu nhằm quản lý các thông tin/thuộc tính của yêu cầu (nguồn gốc, trạng thái, độ khó, độ ổn định, ...); đảm bảo sản phẩm đáp ứng hoàn toàn các mục tiêu đã xác định; theo dõi các thay đổi về yêu cầu; và thúc đẩy các giao tiếp giữa các bên liên quan trong suốt vòng đời phần mềm từ khi bắt đầu dự án.

Các thuộc tính cần quản lý cho mỗi loại yêu cầu chi tiết xem tại Bảng 3.2

- ***Các yêu cầu người dùng:***
  - STRQ (Stakeholder Requests/Needs – Các nhu cầu của đối tác):
  - FEAT (Features – Các tính năng sản phẩm):
- ***Các yêu cầu hệ thống***:
  - UC (Use Cases – Các trường hợp sử dụng:
  - SUPL (Supplementary Requirements – Các yêu cầu bổ sung/phi chức năng)

Bảng 3.2 Các thuộc tính yêu cầu

Thuộc tính/ Attribute	**Giá trị/ Value**	**FEAT**	**SUPL**	**UC**	**STRQ**
Độ ưu tiên/ Priority	High (H) Medium (M) Low (L)	x	x	x	
Kiểu/Type	Functional Usability Reliability Performance Supportability	x			

	Design Constraint Implementation Physical Interface				
Trạng thái/ Status	Proposed Approved Incorporated Validated	x	x	x	
Độ khó/ Difficulty	H (M/L)	x	x	x	
Độ ổn định/ Stability	H (M/L)	x	x	x	
Rủi ro/ Risk	H/M/L	x	x	x	
Nguồn gốc/ Origin	Text	x			x
Hợp đồng/ Contact Name	Text	x	x	x	
Vấn đề/ Defect	Text	x	x	x	
Độ ưu tiên/ Stakeholder Priority	H (M/L)				x

Bản kế hoạch quản lý yêu cầu (RMP) có vai trò nhất định đối với sự thành công dự án SE vì nó cho phép đội phát triển kiểm soát được phạm vi của dự, từ đó lập kế hoạch triển khai công việc. Bản kế hoạch quản lý yêu cầu cần chứa các mô tả về cách thức thu thập, tổ chức, tư liệu hóa và quản lý các yêu cầu phần mềm; thiết lập và duy trì sự thỏa thuận giữa khách hàng và nhóm dự án về mọi thay đổi liên quan đến yêu cầu.

RMP có thể được xây dựng sử dụng mẫu biểu (IBM) tại [15]

Ví dụ về RMP của dự án xây dựng hệ thống đăng ký khoá học online (CRS) của ĐH Wylie (USA), xem tại [16].

Một số công cụ hỗ trợ quản lý yêu cầu gồm:

- Rational RequisitePro của IBM

- EnterpriseArchitecture (version 13; 14)
- JIRA (is available for Cloud)
- Bảng tính Google: Hỗ trợ nhiều tính toán, lọc thông tin, cộng tác chia sẻ dữ dữ liệu.

3.5 Case study: Kỹ nghệ yêu cầu phần mềm

a) Mục tiêu

- Mô tả tổng quan về hệ thống định xây dựng

- Xác định chi tiết các yêu cầu phần mềm

b) Yêu cầu

1. Mô tả tổng quan về hệ thống định xây dựng

2. Xác định các yêu cầu chức năng chính cho bài toán và xây dựng được biểu đồ Use case

3. Xác định các yêu cầu phi chức năng

4. Xây dựng Entity relationship diagram

c) Hướng dẫn, gợi ý: [33 – Chapter 3]

Câu hỏi

Câu 1. Lựa chọn nào sau đây mô tả một yêu cầu phi chức năng?

a) Hệ thống phải phát sinh ra một báo cáo về tất cả các chiến dịch quảng cáo cho một khách hàng cụ thể
b) Hệ thống phải cho phép những người sử dụng nhập vào chi tiết các khách hàng
c) Hệ thống phải có khả năng lưu trữ ban đầu là 500MB dữ liệu, mỗi năm tăng lên 100MB
d) Tất cả các phương án trên đều đúng

Câu 2. Lựa chọn nào sau đây mô tả một yêu cầu chức năng?

a) Hệ thống phải có khả năng trả lời tất cả các truy vấn trong 5 giây
b) Hệ thống cho phép người sử dụng thêm một sản phẩm muốn mua vào giỏ hàng
c) Người sử dụng sẽ gây ra ít lỗi hơn 50% so với hệ thống hiện tại
d) Giao diện hệ thống thân thiện với người dùng

Câu 3. Các tiêu chí cơ bản nào sau đây được sử dụng để đánh giá yêu cầu phần mềm?

a) Khả năng hiểu, hoàn thiện, đầy đủ, không mâu thuẫn
b) Hợp lệ, nhất quán, hoàn thiện, hiện thực, xác thực
c) Hợp lệ, hợp lý, phù hợp với đặc điểm người dùng, không dư thừa

d) Hợp lệ, nhất quán, đầy đủ, hướng người dùng

Câu 4. Những vấn đề nào thường gặp phải khi đặc tả yêu cầu phần mềm người dùng bằng ngôn ngữ tự nhiên?

a) Rõ ràng, thiếu tính mô đun hóa
b) Quá linh hoạt, thiếu trong sáng
c) Chính xác, logic
d) Không rõ ràng, lộn xộn giữa các yêu cầu, pha trộn giữa các yêu cầu

Câu 5. Tại sao phải quản lý sự thay đổi của các yêu cầu?

a) Vì các yêu cầu của hệ thống luôn thay đổi
b) Vì giúp ta phân tích, dự đoán các thay đổi của các yêu cầu, dự đoán chi phí cài đặt thay đổi và góp phần tạo nên thành công của sản phẩm phần mềm
c) Vì giúp ta tạo ra các sản phẩm phần mềm, dễ bảo trì, mở rộng, ít lỗi
d) Giúp ta giảm chi phí trong quá trình kiểm thử, bảo trì, mở rộng phần mềm

Bài tập cuối chương

Bài 3.1. Liệt kê 5 đặc trưng của một yêu cầu tốt?

Bài 3.2. Cho trước danh sách các yêu cầu của một ứng dụng như sau.

Yêu cầu: phân loại các yêu cầu này thành các loại như chỉ ra trong ngoặc (B = Bussiness, U = User, F = Functional, N = Nonfunctional, và I = Implementation):

a) Cho phép người dùng giám sát việc download/upload từ xa
b) Cho phép người dùng đặc tả các tham số log-in vào ứng dụng, ví dụ: địa chỉ internet, cổng, tên người dùng và mật khẩu
c) Cho phép người dùng đặc tả các tham số download/Upload như số lượng đầu vào nếu có vấn đề tồn tại
d) Cho phép người dùng lựa chọn vị trí mạng, 1 file cục bộ và thời gian thực hiện việc upload/download
e) Cho phép người dùng lập lịch upload/download tại thời điểm bất kỳ
f) Cho phép download/upload để chạy tại thời điểm bất kỳ
g) Tạo bộ chuyển đổi upload/download ít nhất 8 Mbps
h) Chạy các upload/download một cách tuần tự. 2 download/upload không thể thực hiện đồng thời tại cùng một thời điểm
i) Nếu một download/upload được lập lịch thực hiện trong khi một upload/download khác đang diễn ra thì nhiệm vụ mới sẽ phải chờ cho đến khi nhiệm vụ download/upload cũ hoàn thành
j) Thực hiện các download/upload như đã lập lịch

k) Lưu giữ lịch sử của tất cả các upload/download đã cố gắng thực hiện và kiểm tra xem liệu chúng đã thành công chưa
l) Cho phép người dùng xoá lịch sử download/upload
m) Hiện thị các báo cáo về các cố gắng upload/download
n) Cho phép người dùng xem các báo cáo về lịch sử trên một thiết bị từ xa, ví dụ: điện thoại di động
o) Gửi email đến quản trị viên nếu việc download/upload bị thất bại sau khi đã cố gắng thực hiện vượt quá ngưỡng tối đa số lần cho trước
p) Gửi thông báo đến quản trị viên nếu việc download/upload thất bại sau khi đã cố gắng thực hiện vượt quá ngưỡng tối đa số lần cho trước

Bài 3.3. Cho trước danh sách các yêu cầu như sau.

Yêu cầu: Phân loại các yêu cầu này thành các loại FURPS (F = Functionality, U = Usability, R = Reliability; P= Performance; and S = Supportability).

a) Cho phép người dùng giám sát việc download/upload từ
b) Cho phép người dùng đặc tả các tham số log-in vào ứng dụng, ví dụ: địa chỉ internet, cổng, tên người dùng và mật khẩu
c) Cho phép người dùng đặc tả các tham số download/Upload như số lượng đầu vào nếu có vấn đề tồn
d) Cho phép người dùng lựa chọn vị trí mạng, 1 file cục bộ và thời gian thực hiện việc upload/download
e) Cho phép người dùng lập lịch upload/download tại thời điểm bất kỳ
f) Cho phép download/upload để chạy tại thời điểm bất kỳ
g) Tạo bộ chuyển đổi upload/download ít nhất 8 Mbps
h) Chạy các upload/download một cách tuần tự. 2 download/upload không thể thực hiện đồng thời tại cùng một thời điểm
i) Nếu một download/upload được lập lịch thực hiện trong khi một upload/download khác đang diễn ra thì nhiệm vụ mới sẽ phải chờ cho đến khi nhiệm vụ download/upload cũ hoàn thành
j) Thực hiện các download/upload như đã lập lịch
k) Lưu giữ lịch sử của tất cả các upload/download đã cố gắng thực hiện và kiểm tra xem liệu chúng đã thành công chưa
l) Cho phép người dùng xoá lịch sử download/upload
m) Hiện thị các báo cáo về các cố gắng upload/download
n) Cho phép người dùng xem các báo cáo về lịch sử trên một thiết bị từ xa, ví dụ: điện thoại di động
o) Gửi email đến quản trị viên nếu việc download/upload bị thất bại sau khi đã cố gắng thực hiện vượt quá ngưỡng tối đa số lần cho trước

p) Gửi thông báo đến quản trị viên nếu việc download/upload thất bại sau khi đã cố gắng thực hiện vượt quá ngưỡng tối đa số lần cho trước

Bài 3.4. Viết một mô tả ngắn gọn về phân tích miền cho một hệ thống

Bài 3.5. Thực hành kỹ thuật phỏng vấn để thu thập các yêu cầu của khách hàng cho một dự án cụ thể

Bài 3.6. Cho trước bài toán xây dựng website bán hàng online. Yêu cầu: xác định các tác nhân cho của hệ thống & danh sách các UC.

Bài 3.7. Chúng ta thấy rằng các hệ thống phần mềm lớn, phức tạp thường được phát triển bởi rất nhiều cá nhân, rất ít người có được bức tranh toàn cảnh về toàn bộ dự án. Vậy, đối với một người làm công, tham gia vào một dự án mà không biết về toàn bộ chức năng của dự án đó thì có hợp lý không? Vì sao?

Bài 3.8. Trình bày ngắn gọn các lợi ích chính của tài liệu đặc tả yêu cầu?

Bài 3.9. Hãy tìm hiểu và mô tả các vấn đề gây ra các khó khăn trong thu thập yêu cầu?

Bài 3.10. Cho trước phát biểu bài toán, xác định các mục tiêu của hệ thống

Bài 3.11. Cho trước các yêu cầu chức năng của một hệ thống, Yêu cầu ánh xạ thành các UC tương ứng + xây dựng ma trận dấu vết

Bài 3.12. Cho trước mô hình UC của một hệ thống. Yêu cầu xây dựng tài liệu đặc tả cho từng UC

Bài 3.13. Cho trước tập yêu cầu của một hệ thống, hãy phân loại chúng thành các yêu cầu chức năng hoặc phi chức năng

Bài 3.14. Cho trước tập yêu cầu của một hệ thống, hãy viết lại các yêu cầu sao cho chúng có khả năng thẩm định

(Tham khảo tại: https://www.site.uottawa.ca/school/research/lloseng/exerciselist.html)

Chương IV: THIẾT KẾ PHẦN MỀM

Nội dung chính của chương

4.1 Tổng quan về thiết kế phần mềm
4.2 Quy trình thiết kế phần mềm

Mục tiêu cần đạt được của chương

- *Nắm vững qui trình thiết kế hệ thống.*
- *Biết áp dụng để phân tích thiết kế một bài toán*

Bài 7: Thiết kế phần mềm (Số tiết: 03 tiết)

4.1 Tổng quan về thiết kế phần mềm

4.1.1 Thiết kế phần mềm là gì?

Thiết kế phần mềm là hoạt động chuyển đặc tả yêu cầu thành mô hình thiết kế mà người lập trình có thể chuyển thành chương trình với một ngôn ngữ lập trình cụ thể, chương trình có thể vận hành được, đáp ứng được yêu cầu đặt ra.

Thiết kế là một quá trình sáng tạo để tìm giải pháp công nghệ (cách thức, phương án), biểu diễn cách thức, phương án đó, xét duyệt lại và chi tiết hóa dần dần. Bản thiết kế phải đủ chi tiết để người lập trình biết phải làm như thế nào để chuyển thành chương trình [4].

4.1.2 Vai trò của thiết kế

Thiết kế hệ thống là một hoạt động nhằm trả lời câu hỏi "làm thế nào để triển khai được hệ thống thỏa mãn các yêu cầu phần mềm". Nó tạo ra mô hình cài đặt của phần mềm và là công cụ giao tiếp giữa những người tham gia phát triển hệ thống, là cơ sở để đảm bảo chất lượng hệ thống:

- Bản thiết kế dễ đọc, dễ hiểu, dễ sửa đổi hơn mã chương trình
- Có nhiều mức chi tiết, cung cấp cái nhìn tổng thể
- Làm cơ sở để trao đổi, cải tiến

Bản thiết kế cũng cung cấp đầy đủ thông tin cho việc bảo trì sau này:

- Giảm công sức mã hóa khi sửa đổi
- Tiện bảo trì, phát triển, mở rộng.

Thiết kế hệ thống đóng vai trò rất quan trọng, đặc biệt với những hệ thống phần mềm lớn, phức tạp, thời gian sống lâu.

4.1.3 Nguyên lý thiết kế

1. Thiết kế không bị bó buộc vào một cách nhìn hạn chế nào. Nó cần được lựa chọn

từ các giải pháp có thể.

2. Thiết kế phải cho phép lần ngược lại mô hình phân tích. Các mô đun và các yêu cầu không nhất thiết phải tương ứng 1-1, nhưng phải kiểm tra được sự thỏa mãn các yêu cầu.

3. Không nên tạo lại các thiết kế (giải pháp) đã có, mà cần tái sử dụng tối đa chúng.

4. Mô hình thiết kế (giải pháp) nên tiến gần đến mô hình thế giới thực (bài toán).

5. Biểu diễn thiết kế phải nhất quán và có tính tích hợp. Thiết kế do nhiều người tiến hành, do đó phải thống nhất cách biểu diễn, thống nhất giao diện.

6. Thiết kế cần có cấu trúc để hiểu, dễ thay đổi tức là phải được mô đun hóa và phân cấp.

7. Thiết kế không phải là mã hóa. Kết quả thiết kế luôn có mức trừu tượng hơn mã hóa, phải đảm bảo dễ hiểu, dễ thay đổi.

8. Thiết kế cần được đánh giá chất lượng ngay trong khi được tạo ra. Chất lượng thiết kế được đánh giá qua tính kết dính, tính ghép nối, hiệu quả thuật toán

9. Thiết kế cần được thẩm định để tránh các lỗi mang tính hệ thống. Các lỗi có thể là thiếu chức năng, chức năng không rõ, mâu thuẫn, ...

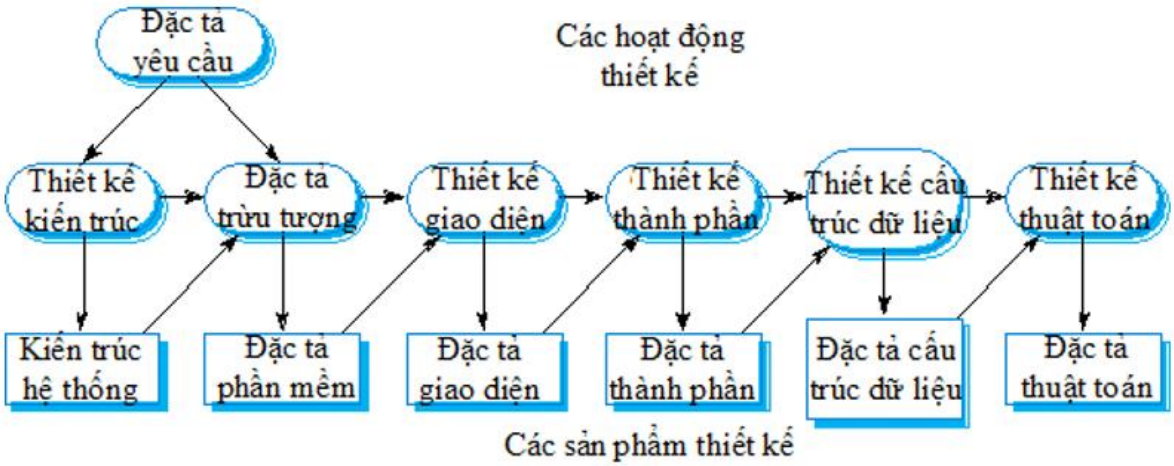
4.1.4 Tiêu chí chất lượng thiết kế

Cần thiết lập các tiêu chí kỹ thuật để đánh giá một bản thiết kế tốt hay không?

- ✓ Thiết kế cần có kiến trúc tốt. Bản thiết kế phải được cáu thành từ các mẫu (pattern), các thành phần có đặc trưng tốt, dễ tiến hóa.
- ✓ Thiết kế được mô đun hóa cho mỗi thành phần chức năng
- ✓ Chứa các biểu diễn tách biệt nhau về dữ liệu, kiến trúc, giao diện, thành phần và môi trường.
- ✓ Liên kết qua giao diện giúp làm giảm độ phức tạp liên kết giữa các mô đun với nhau và giữa hệ thống với môi trường.

4.2 Quy trình thiết kế phần mềm [1]

Bản thiết kế phần mềm là một tài liệu mô tả về cấu trúc của phần mềm được cài đặt. Mô hình chung của quy trình thiết kế như sau:



Hình 4.1: Mô hình chung của quy trình thiết kế

Hình 4.1 là một tiến trình thiết kế trong đó các hoạt động được tiến hành tuần tự. Trong thực tế các hoạt động này là đan xen nhau. Các phản hồi từ giai đoạn này đến giai đoạn khác dẫn đến phải làm lại thiết kế là không thể tránh khỏi trong mọi tiến trình thiết kế .

Đầu vào của quy trình thiết kế là tài liệu đặc tả yêu cầu phần mềm. Đầu ra của quy trình thiết kế là các bản đặc tả thiết kế. Mỗi tài liệu này là đầu ra của từng giai đoạn thiết kế tương ứng. Vì thiết kế là liên tục, các bản đặc tả này trở nên chi tiết dần, hình thức dần. Kết quả cuối cùng của quy trình thiết kế là bản đặc tả chính xác về các giải thuật và các cấu trúc dữ liệu được đặc tả.

4.2.1 Thiết kế kiến trúc

Thiết kế kiến trúc là quá trình xác định các hệ thống con (các phân hệ) lập thành hệ thống và khung làm việc để điều khiển và giao tiếp giữa các hệ thống con với nhau (sự truyền thông giữa chúng).

Thiết kế kiến trúc hệ thống là giai đoạn sớm nhất trong quy trình thiết kế hệ thống, nó được tiến hành song song cùng với một số hoạt động đặc tả. Khi thiết kế kiến trúc, chúng ta sử dụng các biểu đồ cấu trúc (structure chart) để mô tả cái nhìn tổng thể về hệ thống, mối quan hệ giữa các mô đun và giao diện giữa các mô đun mà không cần chỉ ra thứ tự thực hiện, số lần thực hiện và chi tiết thiết kế là gì.

Đầu ra của hoạt động thiết kế kiến trúc là tài liệu mô tả về kiến trúc phần mềm. Bản thiết kế kiến trúc biểu diễn sự liên kết giữa các tiến trình thiết kế và đặc tả.

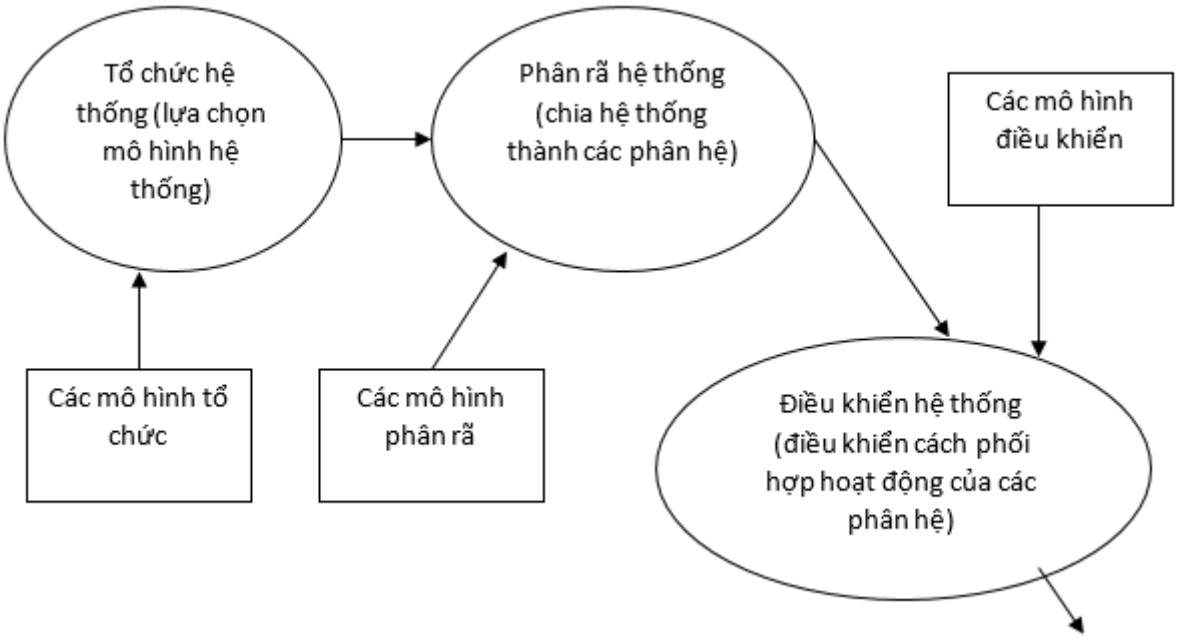
Các mô hình kiến trúc [6]:

Mô hình kiến trúc của một hệ thống cụ thể có thể tuân theo kiểu kiến trúc tổng thể. Hiểu biết về các kiểu kiến trúc tổng thể này giúp ta dễ dàng hơn trong việc định nghĩa kiến trúc hệ thống. Tuy nhiên, hầu hết các hệ thống lớn là hỗn tạp, không đồng nhất và không tuân theo một kiểu kiến trúc đơn lẻ nào đó.

Mô hình kiến trúc được sử dụng để biểu diễn hoạt động thiết kế kiến trúc. Mỗi mô hình kiến trúc chỉ ra một khía cạnh về thiết kế kiến trúc:

- ✓ Mô hình cấu trúc tĩnh: Chỉ ra các thành phần chính của hệ thống.
- ✓ Mô hình tiến trình động: Chỉ ra cấu trúc tổ chức các tiến trình của hệ thống.
- ✓ Mô hình giao diện: Định nghĩa các giao diện hệ thống con.
- ✓ Mô hình mối quan hệ: Ví dụ mô hình luồng dữ liệu, chỉ ra các mối quan hệ hệ thống con.
- ✓ Mô hình phân tán: Chỉ ra cách thức các hệ thống con được phân tán trên các máy tính như thế nào.

Các hoạt động trong quy trình thiết kế kiến trúc:



Hình 4.2: Các hoạt động trong quy trình thiết kế kiến trúc

4.2.1.1 Tổ chức hệ thống

Tổ chức hệ thống là hoạt động đầu tiên phải thực hiện quá trình thiết kế kiến trúc. Mục đích nhằm xây dựng (lựa chọn) mô hình tổ chức hệ thống. Khi thiết kế kiến trúc các mô hình thiết kế khác nhau được tạo ra. Mỗi mô hình biểu diễn một cách nhìn cụ thể của kiến trúc. Một số mô hình kiến trúc thông dụng của hệ thống:

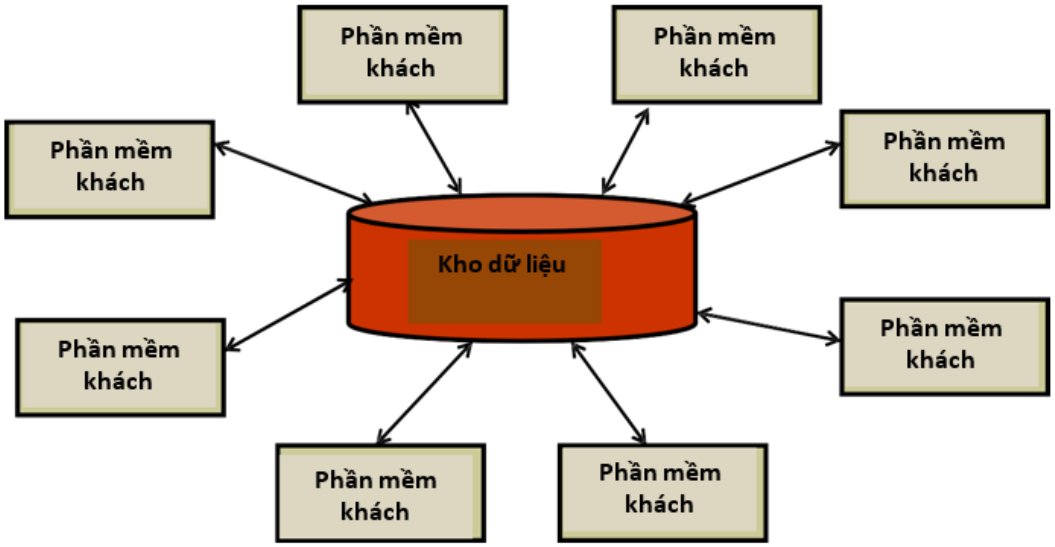
1. Kiến trúc dữ liệu dùng chung (Data centered Architectures)
2. Kiến trúc khách-dịch vụ (Client-server Architectures)
3. Kiến trúc phân tầng (Layered Architecture)
4. Kiến trúc SOA
5. Kiến trúc Mocroservices

a) Kiến trúc dữ liệu dùng chung (Data centered Architectures)

Các hệ thống con cần trao đổi thông tin với nhau. Việc trao đổi dữ liệu được thực hiện theo hai cách:

- Mỗi phân hệ duy trì một cơ sở dữ liệu riêng của mình. Dữ liệu được trao đổi giữa các phân hệ bằng cách chuyển qua các thông báo.
- Mọi dữ liệu được lưu trữ tại một cơ sở dữ liệu trung tâm, các phân hệ trong hệ thống có thể truy cập CSDL này. Mô hình này gọi là mô hình kho dữ liệu dùng chung.

=> Nếu số lượng dữ liệu dùng chung rất lớn thì mô hình kho dữ liệu dùng chung thường được sử dụng phổ biến nhất. Mô hình kiến trúc dữ liệu tập trung chỉ ra tại hình 4.3



Hình 4.3: Mô hình kiến trúc dữ liệu tập chung

Ưu điểm:

- Đây là phương pháp hiệu quả để chia sẻ số lượng lớn dữ liệu mà không cần chuyển đổi dữ liệu từ phân hệ này sang phân hệ khác.
- Các hệ thống con không cần quan tâm tới những hoạt động liên quan đến dữ liệu như: sao lưu, bảo mật, điều khiển truy cập và khôi phục … vì đã có bộ quản lý trung tâm thực hiện nhiệm vụ này.
- Phân hệ tạo dữ liệu không cần lên quan đến việc các phân hệ khác sử dụng dữ liệu như thế nào.

Nhược điểm

- Tất cả các hệ thống con phải chấp nhận mô hình kho dữ liệu, mà thực tế các phân hệ khác nhau có thể có các yêu cầu khác nhau về mức độ bảo mật, khôi phục, sao lưu... dữ liệu.

- Việc phân tán dữ liệu tới các hệ thống con sẽ gặp khó khăn.

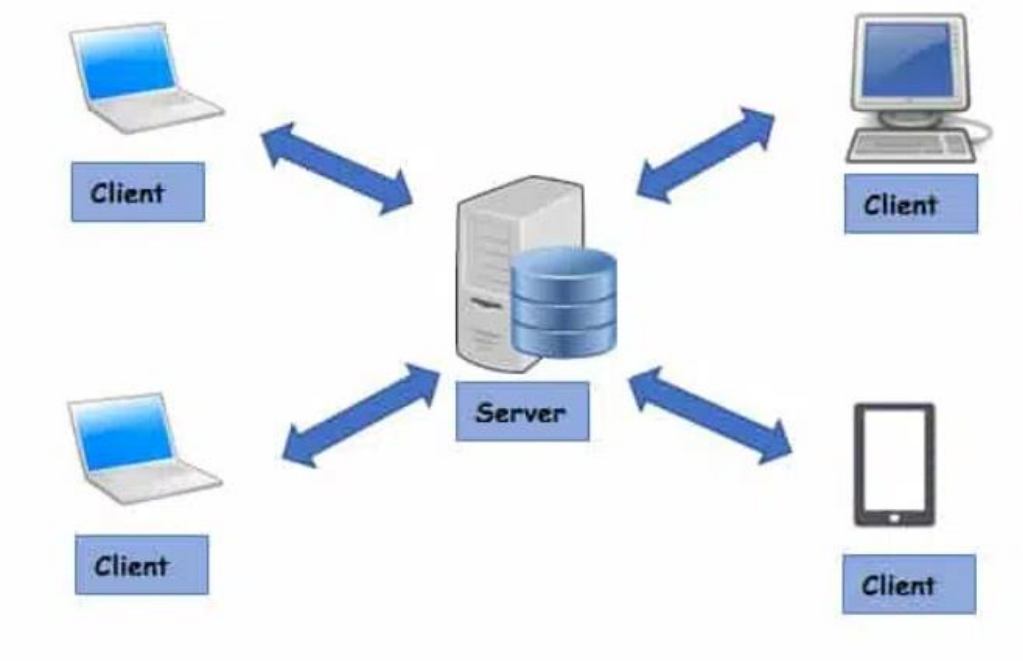
b) Kiến trúc khách-dịch vụ (Client-server Architectures)

Mô hình khách - phục vụ là một mô hình hệ thống phân tán, biểu diễn việc phân tán các dữ liệu và xử lý trên nhiều máy tính khác nhau.

Các thành phần chính của mô hình là:

- Một tập các server độc lập phục vụ cho các phân hệ bằng cách cung cấp dịch vụ như: in ấn, quản lý dữ liệu, ...
- Một tập các khách hàng (client – phân hệ) truy nhập đến server để yêu cầu cung cấp dịch vụ.
- Hệ thống mạng cho phép client truy cập tới dịch vụ mà server cung cấp.

Client phải biết tên của server và các dịch vụ mà server cung cấp. Nhưng server thì không cần xác định rõ client và hiện tại có bao nhiêu client. Client tạo ra một yêu cầu tới server và chờ server trả lời. Kiến trúc khách-dịch vụ được chỉ ra ở hình 4.4



Hình 4.4: Kiến trúc khách - dịch vụ (Client - server)

Ưu điểm:

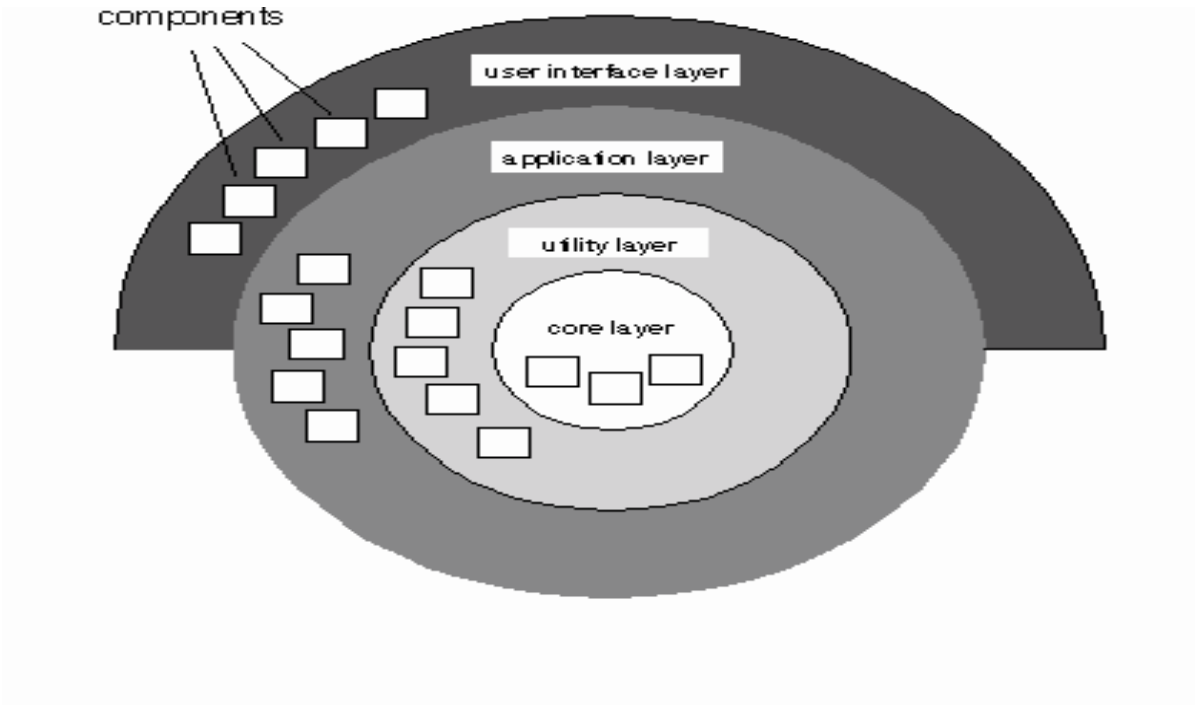
- Phân tán dữ liệu rõ ràng.
- Sử dụng các hệ thống được kết nối mạng một cách hiệu quả và chi phí dành cho phần cứng có thể rẻ hơn.
- Dễ dàng bổ sung hoặc nâng cấp server.

Nhược điểm:

- Không phải là mô hình dữ liệu dùng chung nên các hệ thống con có thể sử dụng các tổ chức dữ liệu khác nhau. Do đó, việc trao đổi dữ liệu có thể không hiệu quả vì phải chuyển đổi.
- Quản lý mỗi server không thống nhất, dư thừa (nhiều sever cung cấp cùng một dịch vụ, ...)
- Không đăng ký tên và dịch vụ tập trung. Điều này làm cho việc tìm kiếm server hoặc các dịch vụ rất khó khăn.

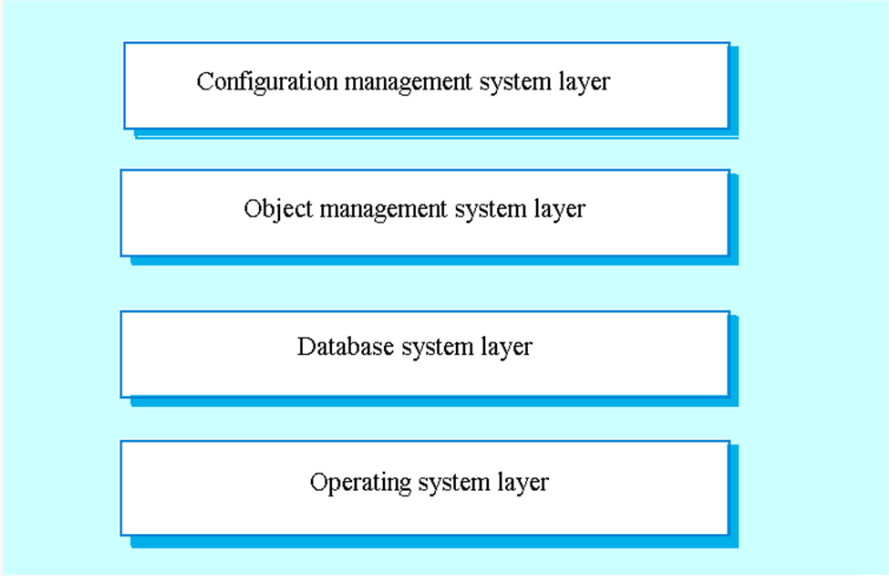
c) Kiến trúc phân tầng (Layered Architecture)

Mô hình phân lớp tổ chức hệ thống thành nhiều lớp và mỗi lớp cung cấp một tập các dịch vụ. Mỗi lớp có thể được coi như một máy trừu tượng (abstract machine) mà ngôn ngữ của máy được định nghĩa bởi các dịch vụ mà lớp đó cung cấp. Kiến trúc phân tầng được chỉ ra ở hình 4.5



Hình 4.5: Kiến trúc phân tầng

Ví dụ: Mô hình phân tầng của hệ thống quản lý phiên bản:



Hình 4.6: Mô hình phân tầng của hệ thống quản lý phiên bản

d) Kiến trúc hướng dịch vụ SOA (Service - Oriented Architecture) [18]

Khái niệm kiến trúc SOA:

Rất nhiều người với những vai trò khác nhau cố gắng đưa ra định nghĩa cho SOA. Họ có thể là những người quản lý, kiến trúc sư phần mềm, người phát triển hệ thống... Tùy theo hướng quan sát, mỗi người có thể đưa ra một định nghĩa riêng cho SOA. Thông thường các định nghĩa này không mâu thuẫn với nhau mà bổ sung cho nhau. Rõ ràng là nếu chúng ta giải thích "SOA là gì?" cho một giám đốc điều hành sẽ khác với cách chúng ta giải thích cho một lập trình viên.

Với người quản lý các hệ thống thông tin, SOA là một giải pháp với tham vọng giảm chi phí của các ứng dụng, tăng lợi nhuận khi đầu tư vào ứng dụng và tài nguyên công nghệ, giảm thời gian đưa ra giải pháp cho nghiệp vụ.

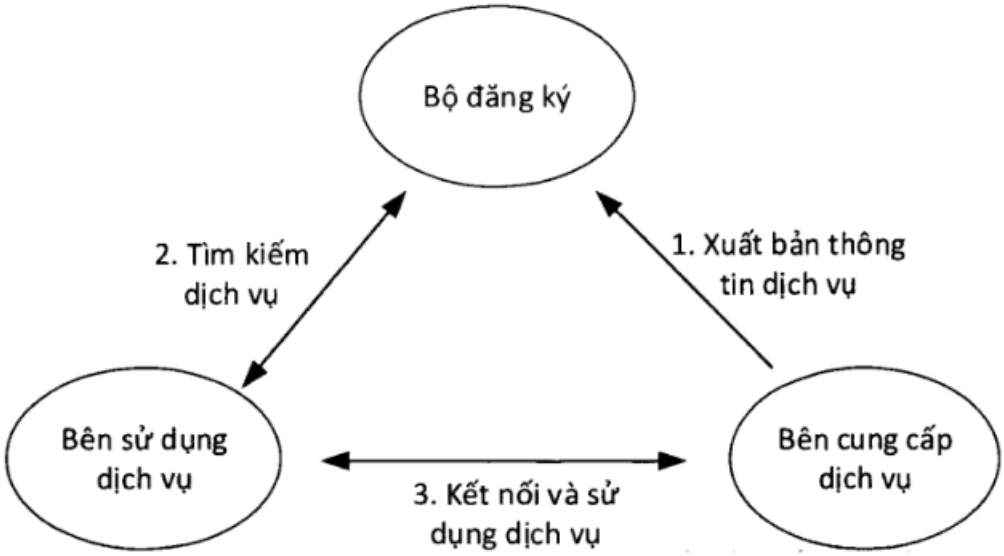
Với những người quản lý nghiệp vụ, SOA là một tập các dịch vụ có thể sử dụng cho khách hàng, đối tác, và các phần khác nhau của tổ chức. Các ứng dụng sẽ hỗ trợ tối đa cho nghiệp vụ bởi vì chúng là kết quả của việc kết hợp các dịch vụ. Các dịch vụ này có thể thay đổi một cách nhanh chóng, tái triển khai trong những ngữ cảnh nghiệp vụ khác nhau, cho phép nghiệp vụ có thể dễ dàng đáp ứng theo nhu cầu của khách hàng, cơ hội kinh doanh và điều kiện thị trường.

Với những người là kiến trúc sư phần mềm, SOA có nghĩa là tạo ra các ứng dụng dễ dàng được cấu hình và kết hợp. SOA giảm thiểu sự phức tạp và khó thay đổi của hạ tầng CNTT. SOA giảm thời gian phát triển và chi phí bởi vì việc giảm độ phức tạp sẽ dẫn đến việc thay đổi và kiểm thử dễ dàng hơn. Nếu quan tâm chi tiết hơn, kiến trúc sư phần mềm có thể cho rằng SOA là một giải pháp kiến trúc cho việc tích hợp nhiều hệ thống khác nhau và cung cấp một kiểu kiến trúc thúc đẩy kết nối mềm dẻo và tái sử dụng.

Đối với những người phát triển, SOA là một mô hình lập trình sử dụng công nghệ dịch vụ Web (Web services) với các chuẩn như SOAP, WSDL, và WS-BPEL. Việc sử dụng các chuẩn sẽ được hỗ trợ thông qua các ngôn ngữ lập trình, công cụ và môi trường phát triển.

Các thành phần chính trong SOA

Những thành phần chính trong SOA được trình bày trong hình 4.7



Hình 4.7: Các thành phần chính trong SOA

Các thành phần chính bao gồm bên cung cấp dịch vụ (Service provider), bên sử dụng dịch vụ (Service consumer), và bộ đăng ký dịch vụ (Service registry). Sau khi phát triển xong dịch vụ, bên cung cấp sẽ xuất bản các thông tin về dịch vụ ở bộ đăng ký. Khi cần tìm một dịch vụ để sử dụng, bên sử dụng sẽ kết nối và tìm kiếm ở bộ đăng ký. Sau khi có thông tin về dịch vụ, bên sử dụng kết nối và sử dụng các chức năng cung cấp bởi dịch vụ.

Dịch vụ:

Khái niệm quan trọng nhất trong kiến trúc hướng dịch vụ là dịch vụ. Trong ngữ cảnh này, một dịch vụ là một chức năng được cung cấp qua mạng. Dịch vụ có giao diện lập trình rõ ràng và được chuẩn hóa. Chúng ta có thể có những dịch vụ đơn giản ví dụ như dịch vụ tra cứu tỉ giá ngoại tệ, hoặc những dịch vụ phức tạp hơn như dịch vụ đặt vé máy bay.

Một dịch vụ trong SOA có những thuộc tỉnh chính sau:

- Không trạng thái (stateless): Khi một bên sử dụng gọi dịch vụ nhiều lần, dịch vụ không cần biết những chức năng đã được gọi, và không quan tâm đến chức năng nào sẽ

được sử dụng trong tương lai.

- Có thể được khám phá: Các dịch vụ đều có thể được tìm thấy bởi bên sử dụng.

- Tự mô tả: Dịch vụ thường đi kèm với giao diện mô tả về dịch vụ. Ở trong các chương sau chúng ta sẽ thấy rằng dịch vụ Web sẽ được mô tả bằng WSDL.

- Có thể kết hợp: Dịch vụ trong SOA phải có khả năng kết hợp với nhau để thực hiện một chức năng nào đấy. Khả năng kết hợp các dịch vụ chính là sức mạnh của SOA. Ở trong Chương 6 của giáo trình này, chúng ta sẽ tìm hiểu chi tiết hơn về việc kết hợp dịch vụ (cụ thể là kết hợp dịch vụ Web).

- Kết nối mềm dẻo (loose coupling): Phía sử dụng có thể kết nối và sử dụng dịch vụ trong thời gian chạy. Trong quá trinh hoạt động, phía sử dụng có thể thay đổi, kết nối và sử dụng những dịch vụ khác có chức năng tương đương.

- Độc lập với vị trí, ngôn ngữ, và giao thức: Dịch vụ phải được thiết kế để trong suốt với vị trí, độc lập với ngôn ngữ lập trình và giao thức vận chuyển. Trong suốt với vị trí có nghĩa là việc hiện thực dịch vụ không phụ thuộc vào địa chỉ IP hay tên miền của máy chủ. Trong quá trình triển khai dịch vụ, việc thay đổi vị trí của dịch vụ chỉ ảnh hưởng đến việc cập nhật lại thông tin về dịch vụ ở bộ đăng ký. Ở khía cạnh ngôn ngữ lập trình, việc dịch vụ được hiện thực bằng ngôn ngữ lập trình nào hoàn toàn độc lập với bên sử dụng. Một dịch vụ có thể được triển khai với nhiều giao thức vận chuyển khác nhau như HTTP, SMTP, hoặc/và FTP.

Lợi ích của SOA

Sử dụng SOA sẽ làm tăng khả năng tích hợp các ứng dụng trong cùng một công ty. Tích hợp các ứng dụng trong công ty sẽ giúp công ty sử dụng tốt hơn các nguồn thông tin. Tuy nhiên, việc tích hợp này không dễ dàng vì các ứng dụng này có thể được cung cấp bởi nhiều công ty khác nhau và sử dụng các công nghệ khác nhau. Tích hợp các ứng dụng được xây dựng trên những nền tảng khác nhau sẽ cần chi phí đáng kể, thậm chí nhiều lúc còn không thực hiện được. SOA sẽ là giải pháp cho vấn đề này.

SOA cho phép các công ty kết nối các hệ thống với nhau, thúc đẩy hợp tác kinh doanh. Trước đây, việc kết nối hệ thống thông tin của các công ty thường bị hạn chế vị trở ngại về mặt kỹ thuật. Với SOA, các chức năng của hệ thống sẽ được chuyển thành dịch vụ dựa trên các chuẩn chung. Điều này dẫn đến việc kết nối giữa các công ty sẽ được tiến hành dễ dàng.

SOA giúp cho hệ thống công nghệ thông tin đáp ứng tốt hơn các yêu cầu thay đổi từ phía nghiệp vụ. Trong các doanh nghiệp CNTT được xây dựng như là cơ sở hạ tầng để doanh nghiệp hoạt động. Vì một lý do nào đó, doanh nghiệp thay đổi về nghiệp vụ (sát nhập các bộ phận, sát nhập với doanh nghiệp khác, đưa ra dịch vụ mới, sản phẩm mới, kinh doanh trực tuyến,...). Việc này sẽ kéo theo sự thay đổi phía CNTT. Nói một cách chính xác hơn, hạ tầng CNTT cần phải thay đổi để phục vụ cho sự thay đổi phía

nghiệp vụ. Nếu không được thiết kế để chuẩn bị cho các sự thay đổi từ phía nghiệp vụ, sự thay đổi ở phía nền tảng CNTT sẽ cần nhiều thời gian, chi phí cao và có thể dẫn đến làm giảm ý nghĩa của việc thay đổi nghiệp vụ. Với SOA, các quy trình nghiệp vụ dựa trên các dịch vụ. Do đó, việc thay đổi, tạo mới các quy trình nghiệp vụ trở nên đơn giản và nhanh chóng hơn.

Trong quá trình xây dựng hệ thống CNTT dựa trên SOA, chúng ta sẽ xác định rõ dịch vụ nào phục vụ cho quy trình nghiệp vụ nào. Từ đấy, chúng ta biết được dịch vụ đẩy có hiệu qua không đối với nghiệp vụ của công ty. Bằng cách xem xét tất cả các dịch vụ, chúng ta sẽ biết được mức độ hiệu quả của việc đầu tư cho CNTT. Từ đó, các doanh nghiệp sẽ xác định được khoản đầu tư và hướng đầu tư phù hợp để phát triển cơ sở hạ tầng CNTT nhằm phục vụ hiệu quả nhất cho nghiệp vụ.

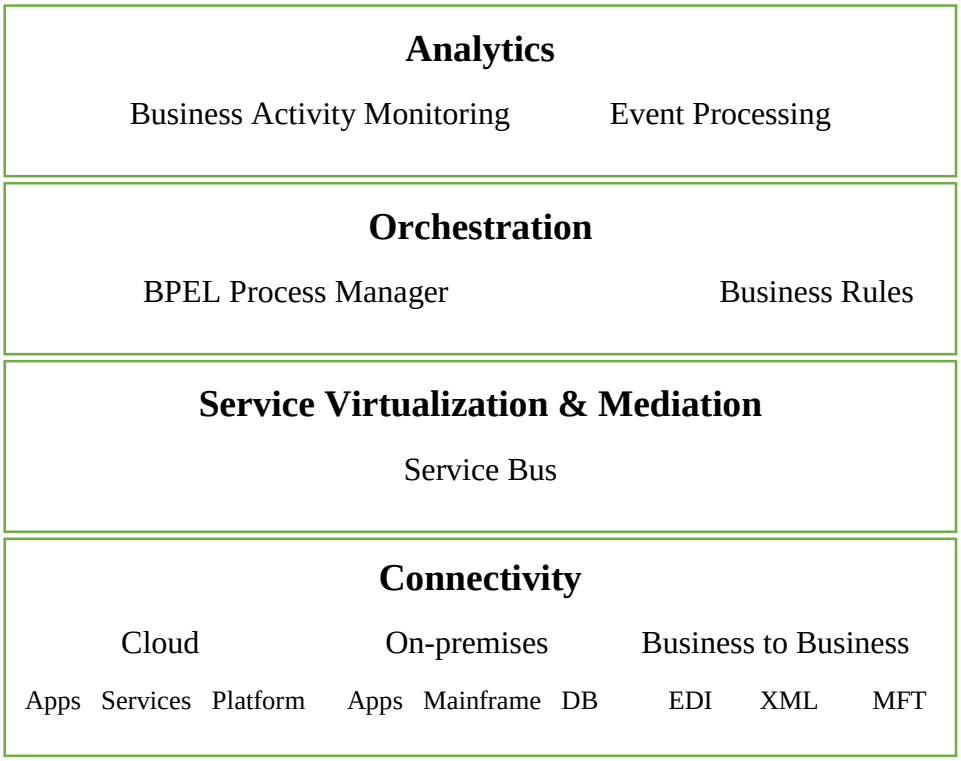
Bằng cách chuyển các chức năng của hệ thống thông tin thành các dịch vụ và chia sẻ trong doanh nghiệp, SOA thúc đẩy việc tái sử dụng. Điều này cũng đồng nghĩa với việc tiết kiệm chi phí và tăng hiệu quả đầu tư cho hạ tầng CNTT.

Một số giải pháp SOA

1. Bộ giải pháp của Oracle

Oracle SOA Suite là một bộ phần mềm sử dụng để xây dựng, triển khai và quản lý các hệ thống theo SOA. Bộ phần mềm này được cấu trúc gồm bốn lớp chính: kết nối (connectivity), áo hóa và trung gian dịch vụ (service virtualization & mediation), kết hợp (orchestration), phân tích (analytics).

Lớp kết nối cho phép các ứng dụng kết nối các nguồn dữ liệu trong và ngoài hệ thống. Lớp áo hóa và trung gian do Oracle Service Bus đảm trách. Oracle Service Bus là một ESB (enterprise service bus). ESB này giúp cho bên sử dụng dịch vụ độc lập với sự phức tạp cũng như những thay đổi từ phía ứng dụng chủ (back- end). Các chức năng chính trong lớp kết hợp được thực hiện bởi Oracle BPEL Process Manager. Đây là công cụ dùng để kết nối các dịch vụ riêng lẻ thành các quy trình. Việc kết hợp được dựa trên ngôn ngữ BPEL. Lớp phân tích bao gồm các công cụ cho phép người dùng tổng hợp thông tin từ nhiều nguồn trong hệ thống và biểu diễn theo nhiều hình thức. Lớp này giúp cho các nhà quản lý có một giao diện tổng hợp về các vấn đề liên quan đến nghiệp vụ của công ty/tổ chức và từ đó có thể đưa ra các quyết định. Oracle còn có Oracle Enterprise Repository là ứng dụng phục vụ cho quản trị SOA.



Hình 4.8: Kiến túc Oracle SOA Suite

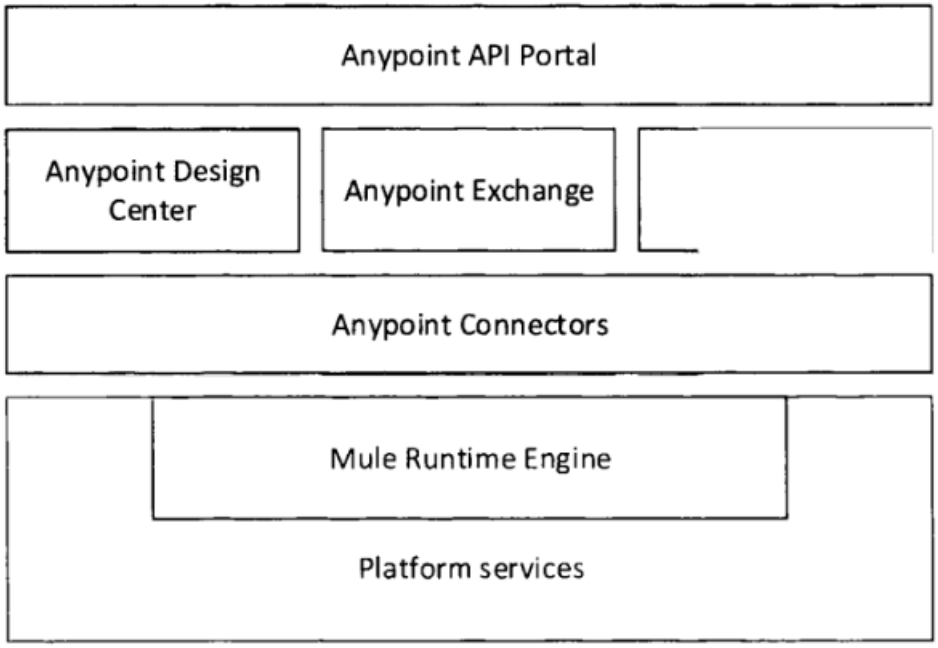
2. Bộ giải pháp của IBM

IBM là công ty hàng đầu về cung cấp các giải pháp SOA. IBM SOA Foundation [15] là một tập các phần mềm, các kỹ năng và mẫu để xây dựng các hệ thống SOA. IBM SOA Foundation được xây dựng dựa trên kiến trúc SQA tham khảo của IBM. Kiến trúc này đề xuất những dịch vụ, nền tảng cần thiết để xây dựng các hệ thống SOA.

Các phần mềm trong IBM SOA Foundation phục vụ cho các giai đoạn mô hình (ví dụ WebSphere Business Modeler), tổng hợp (ví dụ WebSphere Integration Developer và Rational Tester for SOA Quality), triển khai (ví dụ WebSphere ESB, WebSphere MQ và WebSphere Adapters), và quản lý của các hệ thống SOA (ví dụ WebSphere Business Monitor và Tivoli Composite Application Manager for SOA).

3. Bộ giải pháp của MuleSoft

Giải pháp SOA của MuleSoft được tích hợp trong bộ phần mềm Anypoint Platform. Kiến trúc của Anypoint Platform được trình bày ở hình dưới:



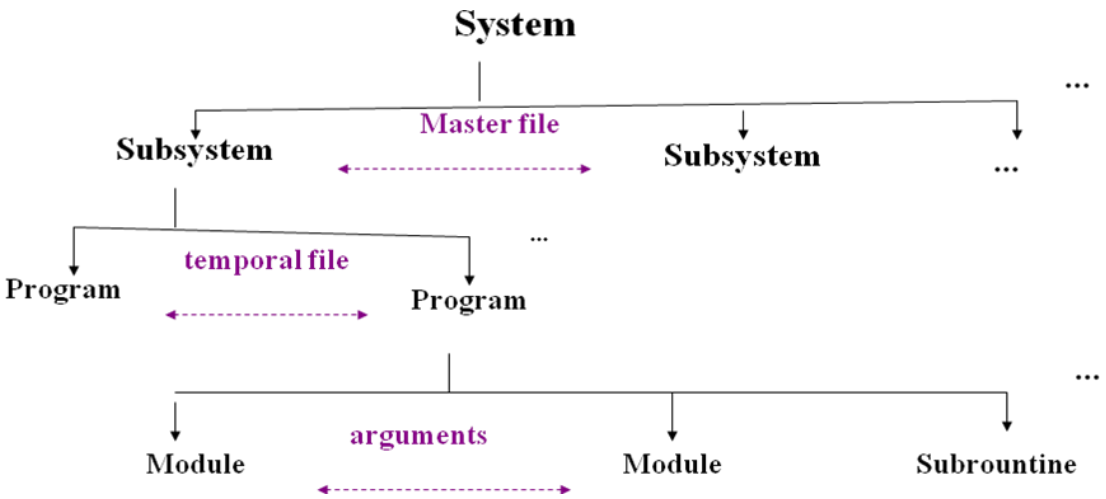
Hình 4.9: Kiến trúc của Anypoint Platform

Mô trường chạy Mule (Mule Runtime Engine) đóng vai trò là ESB trong bộ giải pháp. Mule cung cấp khả năng kết nối với các ứng dụng cũng như nguồn dữ liệu có sẵn. Anypoint Connectors cung cấp các cơ chế kết nối đến các ứng dụng trên đảm mây hoặc đến các ứng dụng được triển khai trên cơ sở hạ tầng của doanh nghiệp. Anypoint Designer Center là ứng dụng được sử dụng để thiết kế các API, luồng xử lý và các đầu kết nổi. Anypoint Management Center là trung tâm quản lý của bộ giải pháp. Thành phần này quản lý người dùng API, các thỏa thuận về chất lượng dịch vụ, các dòng tích hợp và những khía cạnh khác của nền tảng trên một giao diện Web thống nhất.

Anypoint Exchange là nơi người dùng có thể tìm kiếm các đầu kết nối, mẫu và các ví dụ được chia sẻ. API Portal là nơi các nhà phát triển có thể tìm các AP1 cũng như các tài liệu, ví dụ về cách sử dụng các API.

4.2.1.2 Phân rã hệ thống

Sau khi cấu trúc hệ thống đã được lựa chọn, ta cần phải xác định phương pháp phân rã các hệ thống con thành các mô-đun. Kiến trúc hệ thống nhìn từ cấu trúc phân cấp có dạng như hình 4.6 sau:



Hình 4.10: Kiến trúc hệ thống nhìn từ cấu trúc phân cấp

Hệ thống con là một hệ thống có thể vận hành một cách độc lập, có thể sử dụng một số dịch vụ được cung cấp bởi các hệ thống con khác hoặc cung cấp dịch vụ cho các hệ thống con khác sử dụng.

Mô-đun là một thành phần của hệ thống cung cấp các dịch vụ cho các thành phần khác, nhưng nó thường không được coi như là một hệ thống riêng rẽ, độc lập.

Có hai cách để phân rã các hệ thống con thành các mô-đun: Phân rã hướng đối tượng, Pipeline hướng chức năng hoặc luồng dữ liệu.

*** Phân rã hướng đối tượng**

Mô hình kiến trúc hướng đối tượng cấu trúc hệ thống thành một tập hợp các đối tượng gắn kết lỏng dựa trên các giao diện (lớp) đã được định nghĩa.

Phân rã hướng đối tượng liên quan tới việc xác định lớp đối tượng, các thuộc tính và phương thức của nó. Khi cài đặt lớp, các đối tượng sẽ được tạo ra từ các lớp này và có một số mô hình điều khiển được sử dụng để kết hợp các phương thức của đối tượng.

Ưu điểm:

- Đối tượng được gắn kết lỏng nên khi thay đổi cách cài đặt chúng có thể không ảnh hưởng tới các đối tượng khác.
- Đối tượng phản ánh thực thể trong thế giới thực.
- Các ngôn ngữ lập trình hướng đối tượng được sử dụng rộng rãi.

Hạn chế: Khi giao diện của các thực thể trong thế giới thức hay thay đổi, phức tạp có thể gây khó khăn vì rất khó biểu diễn các thực thể như là các đối tượng.

*** Pipeline hướng chức năng**

Hệ thống được phân hoá thành các module chức năng. Chúng nhận các dữ liệu chuyển hoá chúng rồi lại đưa ra các dữ liệu kết quả.

Trong mô hình luồng dữ liệu, các bộ biến đổi xử lý dữ liệu đầu vào và tạo dữ liệu đầu ra. Dữ liệu được chảy tuần tự theo luồng từ bộ biến đổi này sang bộ khác. Mỗi bước của quy trình giống như một phép biến đổi.

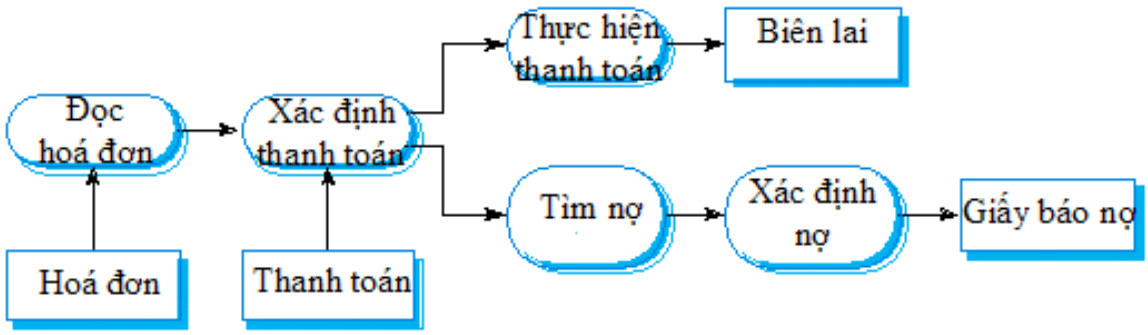
Ưu điểm:

- Nó hỗ trợ việc sử dụng lại các biến đổi.
- Nó phù hợp với suy nghĩ của mọi người quan niệm về dữ liệu được xử lý theo luồng có đầu vào và đầu ra.
- Thêm các xử lý khác vào hệ thống đơn giản.
- Dễ thực hiện xử lý song song hoặc tuần tự.

Nhược điểm:

- Cần phải có một định dạng chung cho các dữ liệu để có thể xử lý bởi mọi bộ biến đổi.
- Các hệ thống tương tác khó được viết theo mô hình luồng dữ liệu

Ví dụ: Mô hình luồng dữ liệu của hệ thống xử lý hoá đơn



Hình 4.11: Mô hình luồng dữ liệu của hệ thống xử lý hoá đơn.

4.2.1.3 Điều khiển hệ thống

Điều khiển hệ thống là hoạt động xác lập mô hình điều khiển giữa các phần khác nhau của hệ thống đã được xác định. Điều khiển hệ thống liên quan đến luồng điều khiển giữa các hệ thống con. Có 2 loại chiến lược điều khiển:

- **Điều khiển tập trung:** một hệ thống con chịu trách nhiệm kiểm soát, khởi tạo hoặc dừng các hệ thống con khác.

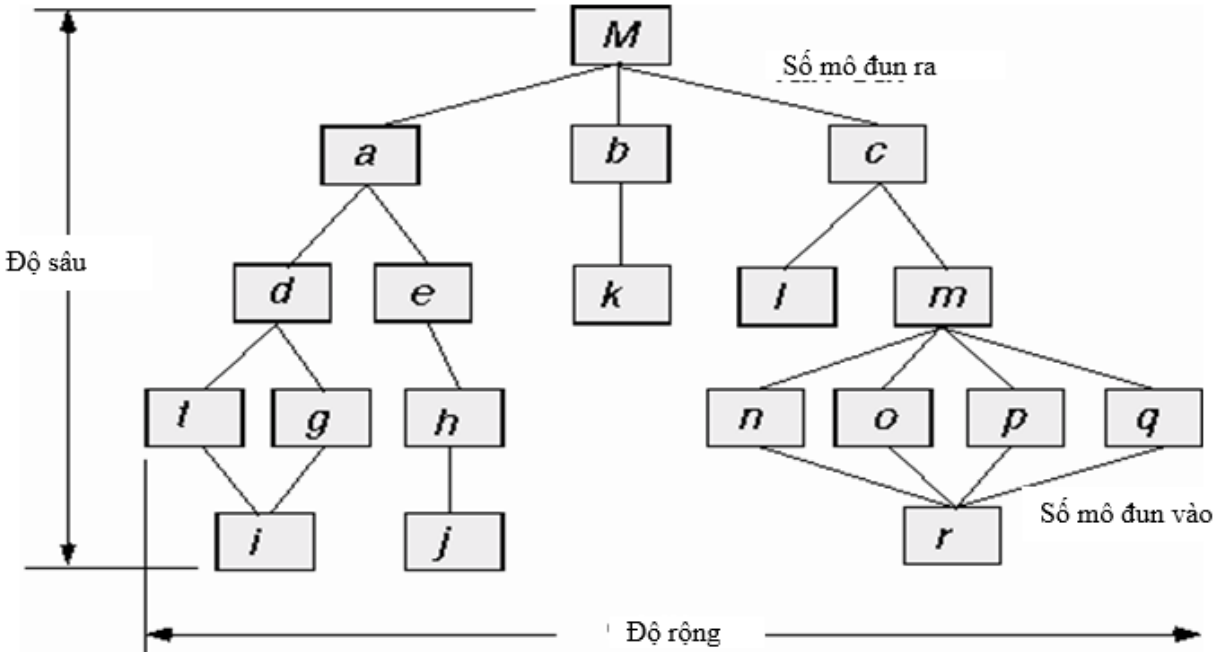
- **Điều khiển hướng sự kiện**: mỗi hệ thống đáp ứng với các sự kiện xảy ra từ các hệ thống con khác hoặc từ môi trường của hệ thống.

a) Điều khiển tập trung

Hệ thống con điều khiển chịu trách nhiệm quản lý việc thực hiện của các hệ thống con khác. Chiến lược điều khiển tập trung gồm 2 loại mô hình:

1. Mô hình gọi - trả lời (call-return)

Mô hình này phù hợp với các mô hình thủ tục top - down. Việc điều khiển bắt đầu ở đỉnh của cây thủ tục và di chuyển xuống. Mô hình này có thể áp dụng có các hệ thống tuần tự. Mô hình này được biểu diễn như hình 4.12



Hình 4.12: Mô hình gọi - trả lời

2. Mô hình quản lý

Thường áp dụng cho các hệ thống song song. Một cấu thành hệ thống được thiết kế như là một bộ quản trị và điều khiển việc khởi động, kết thúc và phối hợp các phân hệ khác.

b) Điều khiển hướng sự kiện

Mô hình hệ thống điều khiển bởi sự kiện có nhiều kiểu khác nhau như :

Mô hình phát tin:

Trong mô hình này, về nguyên tắc, một sự kiện được thông báo cho các phân hệ. Các phân hệ được thiết kế để điều khiển sự kiện này sẽ tự quyết việc trả lời. Mô hình này hiệu quả với các phân hệ được phân bố trên các máy tính khác nhau trên mạng.

Ưu điểm của nó là việc phát triển tương đối đơn giản. Một phân hệ mới xử lý một lớp sự kiện mới có thể được tích hợp khi ghi nhận các sự kiện này vào bộ điều khiển sự kiện. Mỗi phân hệ có thể kích hoạt mọi phân hệ khác không cần biết tên và vị trí của các phân hệ đó.

Nhược điểm của mô hình này là phân hệ không biết sự kiện có được xử lý hay không và khi nào được xử lý. Rất có thể hai phân hệ khác nhau cùng sinh một sự kiện và có thể gây xung đột.

Mô hình điều khiển ngắt:

Có một hệ thống bên ngoài được sử dụng riêng cho việc theo dõi các ngắt bên ngoài và được chuyển tới các phân hệ tương ứng. Mô hình này phù hợp với các hệ thống thời gian.

Ưu điểm của nó là cho phép đáp ứng nhanh nhất với các sự kiện.

Nhược điểm là việc lập trình phức tạp.

4.2.2 Đặc tả trừu tượng

Đặc tả trừu tượng: các đặc tả trừu tượng cho mỗi hệ con về các dịch vụ mà nó cung cấp cũng như các ràng buộc chúng phải tuân thủ.

Abstraction: là một cách nhìn của một đối tượng mà tập trung vào thông tin liên quan để cụ thể hóa mục đích và tránh bỏ xót thông tin". Trong bối cảnh của thiết kế phần mềm, 2 cơ chế trừu tượng hóa chìa khóa là tham số hóa và cụ thể hóa. Trừu tượng bởi tham số hóa trừu tượng đến từ biểu diễn dữ liệu chi tiết bởi biểu diễn dữ liệu như tên những tham số. Trừu tượng hóa bởi cụ thể hóa dẫn đến 3 loại trừu tượng: trừu tượng thủ tục, trừu tượng dữ liệu và trừu tượng điều khiển (trừu tượng tương tác lẫn nhau)

- ✓ Trừu tượng thủ tục (trừu tượng hàm) cung cấp cơ chế để trừu tượng những thủ tục dễ định nghĩa hoặc những thao tác thành những thực thể. Trưu tượng thủ tục đã được áp dụng rộng rãi và các ngôn ngữ lập trình hầu như tất cả đều cung cấp hỗ trợ khái niệm này (ví dụ pascal, java,...)
- ✓ Trừu tượng dữ liệu: đây là nguyên tắc chính trong hướng đối tượng. Trong kiểu trừu tượng này, thay vì chỉ tập trung vào thao tác, chúng ta tập trung và dữ liệu đầu tiên và sau đó những thao tác tác động lên dữ liệu. Một ví dụ đơn giản là queue data và những thao tác liên quan như add() and delete(). Trong trừu tượng hóa thủ tục, thao tác add và delete chỉ là riêng biệt không có quan hệ với dữ liệu. Điểm mạnh của trừu tượng hóa dữ liệu so với trừu tượng hóa thủ tục là dữ liệu và các thao tác liên quan được đưa ra vì vậy rất dễ để sửa code khi dữ liệu thay đổi.
- ✓ Trừu tượng điều khiển liên quan để sử dụng các chương trình con và liên quan đến luồng điều khiển.

Câu hỏi

Câu 1. Giai đoạn đầu tiên trong quy trình thiết kế hệ thống là gì?

a) Thiết kế giao diện
b) Thiết kế thành phần
c) Thiết kế cấu trúc dữ liệu
d) Thiết kế kiến trúc

Câu 2. Thiết kế hướng đối tượng bao gồm trình tự 5 hoạt động nào?

a) Tìm đối tượng, tìm lớp, tìm mối quan hệ, thiết kế tương tác và làm mịn.
b) Tìm lớp, tìm mối quan hệ, tìm các đối tượng, thiết kế tương tác, thiết kế giao thức và làm mịn
c) Tìm đối tượng, tìm lớp, thiết kế tương tác, làm mịn và mã hóa
d) Tìm đối tượng, tìm lớp, tìm thuộc tính, thiết kế tương tác và mã hóa.

Câu 3. Đầu vào của tiến trình thiết kế hệ thống là gì ?

a) Tài liệu đặc tả yêu cầu phần mềm
b) Tài liệu đặc tả thành phần hệ thống
c) Tài liệu triển khai hệ thống
d) Kết quả nghiên cứu tính khả thi của hệ thống.

Câu 4. Quy trình thiết kế giao diện phải trải qua các hoạt động nào để đảm bảo tính chất khách quan của giao diện thiết kế ?

a) Tiếp xúc và trao đổi với khách hàng trong quá trình thiết kế giao diện
b) Tìm hiểu kỹ các hoạt động của người sử dụng đối với hệ thống, lập mẫu thử giao diện, tinh chỉnh mẫu thử, đánh giá mẫu thử cùng người dùng sau đó mới tiến hành cài đặt giao diện
c) Thiết kế phác thảo giao diện trên giấy, đánh giá giao diện cùng người dùng, thiết kế mô phỏng mẫu thử và cài đặt giao diện chính thức
d) Tiếp xúc trao đổi với khách hàng, cùng khách hàng thiết kế giao diện trên giấy, và cùng khách hàng đánh giá giao diện

Câu 5. Thiết kế giao diện người dùng là một quy trình lặp lại bao gồm sự cộng tác giữa những ai?

a) Người sử dụng và người thiết kế
b) Người sử dụng và người lập trình
c) Người phân tích và người lập trình
d) Người phân tích và người thiết kế

Bài 8: Thiết kế phần mềm (Số tiết: 03 tiết)

4.2 Quy trình thiết kế phần mềm (Tiếp theo)

4.2.3 Thiết kế giao diện người dùng

Một nguyên tắc quan trọng khi xây dựng một hệ thống phần mềm, đó là: người sử dụng không quan tâm đến cấu trúc bên trong của hệ thống, đơn giản hay phức tạp; cái mà họ có thể đánh giá được và cảm nhận được chính là giao diện tương tác giữa hệ thống và người sử dụng. Nếu người sử dụng cảm thấy giao diện không thích hợp, khó sử dụng thì rất có thể họ sẽ không sử dụng cả hệ thống; cho dù hệ thống đó có đáp ứng tất cả các chức năng nghiệp vụ mà họ muốn. Và như vậy, dự án của chúng ta sẽ thất bại.

Trong mục này, chúng ta sẽ nghiên cứu những vấn đề sau [4]:

- Các yếu tố liên quan đến giao diện người dùng
- Quy trình xây dựng giao diện người dùng

4.2.3.1 Giao diện người dùng [1]

a) Tác nhân con người trong thiết kế giao diện

Một nhân tố quan trọng ảnh hưởng tới quá trình thiết kế giao diện đó chính là người sử dụng hệ thống. Do đó, chúng ta phải tìm hiểu một số đặc điểm của người sử dụng có liên quan đến giao diện hệ thống:

- ✓ Khả năng nhớ tức thời của con người bị hạn chế: con người chỉ có thể nhớ ngay khoảng 7 loại thông tin. Nếu ta biểu diễn nhiều hơn 7 loại, thì có thể khiến người sử dụng không nhớ hết và gây ra các lỗi.
- ✓ Người sử dụng có thể gây ra lỗi: khi người sử dụng gây ra lỗi khiến hệ thống sẽ hoạt động sai, những thông báo không thích hợp có thể làm tăng áp lực lên người sử dụng và do đó, càng xảy ra nhiều lỗi hơn.
- ✓ Người sử dụng là khác nhau: con người có những khả năng khác nhau. Những người thiết kế không nên chỉ thiết kế giao diện phù hợp với những khả năng của chính họ.
- ✓ Người sử dụng thích các loại tương tác khác nhau: một số người thích hình ảnh, văn bản, âm thanh …

b) Các nguyên tắc thiết kế giao diên

Thiết kế giao diện phải phụ thuộc vào yêu cầu, kinh nghiệm và khả năng của người sử dụng hệ thống. Người thiết kế cũng nên quan tâm đến những giới hạn vật lý và tinh thần của con người và nên nhận ra rằng con người luôn có thể gây ra lỗi.

Không phải tất cả các nguyên tắc thiết kế giao diện đều có thể được áp dụng cho tất cả các giao diện. Sau đây là các nguyên tắc thiết kế giao diện:

- ✓ Sự quen thuộc của người sử dụng: giao diện phải được xây dựng dựa trên các thuật ngữ và các khái niệm mà người sử dụng có thể hiểu được hơn là những khái niệm liên quan đến máy tính. Ví dụ: hệ thống văn phòng nên sử dụng các khái niệm như thư, tài liệu, cặp giấy … mà không nên sử dụng những khái niệm như thư mục, danh mục …
- ✓ Thống nhất: hệ thống nên hiển thị ở mức thống nhất thích hợp. Ví dụ: các câu lệnh và menu nên có cùng định dạng …
- ✓ Tối thiểu hoá sự bất ngờ: nếu một yêu cầu được xử lý theo cách đã biết trước thì người sử dụng có thể dự đoán các thao tác của những yêu cầu tương tư.
- ✓ Khả năng phục hồi: hệ thống nên cung cấp một số khả năng phục hồi từ lỗi của người sử dụng và cho phép người sử dụng khôi phục lại từ chỗ bị lỗi. Khả năng này bao gồm cho phép làm lại, hỏi lại những hành động như xoá, huỷ …
- ✓ Hướng dẫn người sử dụng: như hệ thống trợ giúp, hướng dẫn trực tuyến …
- ✓ Tính đa dạng: hỗ trợ nhiều loại tương tác cho nhiều loại người sử dung khác nhau. Ví dụ: nên hiển thị phông chữ lớn với những người cận thị.

Tương tác giữa người sử dụng và hệ thống được chia thành 5 loại sau:

- Vận hành trực tiếp
- Lựa chọn menu
- Điền vào biểu mẫu (Form)
- Ngôn ngữ ra lệnh
- Ngôn ngữ tự nhiên

c) Biểu diễn thông tin

Biểu diễn thông tin có liên quan tới việc hiển thị các thông tin trong hệ thống tới người sử dụng. Thông tin có thể được biểu diễn một cách trực tiếp hoặc có thể được chuyển thành nhiều dạng hiển thị khác như: dạng đồ hoạ, âm thanh …

Thông tin cần biểu diễn được chia thành hai loại:

- Thông tin tĩnh: được khởi tạo ở đầu của mỗi phiên. Nó không thay đổi trong suốt phiên đó và có thể là ở dạng số hoặc dạng văn bản.

- Thông tin động: thay đổi trong cả phiên sử dụng và sự thay đổi này phải được người sử dụng quan sát.

Các nhân tố ảnh hưởng tới việc hiển thị thông tin:

- Người sử dụng thích hiển thị một phần thông tin hay quan hệ dữ liệu?

- Giá trị của thông tin thay đổi nhanh như thế nào? Sự thay đổi đó có cần phải thể hiện ngay lập tức hay không?

- Người sử dụng có phải thực hiện các hành động để đáp ứng với sự thay đổi không?

- Có phải là giao diện vận hành trực tiếp không?

- Thông tin ở dạng văn bản hay dạng số? Các giá trị quan hệ có quan trọng không?

- Biểu diễn digital hay analogue ?

Nếu chúng ta cần hiển thị số lượng lớn thông tin thì nên trực quan hoá dữ liệu. Trực quan hoá có thể phát hiện ra mối quan hệ giữa các thực thể và các xu hướng trong dữ liệu. Ví dụ: thông tin về thời tiết được hiển thị dưới dạng biểu đồ, trạng thái của mạng điện thoại nên được hiển thị bởi các nút có liên kết với nhau.

Chúng ta thường sử dụng màu trong khi thiết kế giao diện. Màu bổ sung thêm một chiều nữa cho giao diện và giúp cho người sử dụng hiểu được những cấu trúc thông tin phức tạp. Màu có thể được sử dụng để đánh dấu những sự kiện ngoại lệ.

Tuy nhiên, khi sử dụng màu để thiết kế giao diện có thể gây phản tác dụng. Do đó, chúng ta nên quan tâm tới một số hướng dẫn sau :

- Giới hạn số lượng màu được sử dụng và không nên lạm dụng việc sử dụng màu.

- Thay đổi màu khi thay đổi trạng thái của hệ thống

- Sử dụng màu để hỗ trợ cho những nhiệm vụ mà người sử dụng đang cố gắng thực hiện.

- Sử dụng màu một cách thống nhất và cẩn thận.

- Cẩn thận khi sử dụng các cặp màu.

Khi người sử dụng tương tác với hệ thống, rất có thể xảy ra lỗi và hệ thống phải thông báo cho người sử dụng biết lỗi gì đã xảy ra hoặc đã có chuyện gì xảy ra với hệ thống. Do đó, thiết kế thông báo lỗi vô cùng quan trọng. Nếu thông báo lỗi nghèo nàn có thể làm cho người sử dụng từ chối hơn là chấp nhận hệ thống.

Vì vậy, thông báo lỗi nên ngắn gọn, xúc tích, thống nhất và có cấu trúc. Việc thiết kế thông báo lỗi nên dựa vào kỹ năng và kinh nghiệm của người sử dụng.

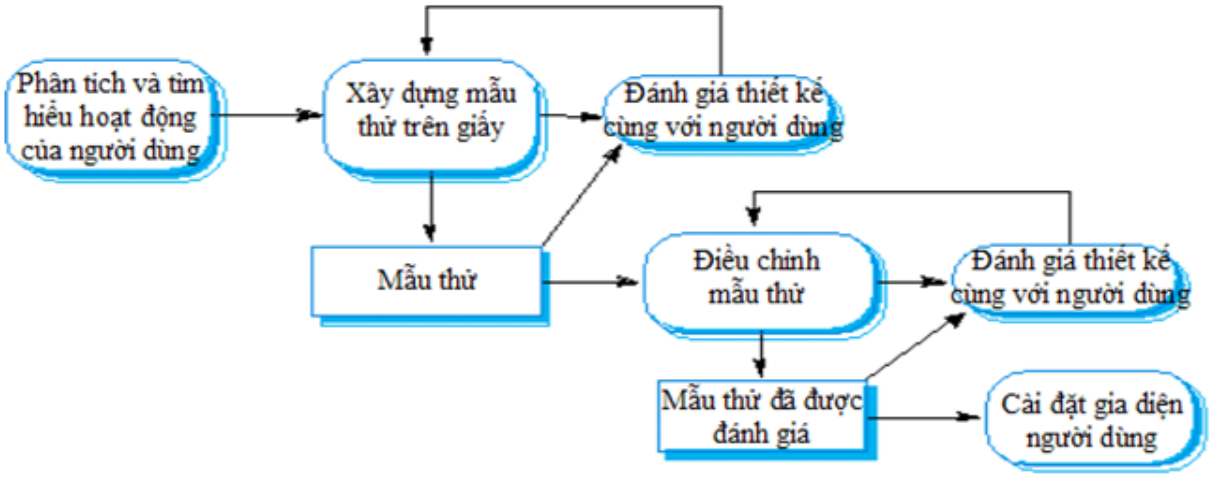
4.2.3.2 Quy trình thiết kế giao diện người dùng

Thiết kế giao diện người dùng là một quy trình lặp lại bao gồm sự cộng tác giữa người sử dụng và người thiết kế. Trong quy trình này gồm 3 hoạt động cơ bản :

- Phân tích người sử dụng : tìm hiểu những gì người sử dụng sẽ làm với hệ thống.

- Lập mẫu thử hệ thống : xây dựng một tập các mẫu thử để thử nghiệm

- Đánh giá giao diện : thử nghiệm các mẫu thử cùng với người sử dụng.



Hình 4.13: Quy trình thiết kế giao diện người dùng

a) Phân tích người sử dụng

Nếu ta không hiểu rõ những gì người sử dụng muốn làm với hệ thống, thì ta sẽ không thể thiết kế được một giao diện hiệu quả. Phân tích người sử dụng phải được mô tả theo những thuật ngữ để người sử dụng và những người thiết kế khác có thể hiểu được.

Các ngữ cảnh mà ta mô tả thao tác ở trong đó là một trong những cách mô tả phân tích người dùng. Ta có thể lấy được rất nhiều yêu cầu của người sử dụng từ đó.

Các kỹ thuật phân tích:

- Phân tích nhiệm vụ: mô hình hoá các bước cần thực hiện để hoàn thành một nhiệm vụ.

- Phân tích nhiệm vụ phân cấp.

- Phỏng vấn và trắc nghiệm: hỏi người sử dụng về những gì mà họ làm. Khi phỏng vấn, chúng ta nên dựa trên những câu hỏi có kết thúc mở. Sau đó, người sử dụng cung cấp những thông tin mà họ nghĩ rằng nó là cần thiết; nhưng không phải tất cả các thông tin đó là có thể được sử dụng. Ngoài ra, chúng ta có thể thực hiện phỏng vấn với cả nhóm người sử dụng, điều đó cho phép người sử dụng thảo luận với nhau về những gì họ làm.

- Mô tả: quan sát người sử dụng làm việc và hỏi họ về những cách mà không được biết tới. Nên nhớ rằng có nhiều nhiệm vụ của người sử dụng thuộc về trực giác và rất khó để mô tả và giải thích chúng. Dựa trên kỹ thuật này ta có thể hiểu thêm về các ảnh hưởng xã hội và tổ chức tác động tới công việc đó.

b) Lập mẫu thử giao diện người dùng

Mẫu thử cho phép người sử dụng có được những kinh nghiệm trực tiếp với giao diện. Nếu không có những kinh nghiệm trực tiếp như vậy thì không thể đánh giá được khả năng có thể sử dụng được của giao diện.

Lập mẫu thử là một quy trình gồm 2 trạng thái:

- Lập các mẫu thử trên giấy.

- Tinh chỉnh mẫu thử và xây dựng chúng

Các kỹ thuật lập mẫu thử:

- Mẫu thử hướng nguyên mẫu: sử dụng công cụ như Macromedia Director để xây dựng một tập hợp các nguyên mẫu và màn hình. Khi người sử dụng tương tác với chúng thì màn hình sẽ thay đổi để hiển thị trạng thái kế tiếp.

- Lập trình trực quan: sử dụng các ngôn ngữ được thiết kế cho việc phát triển nhanh như Visual Basic.

- Mẫu thử dựa Internet: sử dụng web browser và script.

c) Đánh giá giao diện người dùng

Ta nên đánh giá bản thiết kế giao diện người dùng để xác định khả năng phù hợp của nó. Tuy nhiên, việc đánh giá trên phạm vi rộng tốn nhiều chi phí và không thể thực hiện được đối với hầu hết các hệ thống.

Các kỹ thuật đánh giá đơn giản:

- Trắc nghiệm lại các phản hồi của người sử dụng

- Ghi lại quá trình sử dụng mẫu thử của hệ thống và đánh giá nó.

- Lựa chọn những thông tin về việc sử dụng dễ dàng và các lỗi của người sử dụng.

- Cung cấp mã lệnh trong phần mềm để thu thập những phản hồi của người sử dụng một cách trực tuyến.

4.2.4 Thiết kế dữ liệu

Giai đoạn này bao gồm cả thiết kế các cấu trúc dữ liệu của chương trình và cơ sở dữ liệu. Hoạt động thiết kế cấu trúc dữ liệu nhằm lựa chọn và xây dựng mô hình biểu diễn thông tin cho hệ thống. Cách thức tổ chức dữ liệu của hệ thống cũng được đặc tả chi tiết dần ở các mức:

- ✓ Mô hình dữ liệu
- ✓ Cấu trúc dữ liệu

Chọn cách biểu diễn các đối tượng thiết kế có ảnh hưởng mạnh mẽ đến chất lượng phần mềm. Một số tiêu chí lựa chọn cách thức tổ chức dữ liệu của hệ thống**:**

- ✓ Thuận lợi cho các thao tác
- ✓ Tiết kiệm không gian
- ✓ Phản ánh đúng các dữ liệu thực tế

Việc chọn lựa cấu trúc dữ liệu ảnh hưởng lớn đến hiệu suất chương trình và nó tác động đến bản thân thuật toán bởi cấu trúc dữ liệu gắn bó mật thiết với thuật toán. Lựa chọn đúng đắn các cấu trúc dữ liệu giúp làm giảm không gian bộ nhớ của chương trình, giảm thời gian chạy, tăng tính khả chuyển và dễ bảo trì.

Một số nguyên tắc để làm giảm không gian lưu trữ dữ liệu:

- ✓ Đảm bảo tính đơn giản của cấu trúc lưu trữ,
- ✓ Trong một số trường hợp đừng lưu trữ, hãy tính lại khi cần thiết,
- ✓ Nén dữ liệu sau đó giải nén khi dùng,
- ✓ Sử dụng các nguyên tắc cấp phát bộ nhớ: chẳng hạn như cấp phát bộ nhớ động, ...

Các mức thiết kế cấu trúc dữ liệu cho hệ thống:

Thiết kế cấu trúc logic	**Thiết kế cấu trúc vật lý**
✓ Các quan hệ chuẩn ✓ Các khóa ✓ Các tham chiếu ✓ Các cấu trúc thao tác dữ liệu	✓ Các file ✓ Các kiểu dữ liệu ✓ Kích cỡ

4.2.5 Thiết kế thuật toán

Sau khi đã xác định được các yêu cầu mà hệ thống phải đáp ứng, để xử lý các yêu cầu này ta lựa chọn cách xử lý tối ưu nhất (sử dụng thuật toán tốt nhất, làm tăng tốc độ thực thi chương trình, cho ra kết quả chính xác). Thiết kế thủ tục nhằm mô tả các bước hoạt động của từng mô đun. Các phương pháp mô tả có thể là:

- ✓ Giải mã (pseudo code)
- ✓ Sơ đồ luồng (flow chart)
- ✓ Biểu đồ (diagram)
- ✓ Biểu đồ hoạt động (activity diagram)
- ✓ ISP

Khi lựa chọn thuật toán sử dụng, chúng ta nên xác định rõ bài toán. Trước khi bắt tay vào giải bài toán, hãy tìm hiểu kỹ các yêu cầu mà bài toán đặt ra và tận dụng mọi điều đã biết từ bài toán. Các thuật toán có ảnh hưởng quan trọng đến các hệ thống phần mềm và đặc biệt chúng tăng nhanh tốc độ vận hành của chương trình.

Một số nguyên tắc làm tăng tốc độ vận hành của chương trình:

- ✓ Lưu trữ các trạng thái cần thiết để tránh tính lại,

- ✓ Tiền xử lý thông tin để đưa vào các cấu trúc dữ liệu,
- ✓ Sử dụng các thuật toán thích hợp,
- ✓ Sử dụng các kết quả được tích luỹ,...

4.3 Case study: Thiết kế phần mềm

a) Mục tiêu: Hoàn thành đặc tả các yêu cầu cho bài toán: sử dụng UML

b) Yêu cầu

Đặc tả thiết kế phần mềm cụ thể:

- Choice of Architecture design
- Architectural Presentation
- Package design
- Detail design: Class diagram, Sequence diagram, ….
- Database design

c) Hướng dẫn, gợi ý: [33 – Chapter 4]

Câu hỏi

Câu 1. Tác nhân ca sử dụng luôn là con người, không phải là các thiết bị hệ thống?

a) Sai
b) Đúng

Câu 2. Yếu tố nào có ảnh hưởng đến việc thiết kế kiến trúc phần mềm?

a) Công nghệ sử dụng
b) Yêu cầu về thuộc tính chất lượng
c) Chiến lược triển khai hệ thống
d) Tất cả các phương án trên

Câu 3. Biểu đồ ngữ cảnh được sử dụng để làm gì?

a) Mô tả cấu trúc của hệ thống
b) Xác định phạm vi của hệ thống với môi trường bên ngoài
c) Mô tả các ràng buộc hệ thống
d) Mô tả quan hệ giữa các đối tượng

Câu 4. Những biểu đồ UML nào sau đây được sử dụng để mô hình hoá hệ thống?

a) Biểu đồ ca sử dụng
b) Biểu đồ cộng tác
c) Biểu đồ triển khai

d) Tất cả các phương án trên

Câu 5. Đặc điểm nào sau đây được sử dụng để đánh giá một bản thiết kế tốt?

a) Tính gắn kết giữa các mô-đun, tính cố kết trong một mô đun và khả năng tái sử dụng
b) Giải pháp đầy đủ cho mọi các yêu cầu trong pha phân tích
c) Chứa mọi trường hợp kiểm thử của các thành phần phần mềm
d) Cung cấp một mô tả hoàn thiện về sản phẩm phần mềm

Bài tập cuối chương

Bài 4.1. Hãy trình bày các hoạt động trong quá trình thiết kế hệ thống. Hãy nêu tầm quan trọng của thiết kế hệ thống.

Bài 4.2. Hãy trình bày quy trình thiết kế kiến trúc phần mềm

Bài 4.3. Hãy nêu ưu điểm, nhược điểm của các mô hình: Kho dữ liệu dùng chung, mô hình client – server.

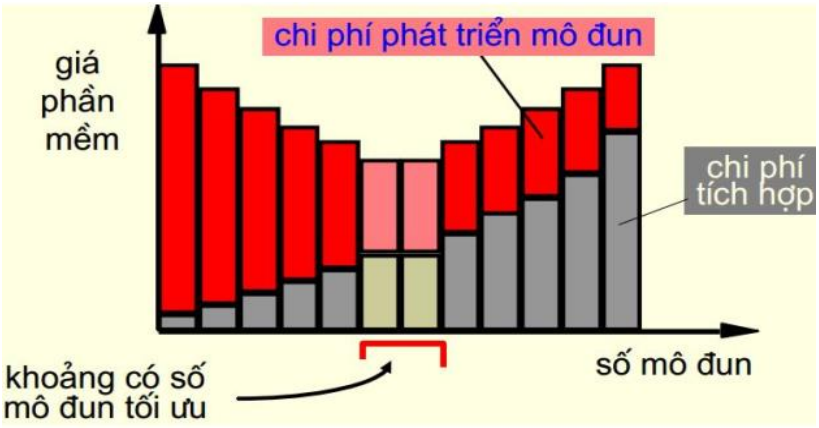
Bài 4.4. Hãy nêu và giải thích các đặc điểm chính của dịch vụ trong SOA. Giải thích vì sao SOA thường không được sử dụng cho các hệ thống có yêu cầu chặt chẽ về hiệu năng?

Bài 4.5. Quy trình thiết kế giao diện phải trải qua các hoạt động nào để đảm bảo tính chất khách quan của giao diện thiết kế? Tại sao phải lập mẫu thử giao diện mô phỏng?

Bài 4.6. Hãy trình bày mô hình được sử dụng để thiết kế chi tiết các thủ tục

Bài 4.7. Trình bày mối quan hệ giữa các giai đoạn phân tích - thiết kế - thực hiện - kiểm thử - bảo trì. Hãy liệt kê các công việc cần được thực hiện ở giai đoạn thiết kế phần mềm. Vì sao nói "Trên thực tế phân tích và thiết kế không có sự tách biệt nhau mà hai giai đoạn này được tiến hành song song và bổ sung cho nhau"?

Bài 4.8. Trình bày khái niệm mô đun. Thông qua hình dưới, hãy phân tích và giải thích vì sao phải phân chia mô đun một cách tối ưu (số lượng mô đun nhiều quá hoặc ít quá thì không tốt, các yếu tố ảnh hưởng: giá PM, độ phức tạp, công sức thực hiện,..)?



Bài 4.9. Khi thiết kế phần mềm, việc che giấu thông tin và phân chia mô đun có quan hệ như thế nào? Vì sao trong thiết kế phần mềm cần thực hiện việc che giấu thông tin? Hãy liệt kê các lợi ích của việc che giấu thông tin.

Bài 4.10. Trình bày đặc điểm của tính ghép nối và kết dính trong phân chia mô đun. Yếu tố ghép nối và kết dính giữa các mô đun có ý nghĩa gì trong thiết kế? Vì sao khi phân chia mô đun phải tăng độ kết dính và giảm sự liên kết?

CHƯƠNG V. CÀI ĐẶT PHẦN MỀM

Nội dung chính của chương

5.1 Tổng quan về cài đặt phần mềm

5.2 Phương pháp luận lập trình

5.3 Một số nguyên tắc chung trong lập trình

5.4 Tổ chức, quản lý và chia sẻ mã nguồn

Mục tiêu cần đạt được của chương

- Nắm được các phương pháp lập trình.
- Biết một số nguyên tắc chung trong lập trình.
- Biết lựa chọn ngôn ngữ phù hợp để cài đặt phần mềm

Bài 9: Tên bài: Cài đặt phần mềm (Số tiết: 03 tiết)

5.1 Tổng quan về cài đặt phần mềm

Kỹ nghệ phần mềm bao gồm tất cả các hoạt động liên quan đến phát triển phần mềm từ các yêu cầu ban đầu của hệ thống cho đến bảo trì và quả nlý hệ thống được triển khai. Tất nhiên, một giai đoạn quan trọng của quá trình này là cài đặt hệ thống, nơi ban tạo ra các bản thực thi của phần mềm. Việc thực hiện này có thể liên quan đến việc sử dụng các ngôn ngữ lập trình để tạo ra hệ thống.

Trong nội dung chương này sẽ đề cập tới các nguyên tắc lập trình, và một số kinh nghiệm trong lập trình, tập trung vào các vấn đề về làm thế nào để có thể lập trình tốt cho các hệ thống mà không đi vào cụ thể một ngôn ngữ lập trình nào cả.

5.2 Phương pháp luận lập trình

Lập trình là công đoạn quan trọng chủ chốt và không thể thiếu để tạo ra sản phẩm phần mềm. Phần mềm càng trở nên đa dạng và ngành công nghiệp phần mềm càng phát triển thì người ta càng thấy rõ tầm quan trọng của phương pháp lập trình. Phương pháp lập trình tốt không chỉ đảm bảo tạo ra phần mềm tốt mà còn hỗ trợ thiết kế phần mềm có tính mở và hỗ trợ khả năng sử dụng lại các mô đun. Nhờ đó có thể dễ dàng bảo trì, nâng cấp phần mềm cũng như giảm chi phí phát triển phần mềm. Có thể chia phương pháp lập trình thành 2 loại:

- Lập trình mệnh lệnh (imperative programming): Bao gồm lập trình tuyến tính, lập trình thủ tục, lập trình hướng đối tượng.

- Lập trình khai báo: Bao gồm lập trình Hàm, lập trình ngôn ngữ thế hệ thứ 4.

5.2.1 Lập trình mệnh lệnh (Imperrative Programming)

5.2.1.1 Lập trình tuần tự/tuyến tính

Chương trình bao gồm một tập các câu lệnh được thực hiện một cách tuần tự. Không có các câu lệnh có cấu trúc/điều khiển trong chương trình (for, while, if, ...). Để thực hiện các hoạt động điều khiển chương trình thường lạm dụng các lệnh nhảy goto, thiếu khả năng khai báo các biến cục bộ trong chương trình. Kỹ thuật lập trình này có những hạn chế là độ ghép nối của chương trình là cao, điều này dẫn đến chương trình khó hiểu, khó sửa và dễ sinh lỗi.

Các ngôn ngữ thường dùng cho lập trình kỹ thuật này là các ngôn ngữ thế hệ 1, 2 như hợp ngữ, basic, ...

5.2.1.2 Lập trình có cấu trúc/thủ tục

Lập trình có cấu trúc là kỹ thuật lập trình được sử dụng khá phổ biến. Chương trình được chia thành nhiều module chương trình con, và cho phép sử dụng các biến cục bộ trong mỗi chương trình con. Mỗi khi chương trình cần thực hiện công việc của các mô đun này ta chỉ việc gọi đến chúng. Điều khiển chương trình sử dụng kỹ thuật này phong phú hơn so với kỹ thuật lập trình tuyến tính. Các ngôn ngữ lập trình trợ giúp kỹ thuật này cũng cung cấp khá nhiều lệnh điều khiển/có cấu trúc như các lệnh for, while, if, ...giúp ta điều khiển của hoạt động của chương trình. Kỹ thuật lập trình này hạn chế/cấm dùng lệnh goto. Kỹ thuật lập trình này phù hợp với kết quả phân tích thiết kế hướng chức năng. Chương trình sử dụng kỹ thuật lập trình này dễ hiểu hơn và an toàn hơn so với kỹ thuật lập trình trước đó.

Các ngôn ngữ lập trình thế hệ 2, 3 được dùng cho kỹ thuật lập trình này như Pascal, C, Fortran,...

5.2.1.3 Lập trình hướng đối tượng

Lập trình hướng đối tượng là xu hướng phát triển của phương pháp lập trình hiện đại. Phương pháp này hỗ trợ các khái niệm hướng đối tượng như kế thừa, bao gói, che dấu thông tin, ...Chương trình được cấu thành từ một tập hợp các lớp đối tượng. Các đối tượng thuộc lớp trao đổi thông tin với nhau qua việc truyền các thông điệp. Chương trình sử dụng kỹ thuật lập trình này có ưu điểm cục bộ dữ liệu và thao tác hơn, dễ tái sử dụng hơn, thuận tiện cho các ứng dụng lớn.

Các ngôn ngữ lập trình dùng cho kỹ thuật lập trình này như: Smalltalk, Java, C#

5.2.2 Lập trình khai báo (Declarative Programming)

5.2.2.1 Lập trình hướng hàm

Lập trình hướng hàm dựa trên nguyên tắc ghép nối dữ liệu. Việc trao đổi dữ liệu được thực hiện bằng cách truyền dữ liệu qua tham số và nhận giá trình trả về. Kỹ thuật

này loại bỏ toàn bộ các dữ liệu dùng chung. Chương trình sử dụng kỹ thuật lập trình là là một hàm. Nó được xây dựng để tính toán các biểu thức và thường ứng dụng trong các lĩnh vực tính toán và trí tuệ nhân tạo.

Ví dụ: Chương trình có cấu trúc và chương trình hướng hàm tương ứng

Chương trình có cấu trúc	Chương trình hướng hàm
Begin *NhapDL();* *XylyDL();* *XuatDL();* *End.*	Chương trình là một biểu thức/hàm *XuatDL(XulyDL(NhạpDL(...)))*

Ngôn ngữ lập trình cho kỹ thuật này là Lisp, Scheme, ...

5.2.2.2 Lập trình logic

Lập trình logic là kỹ thuật lập trình tách tri thức về bài toán ra khỏi kỹ thuật lập trình. Lập trình là mô tả, xây dựng cơ sở tri thức bằng các quy tắc, mệnh đề, các mục tiêu, ... Khi người dùng đặt ra câu hỏi, chương trình chạy bằng cách tự chứng minh để tìm kiếm đường đi đến mục tiêu.

Kỹ thuật lập trình này thường được sử dụng để xây dựng các hệ chuyên gia, các ứng dụng xử lý Ngôn ngữ tự nhiên. Ngôn ngữ dùng cho kỹ thuật lập trình này như Prolog, …

5.2.2.3 Kỹ thuật lập trình thế hệ thứ 4

Lập trình bằng cách khai báo. Ví dụ lập trình CSDL. Ngôn ngữ lập trình dùng cho kỹ thuật này như SQL, ...chúng có năng lực biểu diễn cao và giúp lập trình với tốc độ cao.

5.2.3 Chọn ngôn ngữ lập trình cho ứng dụng

Khi chọn ngôn ngữ cho ứng dụng, ta cần quan tâm đến các yếu tố sau:

- ✓ *Theo yêu cầu của khách hàng*: Nếu khách tự bảo trì phầm mềm
- ✓ *Các đặc trưng của ngôn ngữ*

- ***Năng lực***: Khả năng hỗ trợ các kiểu biến, các cấu trúc lệnh. Những ngôn ngữ bậc cao thường có các cấu trúc và các câu lệnh phong phú. Chúng hỗ trợ nhiều kiểu dữ liệu, hỗ trợ con trỏ, hỗ trợ hướng đối tượng và có thư viện phong phú. Do đó, ta nên dùng các ngôn ngữ bậc cao thay cho bậc thấp nếu không quá cần thiết phải dùng ngôn ngữ bậc thấp.

- ***Tính khả chuyển của ngôn ngữ***: Giúp chương trình thực hiện được trên nhiều máy tính, hệ điều hành khác nhau. Tính khả chuyển là yếu tố quan trọng của ngôn ngữ vì khi thay đổi phần cứng và thay đổi hệ điều hành chương trình vẫn có thể vận hành được. Java và các ngôn ngữ thông dịch (kịch bản, script) có tính khả chuyển. Để nâng cao tính khả chuyển của chương trình ta cần sử dụng các tính năng chuẩn của ngôn ngữ và dùng các kịch bản bất cứ khi nào có thể.

- ***Công cụ hỗ trợ của ngôn ngữ lập trình:*** Nên sử dụng ngôn ngữ có các công cụ hỗ trợ lập trình hiệu quả. Điều này giúp ta dễ dàng quá trình lập trình và bảo trì phần mềm sau này.

Trình biên dịch hiệu quả: Tốc độ biên dịch cao, khả năng tối ưu cao, có khả năng khai thác các tập lệnh và thiết bị phần cứng mới.

Có các công cụ trợ giúp hiệu quả: editor, debugger, linker, make...IDE (Integrated Dvelop Environment).

- ✓ Năng lực, kinh nghiệm của nhóm lập trình viên
- ✓ Miền ứng dụng của ngôn ngữ
    - Phần mềm hệ thống: Chương trình hiệu quả, vạn năng, dễ mở rộng. Ngôn ngữ hỗ trợ như C, C++
    - Hệ thời gian thực: Ngôn ngữ C, C++, ADA, Assembly
    - Hệ thống nhúng: C++, Java, ...
    - Phần mềm nghiệp vụ: Cơ sở dữ liệu (phần mềm Oracle, DB2, SQL Server, MySQL, ...), Ngôn ngữ lập trình như FoxPro, Cobol, VB, VC++...
    - Trí tuệ nhân tạo: Prolog, Lips, OPS5, ...
    - Mạng: .Net, Các công nghệ Java
    - Web: PHP, ASP, JSP, Perl, Java, Java Script, ...
    - Phần mềm khoa học kỹ thuật: cần tính toán chính xác, thư viện toán học mạnh, dễ dàng song song hóa. Fortran là ngôn ngữ được dùng phổ biến
- ✓ Phương pháp luận lập trình.

=> Chưa tồn tại ngôn ngữ đa năng cho mọi ứng dụng.

5.3 Một số nguyên tắc chung trong lập trình [1] [4]

- ✓ Đặt tên: Có ý nghĩa, gợi nhớ
- ✓ Trình bày: Rõ ràng, có cấu trúc, dễ hiểu
- ✓ Chú thích: Đầy đủ, dễ đọc

- ✓ Hạn chế sử dụng các cấu trúc khó kiểm soat: break, goto, continue, ...
- ✓ Viết lệnh rõ ràng, tránh ẩn ý
- ✓ Triển khai các biểu thức phức tạp
- ✓ Mỗi module nên giới hạn từ 25 -> 50 dòng
- ✓ Tránh sử dụng các số tối nghĩa
- ✓ Viết chương trình theo cấu trúc nhô thụt, mỗi lệnh/1 dòng

5.4 Tổ chức, quản lý và chia sẻ mã nguồn [1]

Trong quá trình phát triển phần mềm theo nhóm hay theo từng cá nhân, thường xuyên sẽ gặp phải nhiều vấn đề như:

- Làm thế nào để quản lý được các phiên bản của quá trình quản lý phần mềm?
- Làm thế nào để quản lý source code chung cho cả nhóm?

Để giải quyết vấn đề đó, có thể sử dụng các công cụ quản lý Sourcecode, trong số đó có một số công cụ được sử dụng phổ biến như GitHub, Subverssion (SVN),…

5.4.1 GitHub

GitHub là một dịch vụ cung cấp kho lưu trữ mã nguồn Git dựa trên nền web cho các dự án phát triển phần mềm. GitHub cung cấp cả phiên bản trả tiền lẫn miễn phí cho các tài khoản. Các dự án mã nguồn mở sẽ được cung cấp kho lưu trữ miễn phí. Tính đến tháng 4 năm 2016, GitHub có hơn 14 triệu người sử dụng với hơn 35 triệu kho mã nguồn, làm cho nó trở thành máy chủ chứa mã nguồn lớn trên thế giới. Github đã trở thành một yếu tố có sức ảnh hưởng trong cộng đồng phát triển mã nguồn mở. Thậm chí nhiều nhà phát triển đã bắt đầu xem nó là một sự thay thế cho sơ yếu lý lịch và một số nhà tuyển dụng yêu cầu các ứng viên cung cấp một liên kết đến tài khoản Github để đánh giá ứng viên

Vào ngày 4 tháng 6 năm 2018, Microsoft đã thông báo việc đạt được thỏa thuận mua lại GitHub với giá 7.5 tỷ Đô la Mỹ. Ngày chính thức chuyển nhượng quyền sở hữu không được công bố. Sự phát triển của nền tảng GitHub bắt đầu vào ngày 19 tháng 10 năm 2007. Trang web được đưa ra vào tháng 4 năm 2008 do Tom Preston-Werner, Chris Wanstrath, và PJ Hyett thực hiện sau khi nó đã được hoàn thành một vài tháng trước đó, xem như giai đoạn beta.

Dự án trên Github có thể được truy cập và thao tác sử dụng một giao diện dòng lệnh và làm việc với tất cả các lệnh Git tiêu chuẩn. Github cũng cho phép người dùng đăng ký và không đăng ký để duyệt kho công cộng trên trang web. Github cũng tạo ra nhiều client và plugin cho máy tính để bàn. Trang web cung cấp các chức năng mạng

xã hội như feed, theo dõi, wiki (sử dụng phần mềm Gollum Wiki) và đồ thị mạng xã hội để hiển thị cách các nhà phát triển làm việc trên kho lưu trữ.

Một người sử dụng phải tạo ra một tài khoản cá nhân để đóng góp nội dung lên Github, nhưng các kho mã nguồn công cộng có thể được duyệt và tải về với bất cứ ai. Với một người dùng đã đăng ký tài khoản, họ có thể thảo luận, quản lý, tạo ra các kho, đóng góp cho kho của người dùng khác, và xem xét thay đổi mã.

GitHub cũng có một dịch vụ khác: một trang web kiểu pastebin gọi là Gist [2], dùng để lưu trữ các đoạn mã; trong khi Github sẽ được cho lưu trữ các dự án lớn hơn. Một dịch vụ lưu trữ khác được gọi là Speaker Deck.

Các phần mềm chạy GitHub được viết bằng Ruby on Rails và Erlang bởi GitHub, Inc, phát triển Chris Wanstrath, PJ Hyett, và Tom Preston-Werner.

5.4.2 Subversion

Hệ thống Subversion (viết tắt SVN) là một hệ thống quản lý phần tài nguyên (code, hình ảnh, video…) của một dự án. Hệ thống có khả năng cập nhật, so sánh và kết hợp tài nguyên mới vào tài nguyên cũ được giới thiệu vào năm 2000 bởi công ty CollabNet (http://subversion.tigris.org). Đây là hệ thống hỗ trợ làm việc theo nhóm rất hiệu quả. SVN hoạt động theo phương thwucs Client/Server, code project sẽ được lưu trữ trên server (SVN hosting, Google code).

Phần mềm:

- Cho client: TortoiseSVN, Download:http://tortoisesvn.net/

- Cho server: VisualSVN – Server Download: http://tortoisesvn.net/downloads.html Các client có thể thao tác, cập nhật trực tiếp trên đó, mọi thay đổi của từng client sẽ được lưu lại.

SVN Subversion giúp giải quyết các vấn đề:

- Khi một nhóm làm việc trên cùng một project, việc nhiều người cùng chỉnh sửa nội dung của một file là điều không thể tránh khỏi. SVN Subversion cung cấp các chức năng để có thể thực hiện việc này một cách đơn giản và an toàn.

- SVN Subversion được thiết kế với mục đích thay thế hệ thống quản lý phiên bản Concurrent Versioning System (CVS) đã cũ và có nhiều nhược điểm. Subversion có thể được sử dụng để quản lý bất cứ hệ thống phiên bản nào.

- SVN Subversion là hệ thống quản lý source code tập trung (Centralized).

- SVN Subversion là hệ thống quản lý phiên bản mạnh mẽ, hữu dụng, và linh hoạt.

- SVN Subversion quản lý tập tin và thư mục theo thời gian.

- SVN Subversion giống như một hệ thống file server mà các client có thể download và upload file một cách bình thường.

- Điểm đặt biệt của SVN Subversion là nó lưu lại tất cả những gì thay đổi trên hệ thống file: file nào đã bị thay đổi lúc nào, thay đổi như thế nào, và ai đã thay đổi nó.

- SVN Subversion cũng cho phép recover lại những version cũ một cách chính xác. Các chức năng này giúp cho việc làm việc nhóm trở nên hiệu quả và an toàn hơn rất nhiều.

- Thông thường, client và server kết nối thông qua mạng LAN hoặc Internet. Client và server có thể cùng chạy trên một máy nếu SVN Subversion có nhiệm vụ theo vết lịch sử của dự án do các nhà phát triển phần mềm phát triển trong nội bộ.

- SVN Subversion hỗ trợ khá nhiều giao thức để kết nối giữa client và server. Ví dụ bạn có thể dùng các giao thức của ứng dụng web như http:// hoặc https://, hay các giao thức của svn như svn:// hoặc svn+ssh://, hoặc nếu phần mềm client và server cài chung trên 1 máy thì có thể dùng file://.

Việc cho phép server hỗ trợ giao thức nào phụ thuộc vào lúc cấu hình.

5.5 Case study: Cài đặt phần mềm

a) Mục tiêu

- Thực hành với công cụ GIT
- Lập trình giao diện đồ hoạ người dùng
- Xây dựng kiến trúc cho phần mềm

b) Yêu cầu

1. Thực hành với công cụ GIT
2. Lập trình giao diện đồ hoạ người dùng GUI
   - Làm quen với mô hình MVC các thành phần giao diện người dùng
   - Thiết kế giao diện GUI
   - Viết mã xử lý sự kiện
3. Xây dựng kiến trúc cho phần mềm
   - Sử dụng package tổ chức các class trong mã nguồn phần mềm
   - Thiết kế cơ sở dữ liệu và đặc tả thiết kế các bảng dữ liệu
   - Xây dựng các lớp model và các lớp truy xuất dữ liệu trong Database

c) Hướng dẫn, gợi ý: [33 – Chapter 4]

Câu hỏi

Câu 1. Cài đặt phần mềm là triển khai thiết kế chi tiết thành chương trình?

a) Đúng
b) Sai

Câu 2. Mã nguồn dễ bảo trì là?

a) Dễ hiểu
b) Dễ sửa lỗi
c) Dễ nâng cấp
d) Tất cả các phương án trên

Câu 3. Phương pháp lập trình hàm...

a) Phù hợp với các ứng dụng thời gian thực
b) Giống như phương pháp lập trình hướng thủ tục, vi dụ Pascal, C
c) Áp dụng chủ yếu trong lĩnh vực tính toán và trí tuệ nhân tạo
d) Áp dụng chủ yếu cho hệ thống thiết kế hướng chức năng

Câu 4. Phương pháp lập trình logic...

a) Chủ yếu được sử dụng để thực hiện các biểu thức logic: tìm kiếm giá trị các biến sao cho biểu thức logic có giá trị đúng
b) Khó lập trình
c) Được sử dụng trong cả lập trình hướng chức năng và hướng đối tượng
d) Chủ yêu được sử dụng để kiểm tra các biểu thức logic trong chương trình

Câu 5. Phong cách lập trình ...

a) Là phong cách của người lập trình
b) Là phong cách do tính cách và kinh nghiệm của lập trình viên tạo nên
c) Liên quan đến ngôn ngữ lập trình
d) Là các luật lập trình, các quy ước lập trình để đạt mục đích có 1 chương trình dễ hiểu, dễ kiểm thử, bảo trì và ít lỗi

Bài tập cuối chương

Bài 5.1. Hãy nêu ưu, nhược điểm của các phương pháp lập trình: Lập trình tuần tự, lập trình có cấu trúc, lập trình hướng hàm, lập trình hướng đối tượng

Bài 5.2. Khi chọn ngôn ngữ lập trình cho ứng dụng, cần quan tâm đến các yếu tố nào?

Bài 5.3. Việc chú thích các mô-đun là để làm rõ điều gì?

Bài 5.4. Trình bày một số công cụ và tổ chức, quản lý, chia sẻ SourceCode

Bài 5.5. Tìm hiểu và nêu các phương pháp gỡ lỗi phổ biến

Bài 5.6. Mục đích của "code review" là gì?

Bài 5.7. Thảo luận nhóm và xây dựng bộ quy ước viết mã + phong cách lập trình chung cho các thành viên trong nhóm

Bài 5.8. Việc lựa chọn ngôn ngữ lập trình cho ứng dụng có vai trò quan trọng trong phát triển phần mềm như thế nào?

Bài 5.9. Product Backlog là một danh sách chứa những thông tin cơ bản gì?
Bài 5.10. Trong quá trình phát triển phần mềm một mô hình kho lưu trữ (repository) được sử dụng để làm gì? Các yêu cầu về kho lưu trữ (repository) có giống những yêu cầu đối với cơ sở dữ liệu điển hình không?

Chương VI: KIỂM THỬ PHẦN MỀM

Nội dung chính của chương

6.1 Xác minh và thẩm định phần mềm

6.2 Tổng quan về kiểm thử phần mềm

6.3 Quy trình kiểm thử

6.4 Các mức kiểm thử phần mềm

6.5 Kỹ thuật kiểm thử phần mềm

Mục tiêu cần đạt được của chương

- Nắm được quy trình kiểm thử phần mềm
- Nắm được các kỹ thuật kiểm thử phần mềm, các mức kiểm thử phần mềm
- Có khả năng thiết kế các trường hợp kiểm thử và sử dụng các công cụ kiểm thử tự động

Bài 10: Kiểm thử phần mềm (Số tiết: 03 tiết)

6.1 Xác minh và thẩm định phần mềm (*Verification& Validation*)

Xác minh và thẩm định một hệ thống phần mềm là một quá trình liên tục xuyên suốt mọi giai đoạn của quá trình phần mềm. Xác minh và thẩm định là từ chung cho các quá trình kiểm thử để đảm bảo rằng phần mềm thỏa mãn các yêu cầu của chúng và các yêu cầu đó thỏa mãn các nhu cầu của người dùng phần mềm. Xác minh và thẩm định có hai mục tiêu:

- Phát hiện các khiếm khuyết trong hệ thống
- Đánh giá xem hệ thống liệu có dùng được hay không?

Sự khác nhau giữa xác minh và thẩm định là:

Xác minh (*Verification)* Xem xét cái được xây dựng có đúng là sản phẩm không? Như thế, xác minh là kiểm tra chương trình có phù hợp với đặc tả hay không?

Thẩm định (*Validation*) Xem xét cái được xây dựng có là sản phẩm đúng không? Tức là kiểm tra xem chương trình có được như mong đợi của người dùng hay không?

Trong thực tế, cần thực hiện xác minh (Verification) trước khi thực hiện thẩm định (Validation) sản phẩm phần mềm. Vì thế, chúng ta có thuật ngữ V&V (Verification & Validation). Lý do của việc này là chúng ta cần đảm bảo sản phẩm đúng với đặc tả trước. Nếu thực hiện việc thẩm định trước, một khi phát hiện ra lỗi, chúng ta không thể xác định được lỗi này do đặc tả sai hay do lập trình sai so với đặc tả.

6.2 Tổng quan về kiểm thử phần mềm

6.2.1 Kiểm thử phần mềm

Theo IEEE: Kiểm thử là tiến trình vận hành hệ thống hoặc thành phần dưới những điều kiện xác định, quan sát hoặc ghi nhận kết quả và đưa ra đánh giá về hệ thống hoặc thành phần đó.

Theo Myers (the art of software testing): Kiểm thử là tiến trình thực thi chương trình với mục đích tìm thấy lỗi.

Mục đích của kiểm thử phầm mềm là tìm lỗi. Mục đích của gỡ rối là xác định bản chất lỗi và định vị lỗi trong chương trình và tiến hành sửa lỗi.

6.2.2 Thế nào là lỗi phần mềm?

Có rất nhiều định nghĩa khác nhau về lỗi phần mềm, nhưng tựu chung, có thể phát biểu một cách tổng quát: *"Lỗi phần mềm là sự không khớp giữa chương trình và đặc tả của nó"*.

Dựa vào định nghĩa, chúng ta có thể thấy lỗi phần mềm xuất hiện theo ba dạng sau:

Sai: Sản phẩm được xây dựng khác với đặc tả.

Thiếu: Một yêu cầu đã được đặc tả nhưng lại không có trong sản phẩm được xây dựng.

Thừa: Một yêu cầu được đưa vào sản phẩm mà không có trong đặc tả. Cũng có trường hợp yêu cầu này có thể là một thuộc tính sẽ được người dùng chấp nhận nhưng khác với đặc tả nên vẫn coi là có lỗi.

Một hình thức khác nữa cũng được xem là lỗi, đó là phần mềm khó hiểu, khó sử dụng, chậm hoặc dễ gây cảm nhận rằng phần mềm hoạt động không đúng.

Phân biệt một số khái niệm: error, fault, failure:

- *Error (mistake – sai sót):* là những vấn đề mà người thực hiện đã mắc phải trong quá trình phát triển các sản phẩm phần mềm dẫn đến kết quả sai trở thành nguyên nhân gây ra lỗi
  Ví dụ: Developer đặt tên biến cho 1 function sai cú pháp, dẫn đến khi gọi biến này thì không ra kết quả.
- *Fault (lỗi):* xuất hiện trong phần mềm như là kết quả của một sai sót hay nó là nguyên nhân gây ra lỗi của chương trình
- *Failure (hỏng hóc):* nó là kết quả của một lỗi xuất hiện làm cho chương trình không hoạt động được hay hoạt động nhưng cho kết quả không như mong đợi.

Bug: Là một khiếm khuyết trong một thành phần hoặc hệ thống mà nó có thể làm cho thành phần hoặc hệ thống này không thực hiện đúng chức năng yêu cầu của nó, ví

dụ như thông báo sai hoặc định nghĩa dữ liệu không đúng. Một bug, nếu gặp phải trong quá trình hệ thống hoạt động, có thể gây ra failure trong thành phần hoặc hệ thống đó.

Lỗi phần mềm xảy ra khi nào? Ở đâu?

Lỗi xuất hiện ở mọi nơi trong quy trình phát triển phần mềm: xen lẫn trong quá trình Requirement Definition →Logial-Physical Design →Program Development

Nguyên nhân gây ra lỗi:

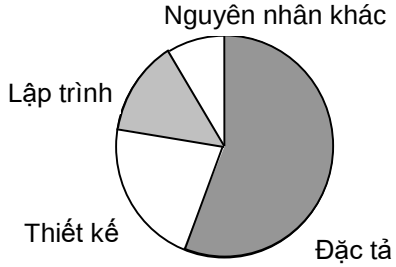
+ Hiểu sai requirement

+ Thiếu cân nhắc specification

+ Design không chính xác

+ Implement không tốt

Khác với sự cảm nhận thông thường, lỗi xuất hiện nhiều nhất không phải do lập trình. Nhiều nghiên cứu đã được thực hiện trong các dự án từ rất nhỏ đến các dự án rất lớn và kết quả luôn giống nhau. Số lỗi do đặc tả gây ra là nhiều nhất, chiếm khoảng 80%. Có một số nguyên nhân làm cho đặc tả tạo ra nhiều lỗi nhất. Trong nhiều trường hợp, đặc tả không được viết ra. Các nguyên nhân khác có thể do đặc tả không đủ cẩn thận, nó hay thay đổi, hoặc do chưa phối hợp tốt trong toàn nhóm phát triển. Sự thay đổi yêu cầu của khách hàng cũng là nguyên nhân dễ gây ra lỗi phần mềm. Khách hàng thay đổi yêu cầu không cần quan tâm đến những tác động sau khi thay đổi yêu cầu như phải thiết kế lại, lập lại kế hoạch, làm lại những việc đã hoàn thành. Nếu có nhiều sự thay đổi, rất khó nhận biết hết được phần nào của dự án phụ thuộc và phần nào không phụ thuộc vào sự thay đổi. Nếu không giữ được vết thay đổi rất dễ phát sinh ra lỗi.



Hình 6.1: Các nguyên nhân gây ra lỗi phần mềm

Ví dụ: Trong đặc tả bài toán phân số:

Với việc đặc tả phi hình thức: Phân số là một cặp t/m, trong đó t là một số nguyên, m là một số tự nhiên lớn hơn 0; t được gọi là tử số, m được gọi là mẫu số của phân số.

Đặc tả hình thức là đặc tả trong đó sử dụng các ký hiệu toán học để mô tả. Một phân số có thể được đặc tả như sau:

+ Phân số = $\{(t,m) \mid t \in Z, m \in N^+\}$ (*)

Trong đó: $N = \{0, 1, 2, 3, \ldots\}$

$N^+ = \{1, 2, 3, \ldots\}$

$$Z = \{0, \pm1, \pm2, \pm3, \ldots\}$$

+ Phép chia hai phân số: (t1,m1):(t2,m2) = Reduce(t1× m2, t2 × m1), trong đó Reduce(t, m) = (t/d, m/d) với d = gcd(t, m). Hàm gcd là hàm tìm ước số chung lớn nhất của hai số tự nhiên.

Đặc tả phép chia như trên là sai với đặc tả phân số (*). Chẳng hạn, thực hiện chia hai phân số (1,3):(-2,5) = (5,-6), thì mẫu số trong trường hợp này lại là một số âm, không đúng đặc tả.Đây là một ví dụ đơn giản về việc đặc tả sai do không đủ cẩn thận. Với các bài toán lớn thì việc đặc tả sẽ rất khó và dễ nhầm lẫn, sai sót.

Nguồn gây ra lỗi lớn thứ hai là thiết kế. Đó là nền tảng mà lập trình viên dựa vào để nỗ lực thực hiện kế hoạch cho phần mềm.

Lỗi do lập trình gây ra cũng khá dễ hiểu. Ai cũng có thể mắc lỗi khi lập trình. Thời kì đầu, phát triển phần mềm có nghĩa là lập trình, công việc lập trình thì nặng nhọc, do đó lỗi do lập trình gây ra là chủ yếu. Ngày nay, công việc lập trình chỉ là một phần việc của quá trình phát triển phần mềm, cộng với sự hỗ trợ của nhiều công cụ lập trình cao cấp, việc lập trình trở nên nhẹ nhàng hơn, mặc dù độ phức tạp phần mềm lớn hơn rất nhiều. Do đó, lỗi do lập trình gây ra cũng ít hơn. Tuy nhiên, nguyên nhân để lập trình tạo ra lỗi lại nhiều hơn. Dó là do độ phức tạp của phần mềm, do tài liệu nghèo nàn, do sức ép thời gian hoặc chỉ đơn giản là những lỗi "không nói lên được". Một điều cũng hiển nhiên là nhiều lỗi xuất hiện trên bề mặt lập trình nhưng thực ra lại do lỗi của đặc tả hoặc thiết kế.

Một nguyên nhân khác tạo ra lỗi là do bản thân các công cụ phát triển phần mềm cũng có lỗi như công cụ trực quan, thư viện lớp, bộ biên dịch,…

6.2.3 Một số khái niệm liên quan

- ✓ **Dữ liệu thử (test data):** là các dữ liệu đầu vào cung cấp cho chương trình trong khi thực thi
- ✓ **Kịch bản kiểm thử (test scenario)**: là các bước thực hiện khi tiến hành kiểm thử
- ✓ **Phán xét kiểm thử (test oracle)**: là hoạt động nhằm đánh giá kết quả kiểm thử. Hoạt động này có thể tiến hành tự động bằng chương trình máy tính hoặc tiến hành thủ công bởi con người.
- ✓ **Kiểm thử viên (tester)**: là người tiến hành hoạt động kiểm thử
- ✓ **Ca kiểm thử (test case)**: Một ca kiểm thử bao gồm:
    - Tập dữ liệu thử
    - Điều kiện thực thi
    - Kết quả mong đợi

6.2.4 Vai trò của kiểm thử phần mềm

Kiểm thử phần mềm đóng vai trò quan trọng trong việc đánh giá và thu được chất lượng cao của sản phẩm phần mềm trong quá trình phát triển. Thông qua chu trình "*kiểm thử - tìm lỗi - sửa lỗi*", ta hy vọng chất lượng của sản phẩm phần mềm sẽ được cải tiến. Mặt khác, thông qua việc tiến hành kiểm thử mức hệ thống trước khi cho lưu hành sản phẩm, ta biết được sản phẩm của ta tốt ở mức nào. Vì thế, nhiều tác giả đã mô tả việc kiểm thử phần mềm là một quy trình kiểm chứng để đánh giá và tăng cường chất lượng của sản phẩm phần mềm. Quy trình này gồm hai công việc chính là phân tích tĩnh và phân tích động.

Phân tích tĩnh: Việc phân tích tĩnh được tiến hành dựa trên việc khảo sát các tài liệu được xây dựng trong quá trình phát triển sản phẩm như tài liệu đặc tả nhu cầu người dùng, mô hình phần mềm, hồ sơ thiết kế và mã nguồn phần mềm. Các phương pháp phân tích tĩnh truyền thống bao gồm việc khảo sát đặc tả và mã nguồn cùng các tài liệu thiết kế. Người ta cũng có thể dùng các kỹ thuật phân tích hình thức như kiểm chứng mô hình (model checking) và chứng minh định lý (theorem proving) để chứng minh tính đúng đắn của thiết kế và mã nguồn. Công việc này không động đến việc thực thi chương trình mà chỉ duyệt, lý giải về tất cả các hành vi có thể của chương trình khi được thực thi. Tối ưu hóa các chương trình dịch là các ví dụ về phân tích tĩnh.

Phân tích động: Phân tích động liên quan đến việc thực thi chương trình để phát hiện những thất bại có thể có của chương trình, hoặc quan sát các tính chất nào đó về hành vi và hiệu quả (performance). Vì gần như không thể thực thi chương trình trên tất cả các dữ liệu đầu vào có thể, ta chỉ có thể chọn một tập con các dữ liệu đầu vào để thực thi, gọi là các "ca kiểm thử". Chọn như thế nào để được các bộ dữ liệu đầu vào hiệu quả (tức là các bộ dữ liệu có xác suất phát hiện thất bại (nếu có) cao hơn là công việc cần suy nghĩ và là nội dung chính của chường này).

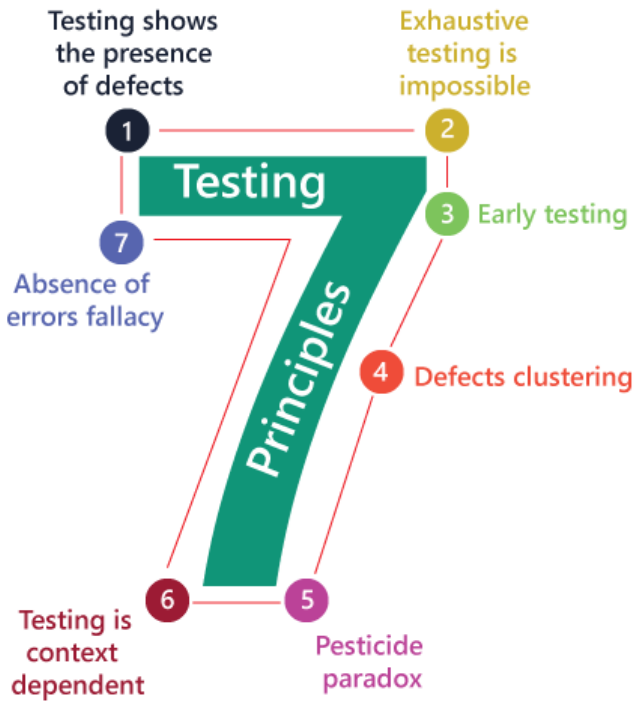
Bằng việc phân tích tĩnh và động, người kiểm thử muốn phát hiện nhiều lỗi nhất có thể được để chúng có thể được sửa ở giai đoạn sớm nhất trong quá trình phát triển phần mềm. Phân tích tĩnh và động là hai kỹ thuật bổ sung cho nhau và cần được làm lặp đi lặp lại nhiều trong quá trình kiểm thử.

6.2.5 Những nguyên tắc trong kiểm thử phần mềm

Bảy nguyên tắc kiểm thử phần mềm bao gồm [20]:

1. Kiểm thử chứng minh sự hiện diện của lỗi
2. Kiểm thử toàn bộ là không khả thi
3. Kiểm thử càng sớm càng tốt
4. Lỗi thường phân bổ tập trung

5. Nghịch lý thuốc trừ sâu
6. Kiểm thử phụ thuộc vào ngữ cảnh
7. Quan niệm sai lầm về việc "hết lỗi"



Hình 6.2: Bảy nguyên tắc kiểm thử phần mềm

1. Kiểm thử phần mềm chứng minh sự hiện diện của lỗi

Bằng việc kiểm thử, chúng ta có thể làm giảm lượng bugs khi áp dụng nhiều phương pháp kiểm thử lên phần mềm. Tuy nhiên khi được đưa lên môi trường thật, người dùng cuối hoàn toàn có thể thấy nhiều lỗi khác không tìm thấy trong quá trình kiểm thử. Kiểm thử chứng minh được sản phẩm có lỗi nhưng không thể chứng minh rằng sản phẩm không còn lỗi. Điều này có nghĩa là, sẽ luôn có lỗi không được phát hiện trong phần mềm, ngay cả khi không tìm thấy lỗi, cũng không đồng nghĩa rằng phần mềm đúng hoàn toàn.

2. Kiểm thử toàn bộ là không khả thi

Rất khó để kiểm tra toàn bộ các module cũng như các tính năng, kết hợp với đầu vào và đầu ra trong suốt quá trình kiểm tra. Các sản phẩm phần mềm hiện nay cực kỳ đa dạng và phức tạp, được phát triển trên nhiều nền tảng, thêm vào đó, ngày càng có nhiều công nghệ mới, khả năng kết nối dữ liệu lớn… khiến việc kiểm thử toàn bộ gần như là không thể. Thay vì cố gắng kiểm thử toàn bộ, hãy xác định mức độ quan trọng và độ ưu tiên của các module để kiểm thử những phần cần thiết hoặc gặp nhiều nguy cơ hơn.

3. Kiểm thử càng sớm càng tốt

Nguyên tắc kiểm thử sớm có nghĩa là việc kiểm thử cần được thực hiện càng sớm càng tốt trong vòng đời phát triển phần mềm. Vậy ở bước nào thì được coi là sớm? Nguyên tắc này cho thấy ta cần phát hiện bug ngay từ các bước đầu tiên như nghiên cứu yêu cầu (requirement) hay design. Phát hiện lỗi càng muộn, chi phí bỏ ra để xử lý càng cao. Vì vậy, việc thay đổi yêu cầu không đúng từ sớm sẽ khiến tốn ít chi phí để thay đổi tính năng hơn.

4. Lỗi thường được phân bố tập trung

Nguyên lý về phân bố lỗi chỉ ra rằng, chỉ một số ít module chứa phần lớn số lỗi phát hiện được. Những module này thường là những thành phần, chức năng chính của hệ thống. Điều này cũng đúng theo nguyên lý Pareto: 80 – 20: 80% số lỗi tìm thấy ở chỉ 20% module. Bằng kinh nghiệm, các QA/ Tester có thể xác định được những module có tính rủi ro và nhiều lỗi như vậy, giúp tập trung tìm kiếm lỗi nhanh và hiệu quả hơn. Tuy nhiên, cách tiếp cận này cũng ẩn chứa vấn đề: nếu thực hiện kiểm thử tương tự lặp đi lặp lại, dễ thấy rằng những test case cũ sẽ khó tìm thêm được bug mới.

5. Nghịch lý thuốc trừ sâu

Trong trồng trọt, nếu người nông dân sử dụng lặp lại một liều trừ sâu, các loại sâu bệnh sẽ dần dần thích nghi và trở nên "nhờn" với loại thuốc trừ sâu đó. Tương tự với kiểm thử phần mềm, khi lặp đi lặp lại một test case, thì xác suất tìm được lỗi là rất thấp. Nguyên nhân là hệ thống hoàn thiện hơn, lỗi tìm thấy đã được sửa theo test case cũ. Để xử lý hiệu ứng "thuốc trừ sâu" này, test case cần được thường xuyên xem lại và chỉnh sửa, thêm nhiều test case mới để tìm lỗi mới (regression test).

Thêm vào đó, QA/ Tester cũng không nên phụ thuộc quá nhiều vào các kỹ thuật test sẵn có. Bạn cần liên tục cải tiến các phương pháp có sẵn để làm việc kiểm thử hiệu quả hơn.

6. Kiểm thử phụ thuộc vào ngữ cảnh

Kiểm thử phụ thuộc vào ngữ cảnh, nói một cách đơn giản, là việc kiểm thử một trang thương mại điện tử sẽ phải khác cách test một ứng dụng đọc tin tức. Tất cả các phần mềm đều được phát triển theo cách khác nhau. Và việc áp dụng chung một "cách giải" là sai lầm. Bạn cần sử dụng cách tiếp cận khác nhau, phương thức, kỹ thuật test khác nhau, loại test phụ thuộc vào loại phần mềm/ ứng dụng/ website.

7. Quan niệm sai lầm về việc "hết lỗi"

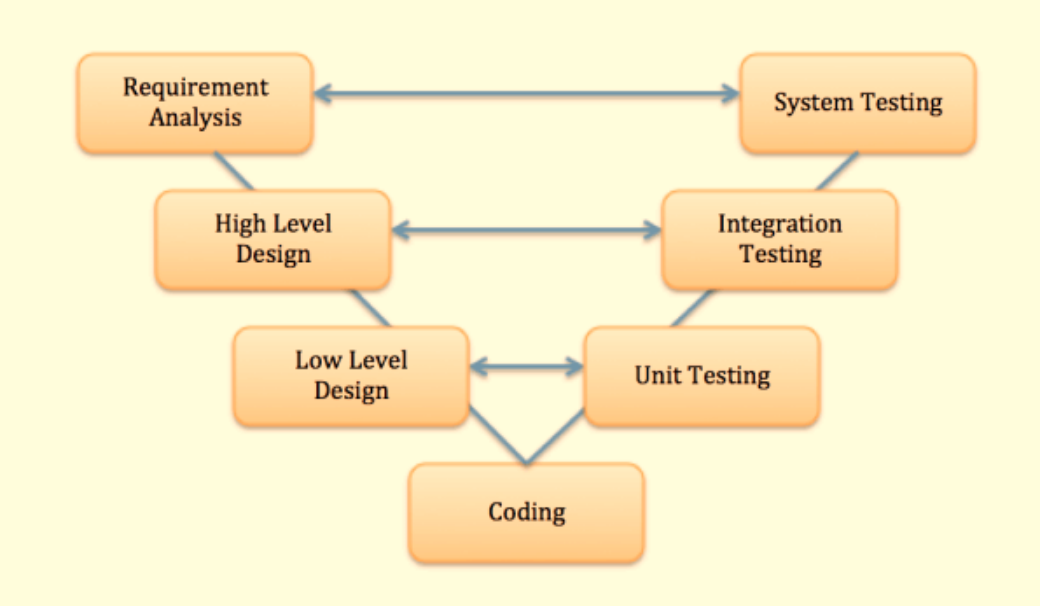
Một phần mềm sạch bug 99% vẫn có thể không sử dụng được. Đây là trường hợp phần mềm được kiểm thử bằng một requirement sai. Kiểm thử không chỉ để tìm ra lỗi, mà còn để kiểm tra xem phần mềm có đáp ứng được đúng nhu cầu hay không. Chính vì vậy, việc không còn lỗi hay hết lỗi là quan niệm sai lầm.

Quan điểm: "Nguyên tắc kiểm thử chỉ là để tham khảo, không có tính ứng dụng thực tế"? Đây là quan điểm cực kỳ sai lầm. Các nguyên tắc kiểm thử giúp tạo ra một chiến lược kiểm thử rõ ràng và tạo ra những trường hợp kiểm thử sát sao, dễ bắt được bug. Những tester dày dạn kinh nghiệm áp dụng những nguyên tắc kiểm thử nhuần nhuyễn đến độ họ không nghĩ rằng họ đang áp dụng chúng. Vì vậy, việc nguyên tắc kiểm thử không có tính ứng dụng thực tế là sai lầm.

6.2.6 V-Model trong kiểm thử phần mềm

Trong kiểm thử phần mềm, V-Model (hay mô hình chữ V) là một mô hình dạng SDLC (Software Development Life Cycle) có tính kỷ luật cao, trong đó có một giai đoạn kiểm thử chạy song song với mỗi giai đoạn của phát triển. Mô hình chữ V là một phần mở rộng của mô hình thác nước (Waterfall), trong đó việc kiểm thử được thực hiện trên từng giai đoạn song song với việc phát triển một cách tuần tự. Nó còn được biết đến với tên gọi Validation Model (mô hình xác thực) hoặc Verification Model (mô hình xác minh).

Mô hình chữ V được tạo ra như một giải pháp để giải quyết vấn đề của mô hình thác nước. Thay vì chỉ kiểm thử khi quá trình phát triển mã nguồn kết thúc như trong mô hình thác nước, mô hình chữ V cung cấp một quá trình kiểm thử chạy song song cho mỗi bước của quá trình phát triển.



Hình 6.3: V-Model

Mô hình chữ V thực chất là tổ hợp của vòng đời phát triển phần mềm **SDLC** ở bên trái và vòng đời kiểm thử phần mềm **STLC** ở bên phải.

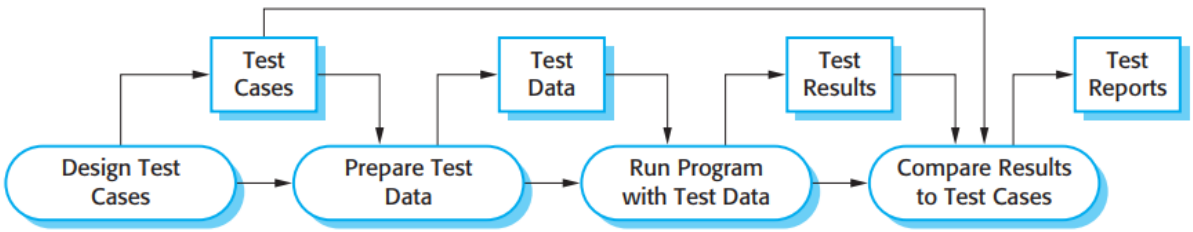
- Requirement Analysis (Phân tích yêu cầu) sẽ có quá trình tương ứng là System Testing (Kiểm thử hệ thống): Ở bước này sẽ kiểm tra tổng quan toàn bộ hệ thống.

- High Level Design (Thiết kế cấp cao) sẽ có quá trình tương ứng là Integration Testing (Kiểm thử tích hợp): Ở bước này sẽ kiểm tra sự kết nối và tương hợp giữa các thành phần của phần mềm.
- Low Level Design (Thiết kế cấp thấp) sẽ có quá trình tương ứng là Unit Testing (Kiểm thử đơn vị): Ở bước này sẽ kiểm tra ở mức function (tính năng) của phần mềm.
- Coding không cần một quá trình kiểm thử tương ứng, thật ra là không cần thiết do ở bước này hầu như các công nghệ và nền tảng kỹ thuật đã hoàn toàn được kiểm tra trước đó bởi nhà sản xuất của mỗi hãng trước khi được sử dụng chính thức. Vì vậy tester không phải kiểm tra lại ở bước này, thường sẽ do Dev đảm bảo.

6.3 Quy trình kiểm thử

Kiểm thử thường bao gồm các bước:

1. Lập kế hoạch kiểm thử, xây dựng kịch bản kiểm thử.
2. Tạo dữ liệu thử (Thiết kế các ca kiểm thử):
   - Kiểm thử với tất cả các dữ liệu vào là cần thiết
   - Chọn tập các dữ liệu thử đại diện từ miền dữ liệu vào (dựa trên các tiêu chuẩn chọn dữ liệu thử).
3. Thực thi chương trình trên dữ liệu thử:
   - Cung cấp dữ liệu thử
   - Thực thi
   - Ghi nhận kết quả
4. Quan sát kết quả kiểm thử
   - Thực hiện trong khi hoặc sau khi thực thi
   - So sánh kết quả nhận được và kết quả mong đợi



Hình 6.4: Quy trình kiểm thử phần mềm

Ví dụ: Cho 1 chương trình tìm số lớn nhất trong 1 dãy số a gồm n số.

* Thiết kế ca kiểm thử

- Các đầu vào (Input)

  N: Integer;

  A={a[i] | a[i] thuộc Z; i=1->n}

- Điều kiện thử (conditions)

- Các đầu ra (outputs): max = a[i] | a[i]>=a[j]; i=1 -> n; j= 1 ->n

* Chuẩn bị dữ liệu thử

Td1={n=-5; a={};}

Td2={n=5; a={7,8,3,6,25};

* Chạy chương trình

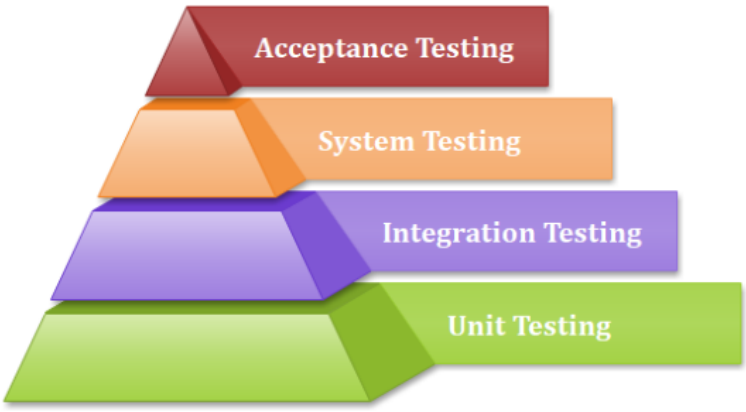
Max = ; (kết quả thực sự mong đợi chạy với td2);

* Phán xét: -> Chương trình chạy sai

-> Chương trình chạy đúng

6.4 Các mức kiểm thử phần mềm

Có 04 mức độ kiểm thử phần mềm chính cần hoàn thành trước khi đưa phần mềm vào sử dụng: unit testing, integration testing, system testing, và acceptance testing.



Hình 6.5: Các mức kiểm thử phần mềm

Nhiều bạn sẽ đặt câu hỏi tại sao không bao gồm Regression Tesing (Kiểm thử hồi quy)? Kiểm thử hồi quy không phải là một cấp độ kiểm thử riêng biệt; nó chỉ là một loại kiểm thử có thể được thực hiện trong bất kỳ giai đoạn nào trong bốn mức độ kiểm thử phần mềm chính ở trên.

6.4.1 Unit Testing (Kiểm thử đơn vị) [21][22][23]

Unit Testing là giai đoạn đầu tiên trong kiểm thử phần mềm. Unit Testing là một loại kiểm thử phần mềm trong đó các đơn vị hay thành phần riêng lẻ của phần mềm được kiểm thử nhằm kiểm tra và xác định các đơn vị riêng lẻ thuộc mã nguồn có hoạt động đúng hay không.

Mục đích của kiểm thử mức đơn vị như sau:

- Xác định mỗi đơn vị phần mềm có đang thực hiện theo đúng thiết kế ban đầu hay không.

- Thông qua thử nghiệm sẽ giúp khắc phục những phát sinh do việc thay đổi hay bảo trì code.

- Unit Testing giúp tiết kiệm chi phí, thời gian và thể diện khi phát hiện ra lỗi.

6.4.1.1 Unit là gì?

Một Unit là một thành phần PM nhỏ nhất mà ta có thể kiểm tra được như các hàm (Function), thủ tục (Procedure), lớp (Class), hoặc các phương thức (Method). Vì Unit được chọn để kiểm tra thường có kích thước nhỏ và chức năng hoạt động đơn giản, chúng ta không khó khăn gì trong việc tổ chức, kiểm tra, ghi nhận và phân tích kết quả kiểm tra nên việc phát hiện lỗi sẽ dễ dàng xác định nguyên nhân và khắc phục cũng tương đối dễ dàng vì chỉ khoanh vùng trong một Unit đang kiểm tra.

Mỗi UT sẽ gửi đi một thông điệp và kiểm tra câu trả lời nhận được đúng hay không, bao gồm:

- Các kết quả trả về mong muốn

- Các lỗi ngoại lệ mong muốn

Các đoạn mã UT hoạt động liên tục hoặc định kỳ để thăm dò và phát hiện các lỗi kỹ thuật trong suốt quá trình phát triển, do đó UT còn được gọi là kỹ thuật kiểm nghiệm tự động. UT có các đặc điểm sau:

- Đóng vai trò như những người sử dụng đầu tiên của hệ thống.

- Chỉ có giá trị khi chúng có thể phát hiện các vấn đề tiềm ẩn hoặc lỗi kỹ thuật.

Một số khái niệm khi thực hiện Unit test:

- ✓ **Assertion**: Là một phát biểu mô tả các công việc kiểm tra cần tiến hành, thí dụ: AreEqual(), IsTrue(), IsNotNull()… Mỗi một UT gồm nhiều assertion kiểm tra dữ liệu đầu ra, tính chính xác của các lỗi ngoại lệ ra và các vấn đề phức tạp khác như: – Sự tồn tại của một đối tượng – Điều kiện biên: Các giá trị có vượt ra ngoài giới hạn hay không – Thứ tự thực hiện của các luồng dữ liệu …
- ✓ **Test Point**: Là một đơn vị kiểm tra nhỏ nhất, chỉ chứa đơn giản một assertion nhằm khẳng định tính đúng đắn của một chi tiết mã nào đó. Mọi thành viên dự án đều có thể viết một test point. Test Case: Là một tập hợp các test point nhằm kiểm tra một đặc điểm chức năng cụ thể, thí dụ toàn bộ giai đoạn người dùng nhập dữ liệu cho đến khi thông tin được nhập vào cơ sở dữ liệu. Trong nhiều trường hợp kiểm tra đặc biệt và khẩn cấp có thể không cần đến test case.
- ✓ **Test Suite**: Là một tập hợp các test case định nghĩa cho từng module hoặc hệ thống con.
- ✓ **Regression Testing (hoặc Automated Testing)**: Là phương pháp kiểm nghiệm tự động sử dụng một phần mềm đặc biệt. Cùng một loại dữ liệu kiểm tra giống nhau nhưng được tiến hành nhiều lần lặp lại tự động nhằm ngăn chặn các lỗi cũ phát sinh trở lại. Kết hợp Regression Testing với Unit Testing sẽ đảm bảo các

đoạn mã mới vẫn đáp ứng yêu cầu thay đổi và các đoạn mã cũ sẽ không bị ảnh hưởng bởi các hoạt động bảo trì.

- ✓ **Production Code**: Phần mã chính của ứng dụng được chuyển giao cho khách hàng.
- ✓ **Unit Testing Code**: Phần mã phụ để kiểm tra mã ứng dụng chính, không được chuyển giao cho khách hàng.

6.4.1.2 Vòng đời Unit Test

UT có 3 trạng thái cơ bản:

- Fail (trạng thái lỗi)

- Ignore (tạm ngừng thực hiện)

- Pass (trạng thái làm việc)

Toàn bộ UT được vận hành trong một hệ thống tách biệt. Có rất nhiều phần mềm hỗ trợ thực thi UT với giao diện trực quan. Thông thường, trạng thái của UT được biểu hiện bằng các màu khác nhau: màu xanh (pass), màu vàng (ignore) và màu đỏ (fail)

UT chỉ thực sự đem lại hiệu quả khi:

- Được vận hành lặp lại nhiều lần

- Tự động hoàn toàn

- Độc lập với các UT khác.

6.4.1.3 Thiết kế Unit test

Mỗi UT đều được thiết kế theo trình tự sau:

- Thiết lập các điều kiện cần thiết: khởi tạo các đối tượng, xác định tài nguyên cần thiết, xây dựng các dữ liệu giả…

- Triệu gọi các phương thức cần kiểm tra.

- Kiểm tra sự hoạt động đúng đắn của các phương thức.

- Dọn dẹp tài nguyên sau khi kết thúc kiểm tra.

6.4.1.4 Ứng dụng Unit test

- Kiểm tra mọi đơn vị nhỏ nhất là các thuộc tính, sự kiện, thủ tục và hàm.

- Kiểm tra các trạng thái và ràng buộc của đối tượng ở các mức sâu hơn mà thông thường chúng ta không thể truy cập được.

- Kiểm tra các quy trình (process) và mở rộng hơn là các khung làm việc (workflow – tập hợp của nhiều quy trình)

Lợi ích của việc áp dụng Unit test:

Thời gian đầu, người ta thường do dự khi phải viết UT thay vì tập trung vào code cho các chức năng nghiệp vụ. Công việc viết Unit Test có thể mất nhiều thời gian hơn code rất nhiều nhưng lại có lợi ích sau:

- ✓ Tạo ra môi trường lý tưởng để kiểm tra bất kỳ đoạn code nào, có khả năng thăm dò và phát hiện lỗi chính xác, duy trì sự ổn định của toàn bộ PM và giúp tiết kiệm thời gian so với công việc gỡ rối truyền thống.
- ✓ Phát hiện các thuật toán thực thi không hiệu quả, các thủ tục chạy vượt quá giới hạn thời gian.
- ✓ Phát hiện các vấn đề về thiết kế, xử lý hệ thống, thậm chí các mô hình thiết kế.
- ✓ Phát hiện các lỗi nghiêm trọng có thể xảy ra trong những tình huống rất hẹp.
- ✓ Tạo hàng rào an toàn cho các khối mã: Bất kỳ sự thay đổi nào cũng có thể tác động đến hàng rào này và thông báo những nguy hiểm tiềm tàng.

Trong môi trường làm việc Unit Test còn có tác dụng rất lớn đến năng suất làm việc:

- ✓ Giải phóng chuyên viên QA khỏi các công việc kiểm tra phức tạp.
- ✓ Tăng sự tự tin khi hoàn thành một công việc. Chúng ta thường có cảm giác không chắc chắn về các đoạn mã của mình như liệu các lỗi có quay lại không, hoạt động của module hiện hành có bị tác động không, hoặc liệu công việc hiệu chỉnh mã có gây hư hỏng đâu đó…
- ✓ Là công cụ đánh giá năng lực của bạn. Số lượng các tình huống kiểm tra (test case) chuyển trạng thái "pass" sẽ thể hiện tốc độ làm việc, năng suất của bạn.

6.4.1.5 Cách code hiệu quả với Unit Test

Phân tích các tình huống có thể xảy ra đối với mã. Đừng bỏ qua các tình huống tồi tệ nhất có thể xảy ra, thí dụ dữ liệu nhập làm một kết nối cơ sở dữ liệu thất bại, ứng dụng bị treo vì một phép toán chia cho không, các thủ tục đưa ra lỗi ngoại lệ sai có thể phá hỏng ứng dụng một cách bí ẩn…

Mọi UT phải bắt đầu với trạng thái "fail" và chuyển trạng thái "pass" sau một số thay đổi hợp lý đối với mã chính.

Mỗi khi viết một đoạn mã quan trọng, hãy viết các UT tương ứng cho đến khi bạn không thể nghĩ thêm tình huống nào nữa.

Nhập một số lượng đủ lớn các giá trị đầu vào để phát hiện điểm yếu của mã theo nguyên tắc:

- ✓ Nếu nhập giá trị đầu vào hợp lệ thì kết quả trả về cũng phải hợp lệ
- ✓ Nếu nhập giá trị đầu vào không hợp lệ thì kết quả trả về phải không hợp lệ

- ✓ Sớm nhận biết các đoạn mã không ổn định và có nguy cơ gây lỗi cao, viết UT tương ứng để khống chế.

Ứng với mỗi đối tượng nghiệp vụ (business object) hoặc đối tượng truy cập dữ liệu (data access object), nên tạo ra một lớp kiểm tra riêng vì những lỗi nghiêm trọng có thể phát sinh từ các đối tượng này.

Để ngăn chặn các lỗi có thể phát sinh trở lại thực thi tự động tất cả UT mỗi khi có một sự thay đổi quan trọng, hãy làm công việc này mỗi ngày. Các UT lỗi cho chúng ta biết thay đổi nào là nguyên nhân gây lỗi.

Để tăng hiệu quả và giảm rủi ro khi viết các UT, cần sử dụng nhiều phương thức kiểm tra khác nhau. Hãy viết càng đơn giản càng tốt.

Cuối cùng, viết Unit Test cũng đòi hỏi sự nỗ lực, kinh nghiệm và sự sáng tạo như viết phần mềm:

- ✓ Chắc chắn rằng mỗi test case kiểm thử mức đơn vị sẽ độc lập với những test case khác. Không nên gọi một test case khác trong một test case. Test case không nên phụ thuộc vào nhau cả về data và thứ tự thực hiện.
- ✓ Luôn luôn kiểm tra từng mô-đun một cách độc lập. Nếu không, sẽ có nhiều sự chồng chéo giữa các ca thử nghiệm và việc thay đổi đối với một đơn vị có thể ảnh hưởng đến tất cả các mô-đun khác và khiến phần mềm bị lỗi.
- ✓ Đặt tên các đơn vị kiểm thử một cách rõ ràng và nhất quán. Đảm bảo rằng các test case dễ đọc, bất kỳ ai cũng có thể chọn test case và chạy nó mà không gặp bất kỳ vấn đề nào.
- ✓ Khi triển khai việc thay đổi giao diện hoặc chức năng, cần chạy lại các test case trước đó nhằm đảm bảo việc thay đổi này không làm ảnh hưởng đến những test case cũ đã pass.
- ✓ Luôn đảm bảo lỗi được xác định trong quá trình Unit test được sửa trước khi chuyển sang giai đoạn tiếp theo.
- ✓ Không cố gắng viết test case để kiểm thử tất cả mọi thứ, thay vào đó nên tập chung vào kiểm thử sự ảnh hưởng của hành vi hệ thống
- ✓ Bên cạnh viết test case để test hành vi hệ thống, cần viết thêm test case để kiểm thử hiệu năng của mã nguồn
- ✓ Các test suite nên đặt riêng ra, độc lập code với module
- ✓ Không nên có nhiều assert trong một test case vì khi một điều kiện không thỏa mãn thì các assert khác sẽ bị bỏ qua

- ✓ Sau một thời gian dài, số lượng test case nhiều, thời gian chạy lớn. Nên chia ra nhóm test case cũ và test case mới, test case cũ sẽ chạy với tần suất ít hơn

UT chỉ thực sự mang lại lợi ích nếu chúng ta đặt vấn đề chất lượng phần mềm lên hàng đầu hơn là chỉ nhằm kết thúc công việc đúng thời hạn. Khi đã thành thạo với công việc viết UT, bạn có thể đọc thêm về các kỹ thuật xây dựng UT phức tạp hơn, trong số đó có mô hình đối tượng ảo.

Thế nào được gọi là Unit test tốt?

- ✓ Chạy nhanh
- ✓ Chạy độc lập giữa các test case, không phụ thuộc vào thứ tự kiểm thử
- ✓ Sử dụng data dễ đọc, dễ hiểu
- ✓ Sử dụng dữ liệu thực tế có thể
- ✓ Test case đơn giản, dễ đọc, dễ bảo trì
- ✓ Phản ảnh đúng hoạt động của module

6.4.2 Integration Testing (Kiểm thử tích hợp)

Mỗi dự án phần mềm được hoàn thành bởi rất nhiều module do nhiều người code khác nhau. Integration Testing là cấp độ kiểm thử phần mềm tích hợp của các đơn vị riêng lẻ được kết hợp và thử nghiệm thành một nhóm thông qua việc tập trung vào kiểm tra truyền dữ liệu giữa các module.

Mục đích của giai đoạn kiểm thử Integration Testing đó là tìm và phát hiện lỗi khi tích hợp các module lại với nhau. Để tiến hành Integration Testing có 4 phương pháp tiếp cận chính bao gồm: big bang, top down, bottom up và sandwich/hybrid.

6.4.2.1 Integration Testing là gì?

Integration Testing (Kiểm thử tích hợp) là một loại kiểm thử trong đó các module của phần mềm được tích hợp logic và được kiểm thử theo nhóm. Một dự án phần mềm điển hình bao gồm nhiều module, được code bởi các lập trình viên. Kiểm thử tích hợp là kiểm thử sự tương thích giữa các module đó. Do đó, kiểm thử tích hợp còn được gọi là I & T (Tích hợp và Kiểm thử), String Testing (Kiểm thử chuỗi) và đôi khi là Thread Testing (Kiểm thử luồng).

Tại sao cần phải Integration Testing?

Mặc dù mỗi module đã được Unit Testing nhưng lỗi vẫn còn tồn tại vì một số lý do như:

- Do mỗi module được thiết kế bởi một developer độc lập, có kiến thức và logic lập trình khác nhau vì vậy có thể sẽ có lỗi phát sinh khi tích hợp các module với nhau.

- Khách hàng sẽ thay đổi yêu cầu thiết kế trong quá trình phát triển module (thêm yêu cầu, update lại yêu cầu khi thấy không hợp lý…) và các yêu cầu mới này có thể không được Unit Testing hay khi tích hợp sẽ phát sinh lỗi.

- Các giao diện của các module trong phần mềm với cơ sở dữ liệu có thể không tương thích.

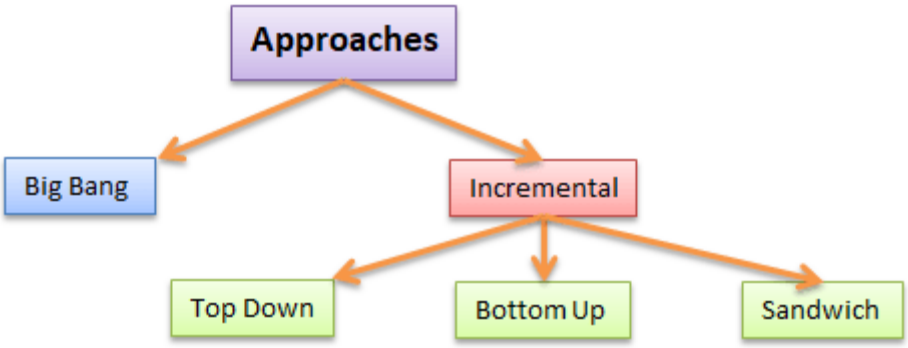
- Khi tích hợp các module vào hệ thống có thể không tương thích với cấu hình chung của hệ thống.

- Xử lý các ngoại lệ không đầy đủ có thể gây ra lỗi.

Test Case của Integration Testing khác với các Test Case khác, kiểm thử tích hợp tập trung chủ yếu vào các giao diện và luồng dữ liệu hay thông tin giữa các module. Bởi kiểm thử đơn vị đã được kiểm tra cho từng module nên ở đây không cần thiết để kiểm tra lại.

6.4.2.2 Phương pháp tiếp cận và Chiến lược của Integration Testing

Phương pháp tiếp cận trong Kiểm thử tích hợp:



Hình 6.6: Phương pháp tiếp cận trong Kiểm thử tích hợp

1. Big Bang: Tất cả các thành phần được tích hợp cùng một lúc, sau đó tiến hành kiểm thử.
Ưu điểm: Thuận tiện cho các hệ thống nhỏ.

Nhược điểm:

- ✓ Khó khăn trong việc phát hiện bug.
- ✓ Với số lượng giao diện cần được kiểm thử theo phương pháp này, một số giao diện liên kết cần kiểm thử có thể dễ dàng bị bỏ qua.
- ✓ Vì kiểm thử Tích hợp chỉ có thể bắt đầu sau khi tất cả các module được thiết kế, nên nhóm kiểm thử sẽ có ít thời gian thực hiện hơn trong giai đoạn kiểm thử.

- ✓ Vì tất cả các module được kiểm thử đồng thời, các module quan trọng có rủi ro cao không bị cô lập và được ưu tiên kiểm thử. Các module có liên quan đến giao diện người dùng cũng không bị cô lập và được ưu tiên kiểm thử.

2. Incremental Testing: Trong phương pháp này, kiểm thử được thực hiện bằng cách ghép hai hoặc nhiều module có liên quan đến logic. Sau đó, các module liên quan khác được thêm vào và kiểm thử chức năng thích hợp. Quá trình tiếp tục cho đến khi tất cả các module được thêm và hoàn thành quá trình kiểm thử. Cách tiếp cận tăng dần được thực hiện bởi hai Phương pháp khác nhau:

- Từ dưới lên (Bottom Up)

- Từ trên xuống (Top Down)

Stub và Driver là gì?

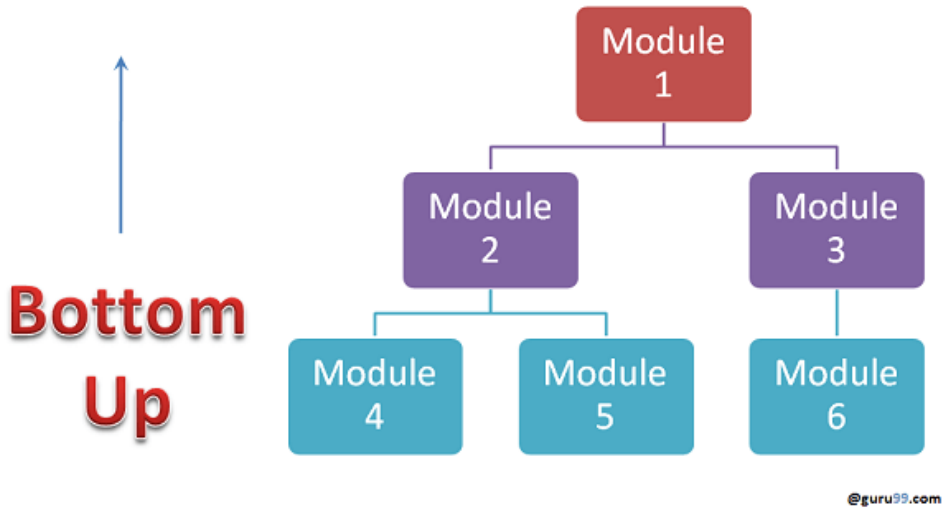
Phương pháp tiếp cận tăng dần được thực hiện bằng cách sử dụng các chương trình giả lập là Stub và Driver. Stub và Driver không thực hiện toàn bộ logic của module mà chỉ mô phỏng kết nối dữ liệu với module đang được gọi.

Stub: Được gọi bởi module đang kiểm thử.

Driver: Gọi module để được kiểm thử.

a) Bottom Up

Trong cách tích hợp từ dưới lên, mỗi module ở các cấp thấp hơn được kiểm thử với các module cao hơn cho đến khi tất cả các module được kiểm thử. Tích hợp từ dưới lên cần sự hỗ trợ của Driver để kiểm thử. Sơ đồ biểu diễn cách tiếp cận từ dưới lên:



Hình 6.7: Sơ đồ biểu diễn cách tiếp cận Bottom Up

***Ưu điểm*:**

- Việc phát hiện lỗi dễ dàng hơn.

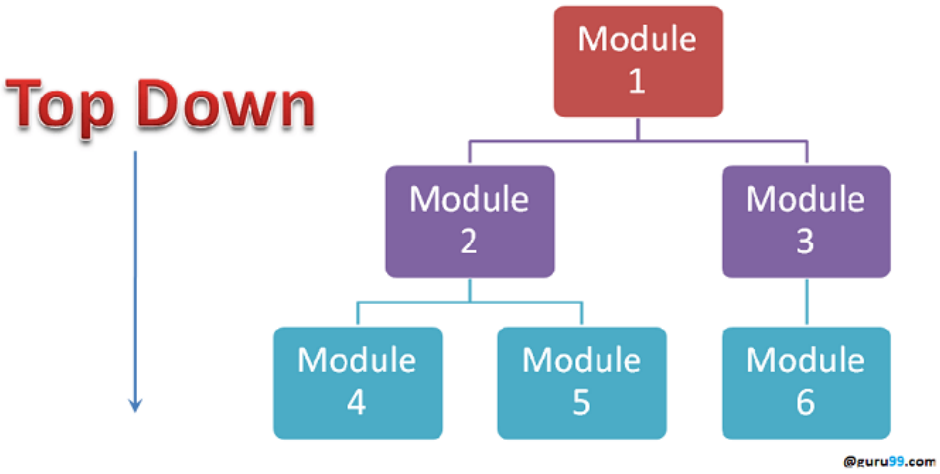
- Không bị lãng phí thời gian chờ đợi tất cả các module được xây dựng, không giống như phương pháp Big-bang

Nhược điểm:

- Các module quan trọng (ở cấp cao nhất của kiến trúc phần mềm) có luồng điều khiển được kiểm thử lần cuối nên dễ bị sót lỗi.
- Thực hiện kiểm thử tích hợp từ dưới lên từ sớm là không thể

b) Top Down

Trong cách tiếp cận từ trên xuống, kiểm thử diễn ra từ trên xuống dưới theo luồng điều khiển của hệ thống phần mềm. Cần sự hỗ trợ của Stub để kiểm thử. Sơ đồ biểu diễn cách tiếp cận từ trên xuống:



Hình 6.8: Sơ đồ biểu diễn cách tiếp cận Top Down

***Ưu điểm*:**

- Việc phát hiện lỗi dễ dàng hơn.
- Có khả năng thực hiện tích hợp từ trên xuống từ sớm.
- Các module quan trọng được ưu tiên kiểm thử; lỗi thiết kế quan trọng có thể được tìm thấy và sửa chữa đầu tiên.

Nhược điểm:

- Cần nhiều Stub.
- Các module ở mức thấp hơn không được kiểm thử đầy đủ.

c) Tích hợp Hybrid/ Sandwich

Chiến lược sandwich / hybrid là sự kết hợp của phương pháp Top Down và Bottom up. Các module trên cùng được kiểm thử cùng thời điểm với các module thấp hơn, đồng thời các module thấp hơn được tích hợp với các module ở trên và được thực hiện kiểm thử. Chiến lược này sử dụng Stubs cũng như Drivers.

6.4.2.3 Quy trình Integration Testing?

Quy trình kiểm thử tích hợp không phân biệt chiến lược kiểm thử phần mềm:

- ✓ Chuẩn bị kế hoạch kiểm thử tích hợp.
- ✓ Thiết kế các test scenarios, test cases và test scripts.
- ✓ Thực thi các test cases, báo cáo các lỗi nếu có.
- ✓ Theo dõi & kiểm thử lại các test cases có lỗi.
- ✓ Bước 3 và 4 được lặp lại cho đến khi kiểm thử hợp được hoàn thành.

Mô tả tóm tắt về kế hoạch Integration Testing: Kiểm thử tích hợp bao gồm các thuộc tính sau:

- ✓ Phương pháp/ hướng tiếp cận kiểm thử.
- ✓ Trong phạm vi và ngoài phạm vi kiểm thử tích hợp.
- ✓ Vai trò và trách nhiệm.
- ✓ Điều kiện tiền đề để kiểm thử tích hợp.
- ✓ Môi trường kiểm thử.
- ✓ Kế hoạch giảm thiểu rủi ro.

6.4.2.4 Tiêu chí bắt đầu và kết thúc của kiểm thử tích hợp

Tiêu chí bắt đầu và kết thúc của giai đoạn kiểm thử tích hợp trong bất kỳ mô hình phát triển phần mềm nào

Tiêu chí bắt đầu:

- ✓ Thành phần / module đã được kiểm thử đơn vị.
- ✓ Tất cả các lỗi có độ ưu tiên cao đã được sửa.
- ✓ Tất cả các module được hoàn thành và được tích hợp.
- ✓ Kế hoạch kiểm thử tích hợp, test cases, các kịch bản, tài liệu đã được thông qua.
- ✓ Môi trường kiểm thử được thiết lập theo yêu cầu để kiểm thử tích hợp.

Tiêu chí kết thúc:

- ✓ Kiểm thử tích hợp thành công.
- ✓ Các trường hợp kiểm thử đã thực thi được ghi lại
- ✓ Tất cả các lỗi có ưu tiên cao đã được sửa
- ✓ Tài liệu kỹ thuật được bàn giao.

6.4.2.5 Thực hiện kiểm thử tích hợp như thế nào để đạt kết quả tốt nhất?

- Đầu tiên, xác định Chiến lược kiểm thử tích hợp được thông qua, sau đó chuẩn bị các trường hợp kiểm thử và dữ liệu kiểm thử phù hợp.
- Nghiên cứu Kiến trúc của Ứng dụng trên bản thiết kế và xác định các module quan trọng, cần phải được ưu tiên kiểm thử ở giai đoạn này.
- Lấy các thiết kế giao diện từ nhóm Kiến trúc và tạo các trường hợp kiểm thử để xác định chi tiết tất cả các giao diện. Giao diện với cơ sở dữ liệu/ ứng dụng phần cứng/ phần mềm phải được kiểm thử chi tiết.
- Sau các trường hợp kiểm thử, dữ liệu kiểm thử cũng đóng vai trò quan trọng.
- Luôn chuẩn bị dữ liệu giả lập trước khi thực hiện kiểm thử. Không chuẩn bị dữ liệu kiểm thử trong khi thực hiện các trường hợp kiểm thử.

Như vậy có thể thấy kiểm thử tích hợp đóng vai trò rất quan trọng bởi nếu chỉ kiểm tra unit testing nhưng đến khi tích hợp các module lại với nhau lại gây ra lỗi thì như vậy sẽ làm cho phần mềm không đáp ứng được các nghiệp vụ mà khách hàng đã yêu cầu. Vì vậy chúng ta cần nên lưu ý thực hiện kiểm thử này sau kiểm thử đơn vị nhé!

6.4.3 System Testing (Kiểm thử hệ thống)

System Testing là mức thứ 3 của kiểm thử phần mềm cho phép phần mềm hoàn chỉnh và tích hợp được kiểm tra. System Testing tập trung nhiều hơn vào các chức năng của toàn bộ hệ thống. Kiểm thử hệ thống bao gồm kiểm thử chức năng và kiểm thử phi chức năng.

Mục đích của System Testing đó là kiểm tra thiết kế và toàn bộ hệ thống sau khi tích hợp có tuân thủ những yêu cầu đã được định sẵn trước đó hay không. Do đó System Testing rất chú trọng các hành vi và lỗi xuất hiện trên hệ thống. Người thực hiện giai đoạn System Testing thường là kiểm thử viên hoàn toàn độc lập so với nhóm phát triển dự án.

6.4.3.1 System Testing là gì?

System Testing (Kiểm thử hệ thống) là một phương pháp theo dõi và đánh giá hành vi của sản phẩm hoặc hệ thống phần mềm hoàn chỉnh và đã được tích hợp đầy đủ, dựa vào đặc tả và các yêu cầu chức năng đã được xác định trước. Đó là giải pháp cho câu hỏi "Liệu hệ thống hoàn chỉnh có hoạt động đúng với yêu cầu hay không?"

System Testing được thử nghiệm trong hộp đen, tức là chỉ có các tính năng làm việc bên ngoài của phần mềm được đánh giá trong quá trình thử nghiệm này. Nó không đòi hỏi bất kỳ kiến thức nội bộ nào về coding, lập trình, thiết kế, v.v. và hoàn toàn dựa trên quan điểm của người dùng.

Đặc điểm của System Testing:

- ✓ Trong Vòng đời phát triển phần mềm (SDLC), đây là thử nghiệm thực hiện nhiệm vụ kiểm tra toàn bộ phần mềm hoặc hệ thống.
- ✓ Đánh giá chức năng của hệ thống hoàn chỉnh theo yêu cầu chức năng được quyết định trước.
- ✓ Cùng với các yêu cầu chức năng, nó cũng xác minh và xác nhận các yêu cầu nghiệp vụ và kiến trúc của phần mềm.
- ✓ Staging server có thể hoạt động như một môi trường để thực hiện thử nghiệm.
- ✓ Một loại thử nghiệm hộp đen.
- ✓ Nó có thể bao gồm, cả thử nghiệm chức năng và phi chức năng.
- ✓ Giảm sự cố và bảo trì sau khi triển khai.
- ✓ Yêu cầu đội ngũ thử nghiệm độc lập với nhóm phát triển.

6.4.3.2 Khi nào thực hiện System Testing?

Như đã nêu trước đó, vòng đời kiểm thử phần mềm bao gồm nhiều cấp độ kiểm thử khác nhau, điều này khiến chúng ta phải hiểu khi nào, trong STLC mà system testing được thực hiện bởi những người kiểm thử. Dưới đây là các tình huống khi người kiểm thử có thể thực hiện system testing, bằng tay hoặc với sự hỗ trợ của các công cụ kiểm tra.

- Sau khi hoàn thành Unit & Integration testing.
- Trước khi bắt đầu Acceptance Testing
- Sau khi tích hợp hoàn toàn các mô-đun.
- Sau khi hoàn thành quy trình phát triển phần mềm, dựa trên đặc tả yêu cầu phần mềm (SRS).
- Sau khi môi trường thử nghiệm sẵn sàng.

6.4.3.3 Các lĩnh vực chính của System Testing

Một số khía cạnh, trong đó system testing tập trung vào:

- ✓ **Hiệu suất**: Đảm bảo rằng hệ thống phần mềm thực hiện theo yêu cầu của người dùng, mà không xuất hiện bất kỳ lỗi hoặc sự cố nào.
- ✓ **Bảo mật**: Bảo vệ sản phẩm khỏi mọi vi phạm bảo mật, đánh cắp dữ liệu, v.v., có thể mất dữ liệu & thông tin quan trọng của tổ chức.
- ✓ **Phục hồi**: Đảm bảo rằng sự phục hồi của hệ thống theo mong đợi.

- ✓ **Giao diện**: Kiểm tra hệ thống cũng tập trung vào giao diện của sản phẩm, đảm bảo rằng tất cả các yêu cầu được đáp ứng chính xác và không có sự cố xảy ra khi các thành phần của hệ thống được tích hợp với nhau.
- ✓ **Khả năng cài đặt**: Ở đây, trọng tâm của kiểm tra hệ thống là đảm bảo rằng sản phẩm được cài đặt và triển khai vào môi trường production mà không gặp bất kỳ khó khăn và sự cố nào.
- ✓ **Tính khả dụng**: Đây là một khía cạnh quan trọng khác được bao phủ bởi system testing. Nó đảm bảo trải nghiệm hệ thống của người dùng là tối ưu nhất.
- ✓ **Tài liệu**: Độ chính xác của tài liệu cũng được kiểm tra và giám sát bởi loại thử nghiệm này.
- ✓ **Load/Stress**: System testing cũng đảm bảo rằng hệ thống thực hiện và hoạt động chính xác dưới tải trọng và mức tải khác nhau.

6.4.3.4 Điều kiện tiên quyết của System Testing

Dưới đây là một số điều kiện tiên quyết quan trọng của system testing:

- Phải đảm bảo phần mềm được thống nhất kiểm tra.
- Kiểm thử tích hợp đã được thực hiện trên sản phẩm.
- Phần mềm nên được phát triển hoàn chỉnh.
- Trước khi thực hiện quy trình kiểm tra hệ thống, phải đảm bảo rằng môi trường kiểm tra đã sẵn sàng.

6.4.3.5 Hoàn thành quá trình System Testing

Vì tầm quan trọng của kiểm thử hệ thống là rất lớn trong STLC, điều quan trọng là chúng ta xác định quy trình của nó, để đảm bảo rằng quy trình được thực hiện chính xác mà không bỏ sót bất kỳ chi tiết hay bước quan trọng nào. Quá trình kiểm tra hệ thống có thể khác nhau tùy theo dự án. Tuy nhiên, có sáu bước phổ biến được xác định dưới đây:

- ➢ **Tạo Test Plan:** Bước đầu tiên là tạo kế hoạch kiểm tra, trong đó leader hoặc test manager xác định phạm vi & mục tiêu kiểm tra, xác định chiến lược, quyết định giữa kiểm tra thủ công và tự động, xác định tiêu chí đầu vào và đầu ra, gán vai trò và trách nhiệm.
- ➢ **Tạo Test Case:** Các trường hợp thử nghiệm được chuẩn bị trên cơ sở các use case và các yêu cầu của thử nghiệm, chẳng hạn như kỹ thuật, giao diện người dùng, chức năng, phi chức năng, hiệu suất, v.v.

- **Chọn Test Data:** Sau khi tạo test case, chúng sẽ phối hợp với nhau để chọn hoặc tạo test data cần thiết. Đây là những điều kiện đầu vào giúp nhóm nhận được kết quả mong đợi.
- **Thực hiện test case:** Cuối cùng là thực hiện kiểm thử các test case được tạo trước đó, liên tục theo dõi quá trình và ghi lại bất kỳ sự khác biệt hoặc vấn đề nào gặp phải trong quá trình này. Ngoài ra, đầu ra của thử nghiệm cũng được ghi lại ở đây.
- **Báo cáo & Sửa lỗi**: Báo cáo tất cả các lỗi và sự cố được ghi lại cho thành viên của team. Sau khi báo cáo, lập trình viên hoặc nhà phát triển làm việc với nhóm thử nghiệm để khắc phục và giải quyết vấn đề.
- **Lặp lại chu trình kiểm tra (Nếu cần):** Sau khi tất cả các vấn đề và lỗi được giải quyết và khắc phục, nhóm kiểm thử lặp lại chu trình kiểm tra để có kết quả như mong đợi.

6.4.3.6 Các loại kiểm tra được thực hiện trong System Testing

Kiểm thử hệ thống là sự kết hợp của các kỹ thuật kiểm thử đa năng, cho phép xác nhận hiệu suất và chức năng tổng thể của sản phẩm. Mỗi kỹ thuật kiểm tra này được tập trung vào các khía cạnh khác nhau của sản phẩm và phục vụ các yêu cầu khác nhau của khách hàng/ người dùng. Những loại system testing là:

- **Kiểm tra cài đặt:** Nó được sử dụng để kiểm tra chức năng mong muốn của phần mềm sau khi cài đặt thành công cùng với tất cả các yêu cầu cần thiết
- **Kiểm tra chức năng:** Một loại thử nghiệm hộp đen cho phép đánh giá hoạt động đúng của phần mềm theo các yêu cầu được xác định trước của nó.
- **Kiểm tra khả năng phục hồi:** Nó được thực hiện bằng cách cố làm cho phần mềm bị crash hoặc fail, để đánh giá khả năng phục hồi của sản phẩm một cách nhanh chóng.
- **Kiểm tra khả năng tương tác:** Nó đảm bảo khả năng phần mềm tương thích và tương tác với phần mềm hoặc hệ thống khác và các thành phần của chúng.
- **Kiểm tra năng suất:** Nó được thực hiện để kiểm tra phản ứng, độ ổn định, khả năng mở rộng, độ tin cậy và các số liệu chất lượng khác của phần mềm dưới các khối lượng công việc khác nhau.
- **Kiểm tra khả năng mở rộng:** Phần mềm phải có khả năng mở rộng, cùng với việc tăng tải, số lượng người dùng đồng thời, kích thước dữ liệu, v.v. Điều này nảy sinh nhu cầu kiểm tra khả năng mở rộng được tiến hành để xử lý các vấn đề liên quan đến khả năng mở rộng của phần mềm.

- **Kiểm tra độ tin cậy:** Việc kiểm tra này, đánh giá mức độ của phần mềm, giữa hai lỗi và thời gian cần thiết để sửa chữa.
- **Kiểm tra hồi quy:** Nó đảm bảo chức năng ban đầu của phần mềm sau mỗi lần sửa đổi trong đó.
- **Kiểm tra tài liệu:** Điều này bao gồm đánh giá tài liệu, được chuẩn bị trước và trong giai đoạn thử nghiệm, để đánh giá các yêu cầu thử nghiệm, bao gồm trong thử nghiệm tài liệu.
- **Kiểm tra bảo mật:** Để đánh giá các tính năng bảo mật của phần mềm để đảm bảo, bảo vệ, tính xác thực, bảo mật và tính toàn vẹn của thông tin và dữ liệu.
- **Kiểm tra khả năng sử dụng:** Đảm bảo tính năng thân thiện với người dùng của phần mềm và ngăn việc end user gặp phải sự cố trong quá trình sử dụng sản phẩm phần mềm.

6.4.3.7 Lý do thực hiện System Testing

Một số lý do để thực hiện system test này là:

- ✓ Đảm bảo sản phẩm đáp ứng các tiêu chuẩn chất lượng.
- ✓ Xác minh hệ thống phần mềm đáp ứng các yêu cầu chức năng, kỹ thuật và kinh doanh theo yêu cầu của khách hàng.
- ✓ Thực hiện kiểm tra từ đầu đến cuối của sản phẩm phần mềm giúp ngăn ngừa lỗi hệ thống và sự cố trong quá trình thực hiện với môi trường thật.
- ✓ Được thực hiện trong một môi trường tương tự như môi trường production, cho phép các nhà phát triển cũng như các bên liên quan có được ý tưởng về phản ứng của người dùng đối với sản phẩm.
- ✓ Đóng một vai trò quan trọng trong việc cung cấp một sản phẩm chất lượng cho người dùng cuối.
- ✓ Chính trong giai đoạn này của vòng đời kiểm thử phần mềm (STLC), các Yêu cầu nghiệp vụ và Kiến trúc ứng dụng được kiểm tra.
- ✓ Đảm bảo rằng đầu vào được cung cấp đầu ra / kết quả như mong đợi.

System testing là một phần không thể thiếu trong vòng đời kiểm thử phần mềm, được thực hiện khi quá trình phát triển phần mềm hoàn tất và sản phẩm đã trải qua kiểm thử đơn vị và tích hợp. Thử nghiệm này không giới hạn ở một khía cạnh hoặc thành phần của sản phẩm mà được sử dụng để kiểm tra toàn bộ hệ thống phần mềm, điều này làm cho nó trở thành một phần quan trọng của bất kỳ chu kỳ thử nghiệm nào và do đó nó được thực hiện bởi tester, những người mong muốn cung cấp trải nghiệm người dùng tốt nhất [24] [25].

6.4.4 Acceptance Testing (Kiểm thử chấp nhận) [26][27][28]

Acceptance Testing được thực hiện bởi khách hàng hoặc ủy quyền cho nhóm thứ ba nhằm kiểm tra hệ thống vừa xây dựng đã phù hợp với yêu cầu của khách hàng trước đó hay chưa. Mục đích của Acceptance Testing đó là xác nhận lại sự tin tưởng vào hệ thống, các đặc tính thuộc về chức năng hoặc phi chức năng của hệ thống.

Acceptance test gồm 2 loại kiểm thử là:

- **Alpha Test**, người dùng kiểm thử phần mềm ngay tại nơi phát triển phần mềm, lập trình viên sẽ ghi nhận các lỗi hoặc phản hồi, và lên kế hoạch sửa chữa.
- **Beta Test**, phần mềm sẽ được gửi tới cho người dùng để kiểm thử ngay trong môi trường thực, lỗi hoặc phản hồi cũng sẽ gửi ngược lại cho lập trình viên để sửa chữa. Lưu ý không nhất thiết phải thực hiện tất cả các loại kiểm tra nêu trên. Tùy yêu cầu và đặc trưng của từng hệ thống, tuỳ khả năng và thời gian cho phép của dự án, khi lập kế hoạch, trưởng dự án sẽ quyết định áp dụng những loại kiểm tra nào.

6.4.4.1 Acceptance Testing là gì?

Acceptance Testing (Kiểm thử chấp nhận) là một kiểm thử nhằm xác định hệ thống phần mềm có đạt yêu cầu kỹ thuật hay không. Bằng việc kiểm tra các hành vi của hệ thống qua dữ liệu thực tế, kiểm thử chấp nhận sẽ xác định có hay không việc hệ thống đáp ứng được các tiêu chí lẫn yêu cầu của khách hàng. Một số kỹ thuật được sử dụng trong Acceptance Testing đó là phân tích giá trị biên giới, phân vùng tương đương và sử dụng bảng quyết định.

User Acceptance Testing (UAT) – Kiểm thử chấp nhận của người dùng có nghĩa là kiểm thử xem phần mềm đã thỏa mãn tất cả yêu cầu của khách hàng và khách hàng chấp nhận sản phẩm (và trả tiền thanh toán hợp đồng) hay không?

Cụ thể hơn UAT trả lời cho những câu hỏi sau:

- *"Cái mình làm ra có phải là cái User muốn không?"*
- *"User có thấy lợi ích phần mềm mang lại xứng đáng so với công sức, tiền bạc, thời gian và cả thông tin mà họ cung cấp cho mình không?"*
- *"User có dễ dàng hiểu và xử lý vấn đề khi gặp lỗi không?"*
- *"User có cảm thấy giao diện hợp với ý họ, giúp họ tập trung vào nội dung và công việc cần làm?"*
- *"Bạn có đang giúp User tiết kiệm các bước làm việc khi họ đã quen với hệ thống?"*

Acceptance Testing - Mục tiêu chính đằng sau kiểm thử chấp nhận là để kiểm tra xem sản phẩm phần mềm được phát triển có vượt qua các tiêu chuẩn chấp nhận được xác định trên cơ sở yêu cầu của người dùng và doanh nghiệp hay không, để tuyên bố rằng người dùng có thể chấp nhận hoặc không chấp nhận sử dụng sản phẩm đó.

Acceptance Testing sẽ được thực hiện khi nào?

Đây thường là bước cuối cùng trước khi sản phẩm được đưa ra hoạt động hoặc trước khi phân phối sản phẩm phải được chấp nhận. Acceptance Testing được thực hiện sau khi bản thân sản phẩm được kiểm tra kỹ lưỡng (tức là sau khi kiểm thử hệ thống).

Điều kiện tiên quyết của Acceptance Testing:

- ✓ Phải đảm bảo các yêu cầu nghiệp vụ chính của ứng dụng hoạt động
- ✓ Phần mềm đã được hoàn thiện nhất
- ✓ Các khâu kiểm thử Unit testing, integration testing, system testing đã được hoàn thành
- ✓ Không có lỗi quan trọng còn tồn tại trong hệ thống
- ✓ Lỗi về thẩm mỹ được chấp nhận trước UAT
- ✓ Regression testing phải được hoàn thành và không có lỗi lớn
- ✓ Tất cả các lỗi đã phát hiện phải được sửa và kiểm tra trước khi UAT
- ✓ Môi trường Acceptance Testing phải được chuẩn bị sẵn sàng
- ✓ Nhà phát triển phải chắc chắn rằng hệ thống đã sẵn sàng thực hiện Acceptance Testing

6.4.4.2 Các bước thực hiện Acceptance Testing

- Phân tích các yêu cầu nghiệp vụ của phần mềm
- Tạo kế hoạch kiểm tra Acceptance Testing
- Xác định các kịch bản kiểm thử
- Tạo các trường hợp kiểm tra Acceptance Testing
- Chuẩn bị data test (giống với data thật nhất)
- Thực hiện kiểm thử
- Ghi nhận kết quả
- Xác nhận các chức năng của sản phẩm

6.4.4.3 Những chuẩn bị tốt nhất cho Acceptance Testing

- Chuẩn bị kế hoạch Acceptance Testing sớm

- Chuẩn bị các test case kiểm thử trước khi bắt đầu Acceptance Testing
- Xác định rõ mục tiêu và phạm vi của Acceptance Testing
- Thực hiện kiểm thử với các kịch bản và dữ liệu thực tế
- Không đè nặng tư tưởng là người xây dựng ứng dụng mà thực hiện như một người dùng sản phẩm
- Kiểm tra khả năng sử dụng
- Báo cáo kết quả trước khi quyết định phát hành sản phẩm

6.4.4.4 Những điểm quan trọng trong kiểm thử chấp nhận

- Kiểm thử chấp nhận xác định xem tất cả những chức năng chính đều hoạt động tốt. Nếu người dùng tìm thấy bug ở những chức năng chính thì tester sẽ phải xem xét lại test case, tìm hiểu nguyên nhân tại sao xảy ra bug đó.
- Đây cũng là cơ hội để tìm thấy lỗi còn tồn tại trong hệ thống
- Kiểm thử chấp nhận được chia làm hai loại: thử nghiệm Alpha và Beta
- Hầu hết trong một dự án phát triển phần mềm thường thì UAT được thực hiện trong môi trường đảm bảo chất lượng nếu không có môi trường dàn dựng hoặc môi trường Acceptance Testing.

6.5 Kỹ thuật kiểm thử phần mềm [29]

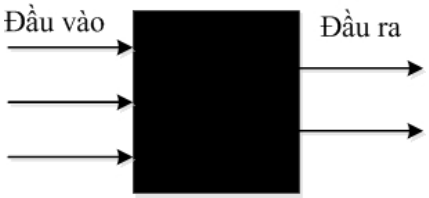
Kỹ thuật kiểm thử phần mềm giúp thiết kế các ca kiểm thử tốt hơn. Vì kiểm thử toàn diện là không thể, các kỹ thuật kiểm thử giúp giảm số lượng các test case được thực hiện trong khi tăng phạm vi kiểm thử, giúp xác định các điều kiện kiểm thử khó nhận biết.

Có hai cách tiếp cận cơ bản để xác định các ca kiểm thử là kiểm thử hàm (kiểm thử chức năng hay kiểm thử hộp đen - black-box testing) và kiểm thử cấu trúc (kiểm thử hộp trắng - white-box testing). Mỗi cách tiếp cận có phương pháp xác định ca kiểm thử khác nhau.

6.5.1 Kiểm thử hàm (functional testing)

Kiểm thử hàm dựa trên quan niệm rằng bất kỳ chương trình nào cũng được coi là một hàm ánh xạ các giá trị từ miền dữ liệu đầu vào vào miền dữ liệu đầu ra của nó. Khái niệm này được dùng chung trong kỹ thuật khi các hệ thống đều được coi là các hộp đen. Chính điều này dẫn đến thuật ngữ kiểm thử hộp đen, trong đó nội dung của hộp đen (việc cài đặt) không được biết/không cần quan tâm, và chức năng của hộp đen được hiểu theo các dữ liệu đầu vào và dữ liệu đầu ra của nó như hình 6.6. Trong thực tế, chúng ta thường thao tác hiệu quả với những kiến thức về hộp đen. Chính điều này là trung tâm của khái niệm định hướng đối tượng nơi mà

các đối tượng được xem xét như là các hộp đen và chúng chỉ tương tác với nhau bằng các lời gọi thông qua các phương thức có thể quan sát được từ bên ngoài.



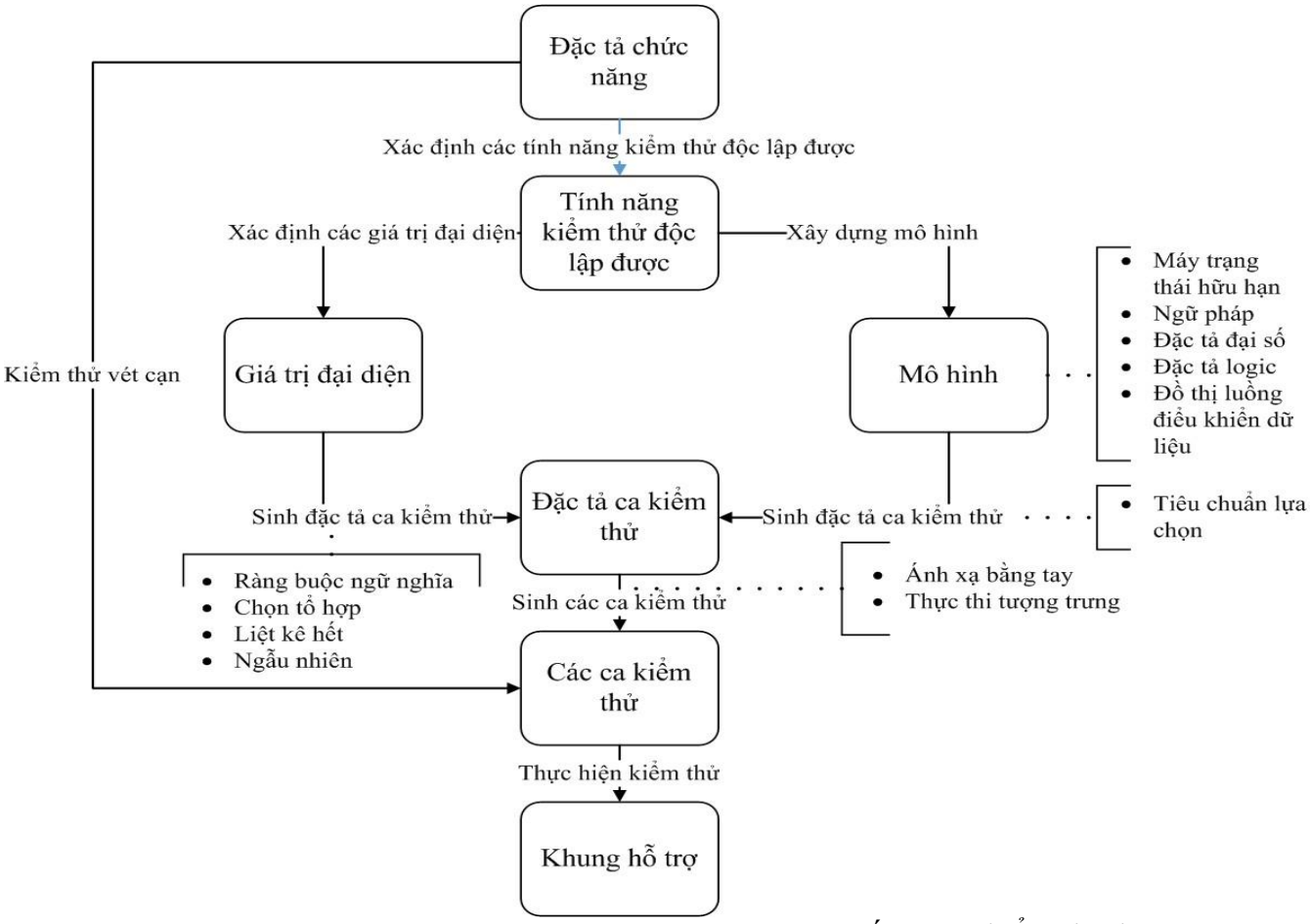
Hình 6.9: Một hộp đen kỹ thuật

Với cách tiếp cận của kiểm thử hàm, để xác định các ca kiểm thử, thông tin duy nhất được dùng là đặc tả của phần mềm cần kiểm thử. Có hai lợi điểm chính của các ca kiểm thử được sinh ra bởi cách tiếp cận kiểm thử hàm: chúng độc lập với việc phần mềm được cài đặt thế nào, và vì thế khi cài đặt thay đổi thì các ca kiểm thử vẫn dùng được, đồng thời các ca kiểm thử được phát triển song song và độc lập với việc cài đặt hệ thống. Do đó, cách tiếp cận này rút gọn được thời gian phát triển của dự án. Điểm yếu của cách tiếp cận này là các ca kiểm thử hàm thường có thể có tính dư thừa đáng kể trong các ca kiểm thử.

Cần nhấn mạnh rằng kiểm thử hàm là việc thiết kế các ca kiểm thử hàm của chương trình chỉ dựa trên đặc tả của chương trình mà không dựa trên việc phân tích mã nguồn của chương trình. Kiểm thử hàm còn được gọi với tên khác là kiểm thử dựa trên đặc tả, hay kiểm thử hộp đen, hay kiểm thử chức năng. Kiểm thử hàm là một kỹ thuật cơ bản giúp phát hiện nhiều lỗi mà kiểm thử hộp trắng không phát hiện được. Ví dụ dễ thấy nhất là các lỗi do cài đặt (implementation) không đầy đủ so với đặc tả. Nếu phần mềm thiếu hẳn một chức năng thì chúng ta dễ nhận thấy, nhưng nếu phần mềm thiếu một trường hợp của một chức năng thì việc này không dễ phát hiện nếu không kiểm tra kỹ chương trình so với đặc tả.

6.5.1.1 Phương pháp hệ thống

Việc xác định các ca kiểm thử từ đặc tả chức năng là quá trình phân tích để xác định và chia không gian đầu vào/đầu ra/hành vi của chương trình thành các miền con. Phương pháp hệ thống sẽ chia quá trình này thành các bước cơ bản, từ đó làm từng bước được dễ dàng hơn hoặc tự động hóa một số bước nếu có thể. Phương pháp hệ thống giúp chúng ta xác định bộ ca kiểm thử đã đủ chưa đồng thời tránh tạo ra nhiều ca kiểm thử trùng lặp, không cần thiết.



Hình 6.10: Các bước chính của phương pháp hệ thống cho kiểm thử hàm.

Xác định các chức năng kiểm thử độc lập được: Đặc tả phần mềm trên thực tế là lớn và phức tạp ngay cả với những hệ thống nhỏ. Chúng thường được phân rã thành các đơn vị nhỏ hơn như các hệ thống con hay cụ thể hơn là các ca sử dụng (use case) hoặc các kịch bản sử dụng (user stories). Chúng ta gọi chung là các chức năng. Ví dụ trang web tìm kiếm của Google có thể có tính năng tìm kiếm, hiển thị kết quả, hiện thị quảng cáo, hay đăng ký người sử dụng mới, v.v.. Tuy nhiên ranh giới của một chức năng nhiều khi không rõ ràng. Ví dụ chức năng tìm kiếm của Google Search có chức năng tính toán một biểu thức số học nếu văn bản tìm kiếm là một biểu thức số học.

Khi thiết kế kiểm thử chúng ta tách các chức năng càng chi tiết, cụ thể càng tốt vì nó vừa giúp đơn giản việc thiết kế kiểm thử và vừa cho phép chúng ta xác định lỗi trong chương trình dễ dàng hơn. Mỗi ca kiểm thử chỉ nên kiểm tra một kịch bản cụ thể, chi tiết của một chức năng của hệ thống.

Tiếp đó với mỗi chức năng chúng ta sẽ xác định tất cả các đầu vào có ảnh hưởng đến việc thực hiện chức năng đó và các đầu ra có thể bị tác động bởi chức năng đó. Các đầu vào và đầu ra thường được mô tả tường minh trong tài liệu đặc tả.

Nhưng cũng có nhiều trường hợp các đầu vào và đầu ra này nằm ẩn trong tài liệu đặc tả mà người thiết kế phải phát hiện ra. Ví dụ như các đặc tả không hình thức mô tả chức năng đăng ký tài khoản mới thường chỉ nêu các trường dữ liệu cần khai báo nhưng không nói gì đến chi tiết kiểu dữ liệu sẽ lưu thông tin hồ sơ người sử dụng.

Xác định các lớp giá trị đại diện: Các giá trị đại diện có thể được xác định trực tiếp từ đặc tả phi hình thức, như ngôn ngữ tự nhiên. Chúng cũng có thể được xác định gián tiếp dựa trên một mô hình được xây dựng để phục vụ cho việc này. Trong cả hai trường hợp này mục tiêu đều là xác định các giá trị của mỗi đầu vào một cách độc lập, bằng việc lấy các giá trị đã liệt kê tường minh trong đặc tả, hoặc thông qua một mô hình phù hợp. Chúng ta chưa xét đến tổ hợp của các giá trị này. Lý do là chúng ta đang tách bài toán xác định các lớp giá trị của mỗi đầu vào và bài toán tổ hợp chúng để tạo thành các ca kiểm thử. Tức là chúng ta đang tách một bước phức tạp thành các bước đơn giản hơn.

Với các đặc tả không hình thức người thiết kế kiểm thử phải dựa vào việc liệt kê tường minh các lớp giá trị đại diện. Trong trường hợp này người thiết kế kiểm thử cần cẩn thận xét tất cả các trường hợp có thể và khai thác tối đa những thông tin có sẵn trong đặc tả. Tương tự, với kết quả mong đợi đầu ra, chúng ta cũng có thể xác định các lớp giá trị của miền đầu ra, cũng như các giá trị biên và các giá trị đặc biệt dễ có lỗi.

Ví dụ khi xem xét các thao tác trên một danh sách không rỗng chúng ta cần xét cả trường hợp danh sách rỗng (giá trị lỗi) và danh sách có một phần tử (giá trị biên chiều dài danh sách) là các trường hợp đặc biệt. Ở bước này chúng ta chỉ quan tâm đến tính chất của giá trị, chứ chưa cần cụ thể giá trị là gì. Tức là chúng ta quan tâm đến đặc tả ca kiểm thứ chứ chưa phải ca kiểm thử cụ thể. Với ví dụ danh sách một phần tử, chúng ta chưa cần quan tâm phần tử của danh sách này cụ thể là gì, chỉ biết danh sách chỉ có một phần tử.

Để liệt kê các giá trị không tường minh chúng ta cần xây dựng mô hình hoặc một phần của mô hình của đặc tả. Ví dụ chúng ta có thể dùng văn phạm để mô tả ngôn ngữ và sinh các từ của ngôn ngữ này để liệt kê các giá trị cần xác định. Mô hình này có thể có sẵn trong đặc tả nhưng thông thường người thiết kế kiểm thử phải xây dựng chúng.

Trong hai cách trên, việc liệt kê trực tiếp có vẻ đơn giản hơn và ít tốn kém hơn việc xây dựng mô hình để sinh ra các giá trị. Tuy nhiên về lâu về dài các mô hình sẽ có giá trị hơn, khi khối lượng giá trị lớn chúng ta có thể tự động hóa việc liệt kê, hay chúng ta cũng có thể điều chỉnh số lượng ca kiểm thử nhiều hay ít để cân bằng với các ràng buộc kiểm thử khác, như chi phí, thời gian. Các mô hình cũng có thể được sử dụng lại, sửa đổi dễ dàng hơn trong quá trình phát triển hệ thống. Quyết

định cuối cùng nên dùng cách nào vẫn tùy thuộc vào lĩnh vực ứng dụng, kỹ năng của người thiết kế kiểm thử, và các công cụ thích hợp đang có.

Sinh các đặc tả ca kiểm thử: Đặc tả kiểm thử được tạo ra bằng việc tổ hợp các giá trị của các đầu vào của chương trình đang được kiểm thử. Bước trên chúng ta đã xác định các giá trị đại diện thì ở bước này chúng ta sinh các tổ hợp của chúng bằng tích Đề-các của các giá trị đại diện đã chọn. Nếu ở bước trước chúng ta tạo ra mô hình hình thức thì ở bước này các đặc tả ca kiểm thử sẽ là các hành vi cụ thể hoặc tổ hợp của các tham số của mô hình và một đặc tả ca kiểm thử có thể được thỏa mãn bởi nhiều đầu vào cụ thể. Trong cả hai cách này việc tạo ra tất cả các tổ hợp thường tạo ra số lượng quá lớn các ca kiểm thử.

Ví dụ một hàm có năm đầu vào, mỗi đầu vào có sáu giá trị đại diện thì tích Đề-các sẽ tạo ra $6^5 = 7,776$ đặc tả ca kiểm thử. Số lượng này là quá lớn với một chức năng, ngay cả khi được tự động hóa và chạy bằng máy tính hoàn toàn. Thông thường, rất nhiều tổ hợp là không hợp lệ hoặc không khả thi. Chúng ta sẽ học các phương pháp để giảm số tổ hợp này nhưng vẫn đảm bảo chất lượng của bộ kiểm thử.

Để giảm số ca kiểm thử, chúng ta cần loại bỏ các tổ hợp giá trị không hợp lệ bằng việc đưa thêm các ràng buộc về cách tổ hợp. Ví dụ hàm *NextDate* chúng ta cần đưa ràng buộc để tránh tạo ra các giá trị không hợp lệ như ngày 31/11/2013. Hoặc chúng ta cũng không xét hết tất cả các tích Đề-các mà chỉ cần mỗi cặp giá trị của hai biến bất kỳ đều xuất hiện là đủ. Quá trình này có thể không phải một bước làm là xong mà có thể cần điều chỉnh và thử một số lần để có kết quả là bộ ca kiểm thử phù hợp, vừa hạn chế được các ca kiểm thử vô lý không cần thiết, vừa tiết kiệm được chi phí thực hiện kiểm thử.

Sinh ca kiểm thử và thực hiện kiểm thử: Quá trình kiểm thử hoàn tất khi chúng ta cụ thể hóa các đặc tả kiểm thử thành các ca kiểm thử cụ thể, rồi tính kết quả mong đợi của từng ca kiểm thử, và tiến hành kiểm tra với phần mềm. Việc thực hiện kiểm thử cần được tự động hóa tối đa khi có thể, vì một hệ thống phần mềm cần được kiểm thử rất nhiều lần. Ngay cả khi sản phẩm hoàn tất, trong tương lai chúng ta vẫn sẽ cần thực hiện kiểm thử nhiều lần nữa, khi phần mềm tiến hóa, thay đổi theo những yêu cầu mới, hay các lỗi được phát hiện và phần mềm cần phải sửa.

6.5.1.2 Các kỹ thuật kiểm thử hộp đen

- Kiểm thử giá trị biên (Boundary Value Analysis - BVA)
- Kiểm thử lớp tương đương (Equivalence Class Partitioning)
- Kiểm thử bằng Bảng quyết định (Decision Table based testing)

- Kiểm thử tổ hợp
- Chuyển đổi trạng thái (State Transition)
- Đoán lỗi (Error Guessing)

6.5.2 Kiểm thử cấu trúc

Kiểm thử cấu trúc hay kiểm thử hộp trắng là cách tiếp cận khác để xác định các ca kiểm thử. Biểu tượng hộp trong suốt là thích hợp cho cách tiếp cận này.

Kiểm thử hộp trắng sử dụng các chiến lược cụ thể và sử dụng mã nguồn của chương trình/đơn vị phần mềm cần kiểm thử nhằm kiểm tra xem chương trình/đơn vị phần mềm có thực hiện đúng so với thiết kế và đặc tả hay không. Trong khi các phương pháp kiểm thử hộp đen chỉ cho phép phát hiện các lỗi/khiếm khuyết có thể quan sát được, kiểm thử hộp trắng cho phép phát hiện các lỗi/khiếm khuyết tiềm ẩn bên trong chương trình/đơn vị phần mềm. Các lỗi này thường khó phát hiện bởi các phương pháp kiểm thử hộp đen. Kiểm thử hộp đen và kiểm thử hộp trắng không thể thay thế cho nhau mà chúng cần được sử dụng kết hợp với nhau trong một quy trình kiểm thử thống nhất nhằm đảm bảo chất lượng phần mềm. Tuy nhiên, để áp dụng các phương pháp kiểm thử hộp trắng, người kiểm thử không chỉ cần hiểu rõ giải thuật mà còn cần có các kỹ năng và kiến thức tốt về ngôn ngữ lập trình được dùng để phát triển phần mềm, nhằm hiểu rõ mã nguồn của chương trình/đơn vị phần mềm cần kiểm thử. Do vậy, việc áp dụng các phương pháp kiểm thử hộp trắng thường tốn thời gian và công sức nhất là khi chương trình/đơn vị phần mềm có kích thước lớn.

Hai phương pháp chủ yếu được sử dụng trong kiểm thử hộp trắng là:

- Kiểm thử dòng điều khiển (control flow testing): Phương pháp kiểm thử dòng điều khiển tập trung kiểm thử tính đúng đắn của các giải thuật sử dụng trong các chương trình/đơn vị phần mềm.

- Kiểm thử dòng dữ liệu (data flow testing): Phương pháp kiểm thử dòng dữ liệu tập trung kiểm thử tính đúng đắn của việc sử dụng các biến dữ liệu sử dụng trong chương trình/đơn vị phần mềm.

Kiểm thử dòng điều khiển:

Kiểm thử dòng điều khiển là một trong những phương pháp kiểm thử quan trọng nhất của chiến lược kiểm thử hộp trắng cho các chương trình/đơn vị chương trình. Phương pháp này cho phép phát hiện ra các lỗi (có thể có) tiềm ẩn bên trong chương trình/đơn vị chương trình bằng cách kiểm thử các đường đi của nó tương ứng với các dòng điều khiển có thể có. Để áp dụng phương pháp này, chúng ta cần xác định độ đo kiểm thử (cần kiểm thử với độ đo nào?). Tiếp theo, đồ thị dòng điều khiển của chương trình/đơn vị chương trình ứng với độ đo kiểm thử sẽ được tạo ra.

Dựa vào đồ thị này, chúng ta sẽ sinh ra các đường đi độc lập. Số đường đi độc lập này là các trường hợp tối thiểu nhất để đảm bảo 100% độ bao phủ ứng với độ đo yêu cầu. Với mỗi đường đi, chúng ta sẽ sinh ra một ca kiểm thử sao cho khi nó được dùng để kiểm thử thì đường đi này được thực thi. Việc sinh các đầu vào cho các ca kiểm thử này là một bài toán thú vị. Chúng ta sẽ chọn một bộ đầu vào sao cho thỏa mãn các điểm quyết định có trong đường đi tương ứng. Giá trị đầu ra mong muốn ứng với mỗi bộ đầu vào của mỗi ca kiểm thử cũng sẽ được tính toán. Đây là bài toán khó và thường chỉ có các chuyên gia phân tích chương trình mới có thể trả lời chính xác giá trị này. Cuối cùng, các ca kiểm thử được chạy nhằm phát hiện ra các lỗi của chương trình/đơn vị chương trình cần kiểm thử. Khi một lỗi được phát hiện bởi một ca kiểm thử nào đó, nó sẽ được thông báo tới lập trình viên tương ứng. Lập trình viên sẽ tiến hành sửa lỗi (phát hiện vị trí của lỗi ở câu lệnh nào và sửa nó). Trong trường hợp này, chúng ta không chỉ thực hiện lại ca kiểm thử phát hiện ra lỗi này mà phải thực hiện lại tất cả các ca kiểm thử của đơn vị chương trình. Lý do chúng ta phải thực hiện công việc này là vì khi sửa lỗi này có thể gây ra một số lỗi khác.

Việc áp dụng phương pháp kiểm thử dòng điều khiển là khó và tốn kém hơn các phương pháp kiểm thử hộp đen (phân hoạch tương đương, phân tích giá trị biên, bảng quyết định, ...). Để áp dụng kỹ thuật này, chúng ta cần đội ngũ nhân lực về kiểm thử có kiến thức và kỹ năng tốt. Hơn nữa, chúng ta cần một sự đầu tư lớn về các nguồn lực khác (tài chính, thời gian, ...) mới có thể thực hiện tốt phương pháp này. Đây là một yêu cầu khó và không nhiều công ty phần mềm đáp ứng được. Kiểm thử dòng điều khiển tự động hứa hẹn sẽ là một giải pháp tốt nhằm giúp cho các công ty giải quyết những khó khăn này. Hiện nay, đã có nhiều công cụ hỗ trợ một phần hoặc hoàn toàn các bước trong phương pháp này.

Phương pháp kiểm thử dòng điều khiển cho phép sinh ra các ca kiểm thử tương ứng với các đường đi (dòng điều khiển) của chương trình. Tuy nhiên, chỉ áp dụng phương pháp này là chưa đủ để phát hiện tất cả các lỗi tiềm ẩn bên trong chương trình. Trong thực tế, các lỗi thường hay xuất hiện tại các biến được sử dụng trong chương trình/đơn vị chương trình. Kiểm thử dòng dữ liệu cho phép ta phát hiện những lỗi này. Bằng cách áp dụng cả hai phương pháp này, chúng ta khá tự tin về chất lượng của sản phẩm phần mềm.

Kiểm thử dòng dữ liệu:

Phương pháp kiểm thử dòng dữ liệu cho phép chúng ta phát hiện các lỗi tiềm ẩn bên trong chương trình liên quan đến việc sử dụng các biến. Các lỗi này thường khó phát hiện bởi các phương pháp khác. Để áp dụng phương pháp này, chúng ta cần xây dựng đồ thị dòng dữ liệu cho chương trình cần kiểm thử. Ứng với mỗi độ đo kiểm thử, chúng ta sẽ xây dựng tập các đường đi đầy đủ (*Complete-paths*) để sinh

các ca kiểm thử tương ứng. Chúng ta nên sử dụng kiểm thử dòng dữ liệu khi chương trình có nhiều tính toán, chương trình có nhiều lệnh rẽ nhánh và biến của biểu thức điều kiện này cũng được tính toán (*p-use*). Tuy nhiên, so với phương pháp kiểm thử dòng điều khiển, phương pháp kiểm thử dòng dữ liệu khó áp dụng hơn và có độ khó hơn. Hơn nữa, phương pháp này thường khá phức tạp khi áp dụng với các đơn vị chương trình có kích thước lớn.

6.6 Case study: Kiểm thử phần mềm

a) Mục tiêu

- Có kỹ năng thiết kế kịch bản kiểm thử, các trường hợp kiểm thử
- Cài đặt và sử dụng các công cụ giúp kiểm thử tự động.

b) Yêu cầu

1. Lập kế hoạch kiểm thử

2. Thiết kế kịch bản kiểm thử, thiết kế các Test Case cơ bản

3. Thực thi test: Có sử dụng các công cụ kiểm thử tự động

4. Báo cáo kết quả kiểm thử thực tế trên phần mềm, đánh giá kết quả và giải pháp khắc phục (nếu có).

c) Hướng dẫn, gợi ý: [33 – Chapter 5]

Câu hỏi

Câu 1. Phương pháp kiểm thử tính logic bên trong của các mô đun phần mềm thường dùng là gì ?

a) Kiểm thử hộp đen
b) Kiểm thử hộp trắng
c) Kiểm thử hộp xám
d) Kiểm thử mức đơn thể

Câu 2. Hoạt động quan sát kết quả kiểm thử cần thực hiện khi nào?

a) Chỉ cần quan sát và đánh giá kết quả sau khi thực thi chương trình
b) Chỉ cần quan sát và đánh giá kết quả sau khi thực thi từng module chương trình
c) Cả (a) và (b) đều đúng
d) Cần quan sát trong khi và sau khi thực thi chương trình, so sánh kết quả nhận được và kết quả mong đợi

Câu 3. Kiểm thử hệ thống được thực hiện khi nào?

a) Sau khi kiểm thử đơn vị kết thúc

b) Trong khi đang kiểm thử tích hợp
c) Đồng thời với các hoạt động kiểm thử đơn vị và kiểm thử tích hợp
d) Sau khi kiểm thử tích hợp kết thúc

Câu 4. Một ca kiểm thử (Test case) bao gồm?

a) Tập dữ liệu kiểm thử, điều kiện thực thi, kết quả mong đợi
b) Tập dữ liệu kiểm thử, các bước thực hiện khi kiểm thử
c) Tập dữ liệu kiểm thử, các đánh giá kết quả kiểm thử
d) Tập dữ liệu kiểm thử, Tester

Câu 5. Kiểm thử hồi quy (regression testing) …

a) Là kiểm thử dựa trên thông tin phản hồi của khách hàng khi sử dụng hệ thống
b) Được thực hiện sau khi phần mềm đưa vào sử dụng
c) Cả (a) và (b) đều đúng
d) Được thực hiện trong khi kiểm thử hệ thống

Bài tập cuối chương

Bài 6.1. Nếu lập trình viên được thông báo là có thể dùng tên biến với 8 ký tự với ký tự đầu viết hoa (chữ in). Điều đó là chuẩn hay hướng dẫn?

Bài 6.2. Viết chương trình tính nghiệm của phương trình bậc hai $ax^2 + bx + c$. Hãy áp dụng các kỹ thuật kiểm thử của chương này để xây dựng bộ dữ liệu kiểm thử cho chương trình.

Bài 6.3. Xây dựng bảng quyết định cho bài toán giải phương trình bậc hai.

Bài 6.4. Tại sao chúng ta cần thực hiện kiểm thử hộp trắng?

Bài 6.5. Phân biệt kiểm thử hộp trắng và kiểm thử hộp đen.

Bài 6.6. Tại sao kiểm thử hộp trắng thường có chi phí và độ khó cao hơn kiểm thử hộp đen?

Bài 6.7. Thế nào là đồ thị dòng điều khiển của một chương trình/đơn vị chương trình?

Bài 6.8. Trình bày các độ đo kiểm thử cho kiểm thử dòng điều khiển.

Bài 6.9. Trình bày các bước nhằm kiểm thử một đơn vị chương trình theo phương pháp kiểm thử dòng điều khiển với một độ đo kiểm thử cho trước.

Bài 6.10. So sánh phương pháp kiểm thử dòng điều khiển và phương pháp kiểm thử dòng dữ liệu. Liệu chúng có thay thế lẫn nhau được không? Hãy giải thích.

Chương VII: TRIỂN KHAI VÀ BẢO TRÌ PHẦN MỀM

Nội dung chính của chương

7.1 Tổng quan về triển khai và bảo trì phần mềm

7.2 Triển khai phần mềm

7.3 Bảo trì phần mềm

7.4 Công cụ, kỹ thuật trợ giúp

Mục tiêu cần đạt được của chương

Áp dụng được các kiến thức vận hành & bảo trì phần mềm trong triển khai các giai đoạn của vòng đời phần mềm

Bài 11: Tổng quan về Triển khai & Bảo trì phần mềm (Số tiết: 03 tiết)

7.1 Giới thiệu về triển khai & bảo trì phần mềm [2]

❖ **Triển khai Phần mềm**

Triển khai phần mềm còn được gọi là triển khai IT . Đây là quá trình lắp/cài đặt thêm các tính năng mới hoặc tích hợp phần mềm mới vào môi trường vận hành của người dùng, nơi các ứng dụng và dịch vụ khác của họ đang vận hành. Để tránh lỗi và các vấn đề xung đột môi trường, việc lập kế hoạch triển khai cần phải được chú trọng. Hình 7.1 là ví dụ minh hoạ việc triển khai không được lập kế hoạch.

Hình 7.1: Minh hoạ kết quả triển khai không được lập kế hoạch

Bên cạnh công việc chuyển giao, cài đặt và thiết lập thông tin cấu hình sản phẩm để đảm bảo hệ thống vận hành thông suốt trên môi trường người dùng; triển khai phần mềm còn bao gồm các hoạt động như:

- Biên dịch và đóng gói phần mềm để sẵn sàng cho hoạt động kiểm thử chấp thuận phần mềm.
- Xây dựng các tài liệu hướng dẫn người dùng.
- Thông báo phát hành/cập nhật sản phẩm.

- Huấn luyện người dùng sử dụng sản phẩm.
- Giám sát, theo dõi quá trình vận hành sản phẩm.
- Định kỳ bảo trì/bảo dưỡng sản phẩm.

Để đánh giá việc triển khai phần mềm thành công, chúng ta có thể dựa vào 4 tiêu chí cơ bản sau:

- Phản hồi tích cực từ phía người dùng cuối về sản phẩm/tính năng mới
- Số lượng khách hàng sử dụng sản phẩm ngày càng tăng
- Giảm chi cho việc hỗ trợ khách hàng sử dụng sản phẩm, bảo trì sản phẩm
- Tăng lợi nhuận từ việc sử dụng sản phẩm, cập nhật tính năng mới cho sản phẩm

❖ **Bảo trì phần mềm**

Theo IEEE (1993), bảo trì phần mềm được định nghĩa như sau: *"Bảo trì là việc sửa đổi phần mềm sau khi phát hành nhằm chỉnh sửa các lỗi phát sinh, cải thiện hiệu năng, các thuộc tính của phần mềm hoặc làm cho phần mềm thích ứng với môi trường vận hành mới".*

Bảo trì phần mềm đóng một vai trò quan trọng trong việc giữ cho phần mềm chạy trơn tru. Ngoài ra, vì đặc tính thay đổi của phần mềm, bảo trì là hoạt động không thể thiếu nhằm duy trì thời gian sống và tính hữu ích của sản phẩm. Bảo trì có mối quan hệ chặt chẽ với hoạt động triển khai phần mềm & nên tiến hành một cách định kỳ nhằm:

- Giảm thiểu thời gian phần mềm hỏng hóc dẫn tới ngưng hoạt động, giảm thiểu chi phí bảo trì.
- Duy trì tính an toàn, an ninh và độ tin cậy của sản phẩm bằng cách khắc phục các lỗ hổng bị xâm nhập khi phần mềm được vận hành trong thời gian dài.
- Loại bỏ những chức năng lỗi thời của sản phẩm, cải thiện hiệu xuất của chương trình.

Lưu ý: Khi quyết định thực hiện bảo trì phần mềm, điều quan trọng là phải xem xét những thay đổi nào có thể xảy ra và ảnh hưởng đến người dùng như thế nào, phản ứng tiềm năng của họ ra sao và các giải pháp để giải quyết vấn đề đó.

Bảo trì có thể được phân loại thành bốn loại:

1) **Bảo trì sửa lỗi (Corrective Maintenance)**: sửa chữa các lỗi, sai sót và khiếm khuyết trong phần mềm; thường xuất hiện dưới dạng các bản cập nhật nhanh, nhỏ & thường xuyên.
2) **Bảo trì thích ứng (Adaptive Maintenance)**: tập trung vào những thay đổi liên quan đến cơ sở hạ tầng phần mềm (hệ điều hành, phần cứng, và nền tảng mới) để giữ cho chương trình tương thích với chúng.
3) **Bảo trì nâng cấp/hoàn thiện (Perfective Maintenance)**: là loại bảo trì chiếm phần lớn kinh phí, thường gấp nhiều lần so với các loại hình bảo trì khác (chiếm

~ 65% chi phí bảo trì), đặc biệt với các phần mềm có tuổi thọ cao. Bảo trì nâng tập trung vào các thay đổi liên quan đến việc nâng cấp, hiệu chỉnh các tính năng & khả năng sử dụng của phần mềm bằng cách tinh chỉnh, xóa, thêm các tính năng mới.

4) **Bảo trì phòng ngừa (Preventative Maintenance)**: trọng tâm của loại bảo trì là ngăn chặn sự xuống cấp của phần mềm khi nó bị thay đổi trong một thời gian dài. Bảo trì phòng ngừa giúp phần mềm trở nên ổn định, dễ hiểu, dễ bảo trì hơn, và duy trì năng lực của nó trong việc chịu đựng các thay đổi theo thời gian. Các hoạt động bảo trì có thể bao gồm: tối ưu hóa mã; cập nhật tài liệu (nếu cần); tái cấu trúc, …Theo một báo cáo được công bố trên transcendent , bất kỳ doanh nghiệp nào cũng có thể tiết kiệm 12-18% bằng cách đầu tư vào bảo trì phòng ngừa thay vì bảo trì theo phản ứng của người dùng.

7.2 Quy trình triển khai phần mềm

Quy trình triển khai phần mềm thường trải qua sáu bước (xem Hình 7.2) như sau:

Bước 1: Lập kế hoạch triển khai

Bước 2: Xác định mục tiêu rõ ràng và thiết kế triển khai

Bước 3: Đào tạo nhóm triển khai phần mềm, đào tạo người dùng sử dụng/cấu hình sản phẩm

Bước 4: Kiểm thử triển khai sau khi khởi chạy hệ thống trên môi trường mô phỏng và lập báo cáo

Bước 5: Bắt đầu triển khai

Bước 6: Người dùng trải nghiệm hệ thống: Nếu khách hàng & người dùng cuối không hưởng lợi được lợi ích gì từ hệ thống/các thay đổi mới, việc triển khai phần mềm xem như là thất bại.

Hình 7.2: Các bước triển khai phần mềm

Một số công cụ hỗ trợ triển khai phần mềm:

- UserGuiding: hướng dẫn người dùng trải nghiệm lần đầu tiên với phần mềm của bạn, giúp họ làm quen bằng cách yêu cầu họ cung cấp các đầu vào trong suốt quá trình vận hành các chức năng của hệ thống và theo dõi các hoạt động vận hành của người dùng
- Flourish giúp việc thu thập và phân tích phản hồi từ người dùng cuối về sản phẩm. Theo dõi việc sử dụng phần mềm của họ để phát hiện sớm các vấn đề về vận hành.
- Optimizely là một công cụ phân tích và thử nghiệm A / B. Cho phép kiểm tra các chức năng của sản phẩm có hoạt động hay không mà không cần phải đợi đến toàn bộ hệ thống khởi chạy; giúp phân tích chức năng, xem cách thức người dùng tương tác với chức năng đó và thực hiện các thay đổi nếu cần. Bằng cách này, nhóm phát triển có thể tự tin vào việc triển khai các chức năng của họ.

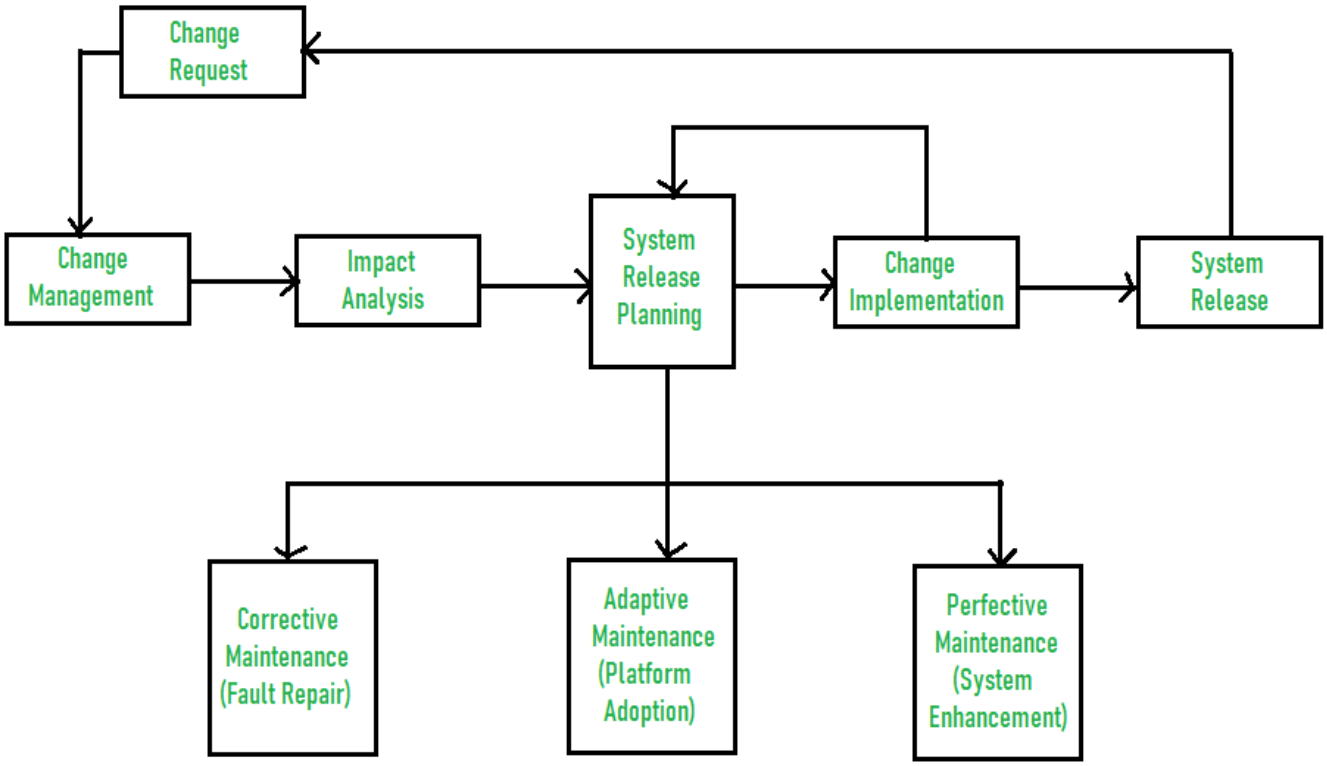
7.3 Quy trình bảo trì phần mềm

Hoạt động bảo trì phần mềm thường được kích hoạt khi có các yêu cầu thay đổi/phản ứng từ phía người dùng sản phẩm, hoặc định kỳ bảo trì dựa trên các kết quả giám sát sự vận hành của phần mềm của người dùng.

Quy trình bảo trì thường được tiến hành theo các bước:

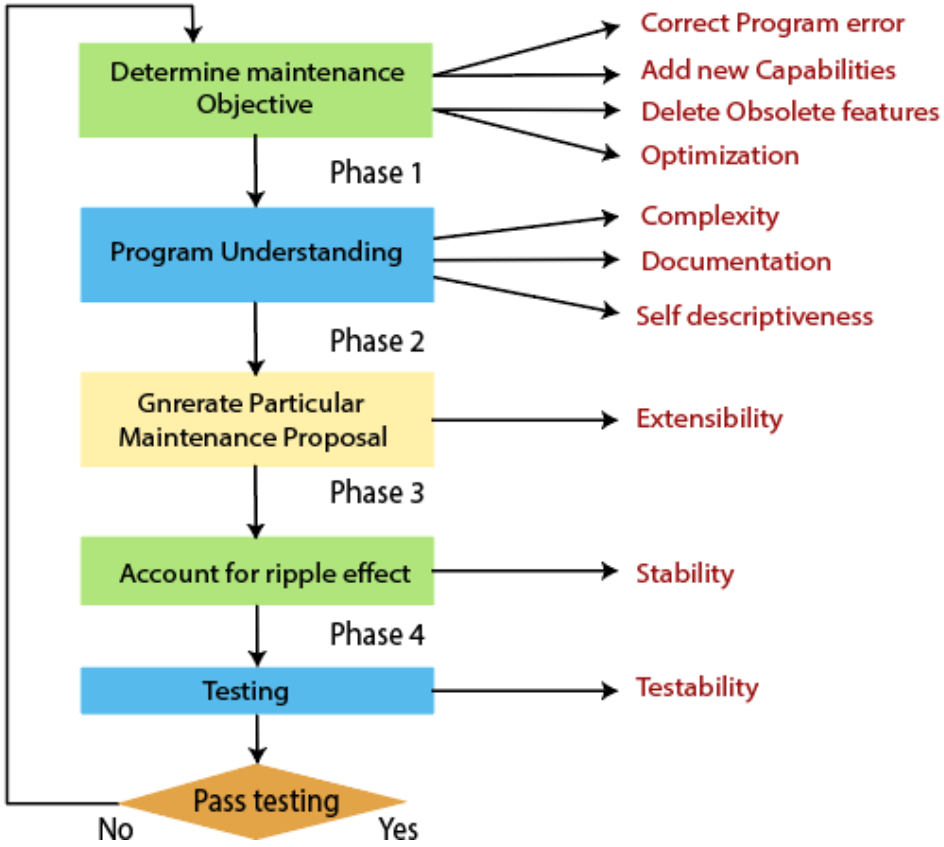
1. Quản lý các yêu cầu thay đổi (Change Request);
2. Phân tích ảnh hương (Inpact Analysis): Phân tích những tác động của sự thay đổi đến những thành phần của hệ thống.
3. Lập kế hoạch phát hành hệ thống (System Release Planning): xác định các mục tiêu bảo trì, xác định loại hình bảo trì (sửa lỗi? thích ứng? hoàn thiện?); hiểu hệ thống; xây dựng giải pháp triển khai; xử lý các hiệu ứng lề; & kế hoạch kiểm thử hệ thống sau khi bảo trì. Quy trình lập kế hoạch được tiến hành theo các bước như Hình 7.4.
4. Triển khai các thay đổi (Change Implementation): thực hiện các thay đổi hệ thống theo kế hoạch đã lập
5. Phát hành hệ thống (System Release): đóng gói, phát hành, triển khai hệ thống/các tính năng được cập nhật đến môi trường vận hành của người dùng.

Chu trình lại tiếp tục cho lần bảo trì tiếp theo (chi tiết, xem Hình 7.3).



Hình 7.3: Quy trình bảo trì phần mềm

Quy trình lập kế hoạch phát hành/bảo trì hệ thống:

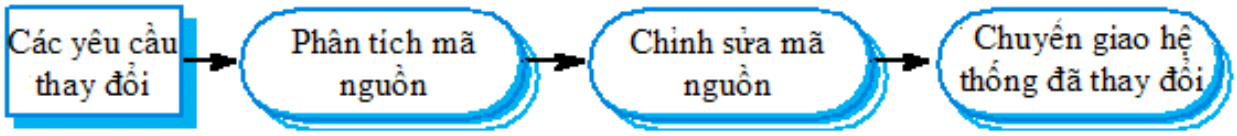


Hình 7.4: Quy trình lập kế hoạch bảo trì phần mềm

Đối với những yêu cầu thay đổi khẩn cấp (hotfixs), quy trình bảo trì khẩn cấp sẽ được áp dụng mà không cần phải trải qua đầy đủ các pha như quy trình bảo trì thông thường. Một số tình huống cần bảo trì khẩn cấp:

- Có lỗi hệ thống nghiêm trọng xảy ra và cần phải sửa chữa ngay.
- Các thay đổi về môi trường gây ra tác động làm phần mềm vận hành không mong đợi.
- Các thay đổi nghiệp vụ yêu cầu phải có đáp ứng nhanh.

Quy trình bảo trì khẩn cấp được chỉ ra trong Hình 7.5 Tuy nhiên, chúng ta nên hạn chế áp dụng quy trình này vì nó có thể dẫn đến phần mềm khó theo dõi, khó hiểu và khó bảo trì trong tương lai.



Hình 7.5: Quy trình cài đặt thay đổi khẩn cấp

Bảo trì phần mềm giữ một vai trò quan trọng trong việc duy trì thời gian sống & tính hữu ích của các phần mềm. DevOps là một giải pháp tối ưu giám sát vận hành; lập kế hoạch bảo trì phần mềm; phát hành và tích hợp ứng dụng liên tục (CI/CD); tăng tính tự động, tốc độ phát hành dịch vụ; đảm bảo chất lượng và sự hài lòng của khách hàng đối với chất lượng dịch vụ bảo trì.

Có nhiều chiến lược bảo trì có thể được áp dụng. Trong các chiến lược bảo trì này thì Tái kỹ nghệ (Re-Engineering) là một trong các chiến lược hiệu quả trong việc bảo trì các hệ thống di sản. Tái kỹ nghệ hệ thống là tiến trình bảo trì hệ thống bởi tiến trình gồm 2 giai đoạn: kỹ nghệ ngược, sau đó là kỹ nghệ tiến nhằm tái sử dụng lại các thành phần của hệ thống di sản và tạo ra hệ thống mới có kiến trúc tốt hơn, chất lượng cao hơn, dễ bảo trì hơn, … với các chức năng không đổi.

7.4 Công cụ, kỹ thuật trợ giúp

Bảo trì phần mềm có thể xem như bao gồm tất cả các hoạt động được tiến hành sau khi phát hành, triển khai phần mềm đến môi trường vận hành của người dùng nhằm duy trì tuổi thọ và tính hữu ích của phần mềm

Các công cụ hỗ trợ bảo trì khá phong phú, ví dụ: hỗ trợ hiểu mã trước khi bảo trì như: gỡ rối code (code debuggers), duyệt code (code browsers); hỗ trợ dịch ngược mã (decompilers); phân tích mã nguồn (code analysis); ghi lại nhật ký vận hành phần mềm (logging) & báo lỗi tự động (ví dụ:Bugzilla để theo dõi lỗi)); kiểm thử hồi quy; và quản lý cấu hình phần mềm. Mục này sẽ tập trung giới thiệu một số công cụ, kỹ thuật hỗ trợ bảo trì gồm:

- Các công cụ kỹ nghệ ngược;
- Chuỗi công cụ hỗ trợ DevOps Framework

7.4.1 Tái kỹ nghệ, kỹ nghệ ngược, kỹ nghệ tiến

Tái kỹ nghệ phần mềm (Software Re-Engineering) là tiến trình cải tiến khả năng bảo trì của sản phẩm bằng cách chuyển dịch nó sang dạng biểu diễn mới trong khi vẫn đảm bảo tính nguyên vẹn của các chức năng phần mềm.

Theo định nghĩa kỹ thuật, tái kỹ nghệ được xem là tiến trình kiểm tra và biến đổi hệ thống để hoàn nguyên nó sang một dạng biểu diễn mới. Tiến trình này nhấn mạnh đến sự kết hợp của các quy trình con như:

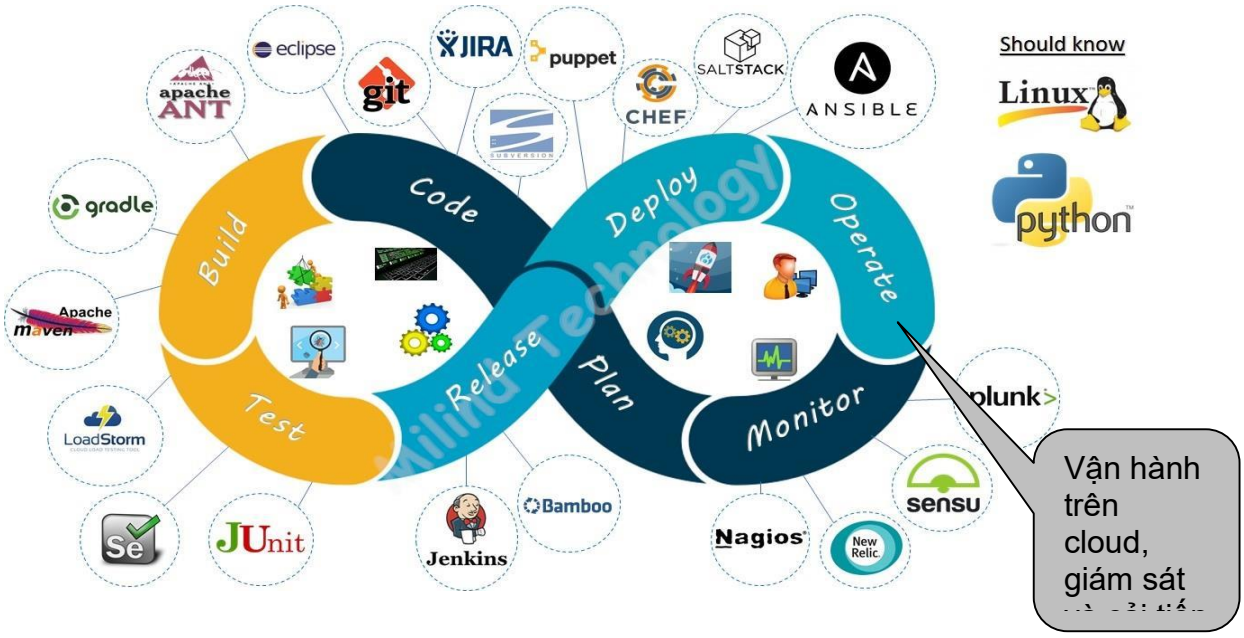
- Kỹ nghệ đảo ngược (Reverse Engineering);
- Tái cấu trúc (Restructing);
- Tái tự liệu (Redocumentation);
- Kỹ nghệ tiến (Forward Engineering); và
- Tái nền tảng vận hành (Retargeting): chuyển dịch phần mềm sang nền tảng hệ thống mới.

Một số công cụ hỗ trợ kỹ nghệ ngược gồm:

- ✓ Từ mã nhị phân (bộ cài đặt phần mềm, gói phần mềm sau khi biên dịch và đóng gói) dịch ngược lại mã nguồn dự án:
  - *Dotpeek (.NET), Hex-Rays (C));*
  - *JD-Eclipse (Java); JDCore service (Java)*
- ✓ Từ mã nguồn sinh ngược lại biểu đồ lớp đối tượng (áp dụng cho Java code)
  - *ObjectAid UML Explorer (plugin vào Eclipse IDE để sử dụng);*
  - *EasyUML (cho NetBeans)*
- ✓ Từ cơ sở dữ liệu quan hệ sinh ngược các lớp thực thể (Entity classes, Java):
  - *JPA/Hibernate Framework*

7.4.2 DevOps Framework – Chuỗi công cụ trợ giúp [17]

Với DevOps, phần mềm thường được phát triển dưới dạng các dịch vụ. DevOps áp dụng các cách tiếp cận thực tế tốt nhất nhằm tự động hoá mọi giai đoạn trong quy trình (xem Hình 7.6).



Hình 7.6: Khung DevOps

Chuỗi các công cụ hỗ trợ DevOps gồm:

- Tools quản lý mã nguồn, xét duyệt mã nguồn tự động: Git, GitHub, Bitbucket, …
- Tools quản lý build (quản lý phụ thuộc, biên dịch và đóng gói): Maven, Ant, Make, MsBuild, …
- Tools kiểm thử tự động: Junit, Selenium, Cucumner, ..
- Tools quản lý kho cấu hình: Nexus, Artifactory, ..,
- Tools tích hợp liên tục (CI): Jenkins, Bamboo, TeamCity, …
- Tools giám sát cấu hình: Chef, Puppet, Ansible, …
- Tools quản lý phát hành: Visusal Studio, StackStorm, ..,
- Dịch vụ kho chứa hosting/đám mây: AWS, Azure, Docker, …
- Tools quản lý triển khai: Rapid Deploy, Code Deploy, …
- Tools hỗ trợ làm việc cộng tác: Jira, Team Foundation, Slack, …
- Tools giám sát vận hành: Kibana, Nagios, …
- Tools theo dõi, loggging và báo lỗi tự động: Splunk, Logstash, …

7.5 Case study: Triển khai, bảo trì phần mềm

a) Mục tiêu

- Xây dựng tài liệu hướng dẫn sử dụng phần mềm

- Lập kế hoạch bảo trì phần mềm

- Lập kế hoạch đào tạo phần mềm

- Hoàn thiện các chức năng trên phần mềm cùng các mẫu biểu giấy tờ để kết thúc dự án.

b) Yêu cầu

1. Xây dựng tài liệu hướng dẫn sử dụng phần mềm

2. Kế hoạch quản lý cấu hình phần mềm.

3. Báo cáo bản kế hoạch bảo trì phần mềm.

4. Báo cáo bản kế hoạch đào tạo phần mềm cho người sử dụng.

5. Hoàn thiện phần mềm với đầy đủ các chức năng cơ bản.

c) Hướng dẫn, gợi ý: [33 – Chapter 6]

Câu hỏi

Câu 1. Bảo trì được phân thành 4 loại nào?

a) Bảo trì sửa lỗi, Bảo trì thích ứng, Bảo trì hoàn thiện, Bảo trì phòng ngừa
b) Bảo trì sửa lỗi, bảo trì bổ sung, bảo trì tích hợp, bảo trì vận hành
c) Bảo trì sửa lỗi, bảo trì bổ sung, Bảo trì để tu chỉnh, Bảo trì để thích nghi
d) Bảo trì để hoàn thiện, Bảo trì để phòng ngừa, Bảo trì sửa lỗi, bảo trì bổ sung

Câu 2. Quy trình cải tiến phần mềm phụ thuộc vào?

a) Kiểu phần mềm cần bảo trì
b) Quy trình phát triển phần mềm đã được sử dụng
c) Kỹ năng và kinh nghiệm của các Stakeholder
d) Cả a,b,c

Câu 3. Quy trình tái kỹ nghệ bao gồm các hoạt động?

e) Dịch mã nguồn, Kỹ nghệ ngược
f) Cải thiện cấu trúc chương trình
g) Mô đun hóa chương trình, tái kỹ nghệ dữ liệu
h) Cả a,b,c

Câu 4. Bảo trì phần mềm là gì?

a) Hoạt động chỉnh sửa phần mềm sau khi nó đã được đưa vào sử dụng
b) Không bao gồm những thay đổi chính liên quan tới kiến trúc của hệ thống
c) Bảo trì là không thể tránh khỏi
d) Cả a, b, c

Câu 5. Tái kỹ nghệ hệ thống là?

a) Kỹ thuật cấu trúc lại (viết lại) một phần hoặc toàn bộ hệ thống cũ được kế thừa mà không thay đổi các chức năng của nó.
b) Kỹ thuật cấu trúc lại (viết lại) một phần hoặc toàn bộ hệ thống cũ được kế thừa mà làm thay đổi các chức năng của nó
c) Kỹ thuật dịch mã nguồn
d) Cả a và c

Bài tập cuối chương:

Bài 7.1. Nêu một số vấn đề thường gặp trong khi bảo trì phần mềm? Mô tả một số giải pháp của bạn để giải quyết các vấn đề này
Bài 7.2. Giải thích các bước bảo trì phần mềm dưới dạng một biểu đồ?
Bài 7.3. Viết một ghi chú ngắn về mô hình Bole và Lehman về việc tính toán nỗ lực bảo trì.
Bài 7.4. Trình bày các mô hình ước lượng chi phí bảo trì phần mềm
Bài 7.5. Cho trước nỗ lực phát triển của một dự án là 600PMs. Hằng số quyết định thực nghiệm (K) của mô hình Belady và Lehman là 0.5. Độ phức tạp mã nguồn là khá cao và bằng 7. Hãy tính tổng nỗ lực (M) nếu nhóm bảo trì có mức độ hiểu dự án trung bình (d=0.7)
Bài 7.6. Cho trước lưu lượng thay đổi hàng năm (ACT – Annual Change Traffic) của một hệ thống phần mềm là 25% /năm. Chi phí phát triển ban đầu của hệ thống 600 PMs. Hãy ước lượng tổng chi phí đã chi cho hoạt động bảo trì hệ thống hàng năm (AME). Nếu tổng thời gian sống của phần mềm là 10 năm thì tổng nỗ lực của dự án là bao nhiêu?
Bài 7.7. Giải thích các loại kỹ thuật tái cấu trúc. Tái cấu trúc trợ giúp như thế nào cho hoạt động bảo trì một chương trình?
Bài 7.8. Các công cụ và kỹ thuật nào là sẵn dùng cho hoạt động bảo trì phần mềm. Trình bày về 2 trong số các công cụ, kỹ thuật này.
Bài 7.9. Tại sao bảo trì phần mềm lại có nhiều thách thức hơn so với phát triển mới sản phẩm? Nêu những đặc điểm mà một bảo trì viên cần có?
Bài 7.10. Nỗ lực phát triển cho một dự án phần mềm là 500 PM (person months). Hằng số quyết định thực nghiệm (K) = 0.3. Độ phức tạp mã nguồn là khá cao, bằng 8. Tính tổng nỗ lực mở rộng (M) nếu:

a. Nhóm bảo trì có mức độ hiểu biết tốt về dự án, d = 0.9
b. Nhóm bảo trì có mức độ hiểu biết nghèo nàn về dự án (d=0.1)

Chương VIII: CÁC XU HƯỚNG MỚI TRONG CÔNG NGHỆ PHẦN MỀM

Nội dung chính của chương

8.1 IOT

8.2 Công nghệ xác thực không dùng mật khẩu

8.3 Công nghệ thực tại ảo (Virtual reality)

8.4 Tự động hóa quy trình bằng Robot

8.5 Công nghệ AI (Artificial Intelligence)

8.6 Hệ thống nhúng

Mục tiêu cần đạt được của chương

Nắm bắt được các xu hướng mới trong lĩnh vực công nghệ phần mềm hiện nay.

Bài 12: Các xu hướng mới trong công nghệ phần mềm (Số tiết: 03 tiết)

8.1 IOT (Internet of Things)

8.1.1 Internet vạn vật (IoT) là gì?

Thuật ngữ IoT hay Internet vạn vật đề cập đến mạng lưới tập hợp các thiết bị thông minh và công nghệ tạo điều kiện thuận lợi cho hoạt động giao tiếp giữa thiết bị và đám mây cũng như giữa các thiết bị với nhau. Nhờ sự ra đời của chip máy tính giá rẻ và công nghệ viễn thông băng thông cao, ngày nay, chúng ta có hàng tỷ thiết bị được kết nối với internet. Điều này nghĩa là các thiết bị hàng ngày như bàn chải đánh răng, máy hút bụi, ô tô và máy móc có thể sử dụng cảm biến để thu thập dữ liệu và phản hồi lại người dùng một cách thông minh [31].

Internet vạn vật tích hợp "vạn vật" với Internet mỗi ngày. Các kỹ sư máy tính đã và đang thêm các cảm biến và bộ xử lý vào các vật dụng hàng ngày kể từ những năm 90. Tuy nhiên, tiến độ ban đầu rất chậm vì các con chip còn to và cồng kềnh. Loại chip máy tính công suất thấp gọi là thẻ tag RFID, lần đầu tiên được sử dụng để theo dõi các thiết bị đắt đỏ. Khi kích cỡ của các thiết bị điện toán dần nhỏ lại, những con chip này cũng trở nên nhỏ hơn, nhanh hơn và thông minh hơn theo thời gian.

Chi phí tích hợp công suất điện toán vào trong các vật dụng nhỏ bé hiện nay đã giảm đáng kể. Ví dụ: bạn có thể thêm khả năng kết nối với các tính năng của dịch vụ giọng thoại Alexa vào các MCU tích hợp sẵn RAM chưa đến 1 MB, chẳng hạn như cho công tắc đèn. Nguyên cả một ngành công nghiệp đã bất ngờ xuất hiện với trọng tâm xoay quanh việc trang bị các thiết bị IoT khắp mọi ngóc ngách căn nhà, doanh nghiệp và văn phòng của chúng ta. Những vật dụng thông minh này có thể tự động truyền và nhận dữ liệu qua Internet. Tất cả những "thiết bị điện toán vô hình" này và công nghệ liên quan được gọi chung là Internet vạn vật.

8.1.2 IoT hoạt động như thế nào?

Một hệ thống IoT thông thường hoạt động thông qua việc thu thập và trao đổi dữ liệu theo thời gian thực. Một hệ thống IoT có ba thành phần:

Thiết bị thông minh

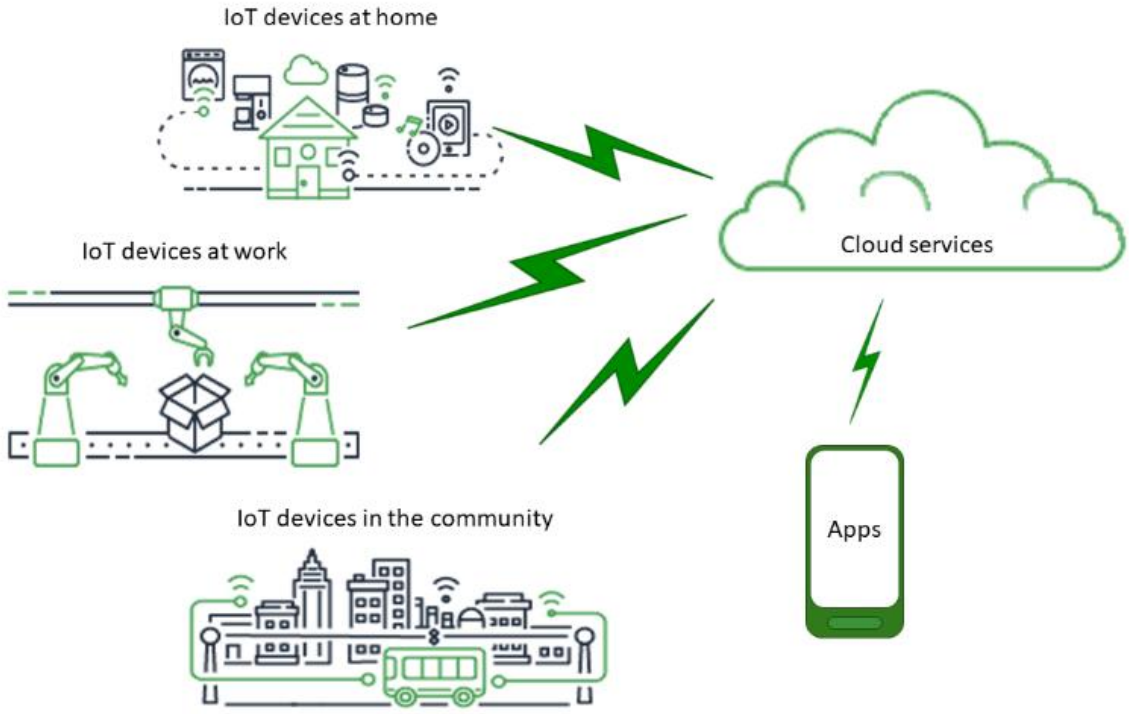
Đây là một thiết bị, giống như tivi, camera an ninh hoặc thiết bị tập thể dục đã được trao cho khả năng điện toán. Thiết bị này thu thập dữ liệu từ môi trường xung quanh, thao tác nhập liệu của người dùng hoặc mô thức sử dụng và truyền cũng như nhận dữ liệu qua Internet từ ứng dụng IoT của nó.

Ứng dụng IoT

Ứng dụng IoT là một tập hợp các dịch vụ và phần mềm có chức năng tích hợp dữ liệu nhận được từ các thiết bị IoT khác nhau. Ứng dụng này sử dụng công nghệ máy học hoặc trí tuệ nhân tạo (AI) để phân tích dữ liệu và đưa ra các quyết định sáng suốt. Những quyết định này được truyền trở lại thiết bị IoT và sau đó, thiết bị IoT đó sẽ phản hồi lại dữ liệu đầu vào một cách thông minh.

Giao diện đồ họa người dùng

Một hoặc một nhóm các thiết bị IoT có thể được quản lý thông qua giao diện đồ họa người dùng. Các ví dụ phổ biến bao gồm một ứng dụng di động hoặc trang web có thể được sử dụng để đăng ký và kiểm soát các thiết bị thông minh.



Hình 8.1: ……

8.1.3 Một vài ví dụ về các thiết bị IoT

Ô tô thông minh

Những phương tiện, chẳng hạn như ô tô, có thể kết nối với Internet bằng rất nhiều cách. Có thể là thông qua camera hành trình thông minh, hệ thống tin học giải trí hoặc thậm chí qua cổng kết nối của phương tiện. Chúng thu thập dữ diệu từ chân ga, phanh, đồng hồ đo tốc độ, đồng hồ đo quãng đường, bánh xe và bình xăng để giám sát cả hiệu suất của người lái và tình trạng phương tiện. Ô tô thông minh được sử dụng cho hàng loạt mục đích:

- ✓ Giám sát đội xe ô tô cho thuê để tăng cường hiệu quả sử dụng nhiên liệu và giảm chi phí.
- ✓ Giúp cha mẹ theo dõi hành vi lái xe của con cái.
- ✓ Tự động thông báo cho bạn bè và người thân trong trường hợp xảy ra tai nạn xe.
- ✓ Dự đoán và hạn chế nhu cầu bảo dưỡng xe.

Nhà thông minh

Các thiết bị gia đình thông minh tập trung chủ yếu vào hoạt động cải thiện hiệu quả và độ an toàn của ngôi nhà, cũng như mạng lưới kết nối trong nhà. Các thiết bị như ổ điện thông minh có thể giám sát mức sử dụng điện và bộ điều nhiệt thông minh có thể cung cấp khả năng kiểm soát nhiệt độ tốt hơn. Các hệ thống thủy canh có thể sử dụng cảm biến IoT để quản lý khu vườn, trong khi đó, máy báo khói IoT có thể phát hiện khói thuốc lá. Các hệ thống an ninh gia đình như khóa cửa, camera an ninh và máy phát hiện rò nước có thể phát hiện và ngăn chặn các mối nguy hiểm, đồng thời gửi cảnh báo tới chủ nhà.

Gia đình có thể sử dụng những thiết bị thông minh cho các mục đích:

- ✓ Tự động tắt các thiết bị khi không sử dụng.
- ✓ Quản lý và bảo trì các bất động sản cho thuê.
- ✓ Tìm đồ thất lạc như chìa khóa hoặc ví.
- ✓ Tự động hóa các công việc hàng ngày như hút bụi, pha cà phê, v.v.

Thành phố thông minh

Các ứng dụng IoT đã giúp quá trình quy hoạch đô thị và bảo trì cơ sở hạ tầng hiệu quả hơn. Các chính phủ đang sử dụng ứng dụng IoT để giải quyết những vấn đề về cơ sở hạ tầng, y tế và môi trường. Ứng dụng IoT có thể được sử dụng cho các mục đích:

- ✓ Đo lường chất lượng không khí và mức độ bức xạ.
- ✓ Giảm chi phí năng lượng nhờ hệ thống chiếu sáng thông minh.
- ✓ Xác định thời điểm cần bảo trì các cơ sở hạ tầng quan trọng như đường xá, cầu cống và đường ống.

- ✓ Tăng lợi nhuận thông qua công tác quản lý bãi đỗ xe hiệu quả.

Công trình thông minh

Các công trình như khuôn viên trường đại học và công trình thương mại sử dụng ứng dụng IoT để thúc đẩy hoạt động hiệu quả hơn. Những công trình thông minh có thể sử dụng các thiết bị IoT cho mục đích:

- ✓ Giảm mức tiêu thụ năng lượng.
- ✓ Giảm chi phí bảo trì.
- ✓ Tận dụng không gian làm việc hiệu quả hơn.

8.1.4 IoT công nghiệp là gì?

IoT công nghiệp (IIoT) đề cập đến các thiết bị thông minh được sử dụng trong sản xuất, bán lẻ, y tế và các lĩnh vực khác để tạo ra hiệu quả kinh doanh. Các thiết bị công nghiệp, từ cảm biến đến máy móc, cung cấp cho chủ doanh nghiệp dữ liệu chi tiết theo thời gian thực để các doanh nghiệp có thể sử dụng phục vụ mục đích cải thiện quá trình kinh doanh. Những thiết bị này cung cấp thông tin chuyên sâu về quản lý chuỗi cung ứng, kho vận, nguồn nhân lực và sản xuất – giảm chi phí và tăng luồng doanh thu.

Các hệ thống công nghiệp thông minh trong các ngành dọc khác nhau:

Sản xuất

IoT doanh nghiệp trong sản xuất sử dụng khả năng bảo trì dự đoán để giảm thời gian ngừng hoạt động ngoài dự kiến và công nghệ đeo trên người để cải thiện an toàn cho người lao động. Các ứng dụng IoT có thể dự đoán lỗi máy móc trước khi nó xảy ra, giúp giảm thời gian ngừng sản xuất. Thiết bị gắn trên mũ bảo hiểm và vòng tay, cũng như các camera thị giác máy tính được sử dụng để cảnh báo người lao động về những mối nguy hiểm tiềm ẩn.

Công nghiệp ô tô

Phân tích dựa theo cảm biến và robot giúp tăng hiệu quả trong sản xuất ô tô và công tác bảo dưỡng. Ví dụ: các cảm biến công nghiệp được sử dụng để cung cấp hình ảnh 3D của các thành phần bên trong phương tiện theo thời gian thực. Việc chẩn đoán và khắc phục sự cố có thể được thực hiện nhanh hơn nhiều trong khi hệ thống IoT tự động đặt hàng các phụ tùng thay thế.

Kho vận và vận tải

Các thiết bị IoT thương mại và công nghiệp có thể hỗ trợ quản lý chuỗi cung ứng, bao gồm quản lý hàng tồn kho, mối quan hệ với nhà cung cấp, quản lý đội xe và bảo trì theo lịch. Các công ty vận chuyển sử dụng ứng dụng IoT công nghiệp để theo dõi tài sản và tối ưu hóa mức tiêu thụ nhiên liệu trên các tuyến vận chuyển. Công nghệ này đặc biệt hữu dụng trong việc kiểm soát chặt chẽ nhiệt độ trong các công-ten-nơ lạnh. Nhờ các

thuật toán định tuyến và tái định tuyến thông minh, các quản lý chuỗi cung ứng có thể đưa ra những dự đoán sáng suốt.

Bán lẻ

Amazon đang thúc đẩy quá trình đổi mới tự động hóa và hợp tác giữa con người và máy móc trong lĩnh vực bán lẻ. Các cơ sở của Amazon sử dụng robot được kết nối Internet để theo dõi, định vị, phân loại và vận chuyển sản phẩm.

8.1.5 IoT có thể cải thiện cuộc sống của chúng ta như thế nào?

Internet vạn vật tác động sâu rộng tới cuộc sống cũng như công việc của con người. IoT cho phép máy móc xử lý phần việc nặng nhọc, đảm nhiệm những nhiệm vụ nhàm chán và giúp cuộc sống trở nên lành mạnh, năng suất và thoải mái hơn.

Ví dụ: các thiết bị thông minh có thể thay đổi hoàn toàn thói quen buổi sáng của bạn. Khi bạn nhấn nút tạm hoãn, chiếc đồng hồ báo thức của bạn sẽ tự động bật máy pha cà phê và kéo mở rèm cửa sổ. Tủ lạnh của bạn sẽ tự động phát hiện những thực phẩm sắp hết và đặt mua giao tận nhà. Lò nướng thông minh sẽ cho bạn biết thực đơn trong ngày và thậm chí còn nấu những nguyên liệu đã được chuẩn bị sẵn để đảm bảo rằng bữa trưa của bạn đã sẵn sàng. Chiếc đồng hồ thông minh sẽ lên lịch họp cho bạn, trong khi đó, chiếc ô tô thông minh của bạn tự động đặt vị trí GPS dừng xe để nạp nhiên liệu. Tiềm năng là vô hạn trong thế giới IoT!

8.1.6 IoT mang tới cho doanh nghiệp những lợi ích gì?

Tăng tốc độ đổi mới: Internet vạn vật mang tới cho các doanh nghiệp khả năng tiếp cận với những phân tích nâng cao để khám phá các cơ hội mới. Ví dụ: các doanh nghiệp có thể tạo ra những chiến dịch tiếp thị nhắm mục tiêu chuẩn xác bằng cách thu thập dữ liệu về hành vi của khách hàng.

Chuyển đổi dữ liệu thành thông tin chuyên sâu và hành động bằng AI và ML: Dữ liệu và xu hướng trong quá khứ đã thu thập có thể được sử dụng để dự đoán kết quả trong tương lai. Ví dụ: thông tin bảo hành có thể được ghép cặp với dữ liệu do IoT thu thập để dự đoán các sự cố bảo trì. Lợi thế này có thể được sử dụng để chủ động cung cấp dịch vụ khách hàng cũng như xây dựng lòng trung thành của khách.

Tăng tính bảo mật: Việc liên tục giám sát cơ sở hạ tầng kỹ thuật số cũng như vật lý có thể tối ưu hóa hiệu suất, cải thiện mức độ hiệu quả và giảm bớt rủi ro an toàn. Ví dụ: dữ liệu được thu thập từ một thiết bị giám sát tại chỗ có thể kết hợp với dữ liệu phần cứng và phiên bản firmware để tự động lên lịch cập nhật hệ thống.

Thay đổi quy mô các giải pháp khác biệt: Công nghệ IoT có thể được triển khai theo hướng tập trung vào khách hàng để cải thiện mức độ hài lòng. Ví dụ: các sản phẩm bán chạy có thể được bổ sung kịp thời để tránh tình trạng thiếu hụt hàng hóa.

8.1.7 Các công nghệ IoT là gì?

Các công nghệ được sử dụng trong hệ thống IoT có thể bao gồm:

❖ Điện toán biên

Điện toán biên đề cập đến công nghệ được sử dụng để điều khiển các thiết bị thông minh thực hiện nhiều tác vụ hơn, không chỉ đơn thuần là gửi hay nhận dữ liệu từ nền tảng IoT của chúng. Công nghệ này tăng cường công suất điện toán tại biên của một mạng lưới IoT, giảm bớt độ trễ trong giao tiếp và cải thiện tốc độ phản hồi.

❖ Điện toán đám mây

Công nghệ đám mây được sử dụng để lưu trữ dữ liệu từ xa và quản lý thiết bị IoT, giúp nhiều thiết bị trong mạng lưới có thể truy cập dữ liệu.

❖ Máy học

Máy học đề cập đến phần mềm và thuật toán được sử dụng để xử lý dữ liệu và đưa ra các quyết định theo thời gian thực dựa trên dữ liệu đó. Những thuật toán máy học này có thể được triển khai trên đám mây hoặc tại biên.

8.2 Công nghệ xác thực không dùng mật khẩu *(Passwordless Authentication)*

8.2.1 Công nghệ xác thực không dùng mật khẩu là gì?

Xác thực không dùng mật khẩu là quá trình xác minh danh tính của người dùng bằng một yếu tố khác không phải là mật khẩu. Phương pháp này tăng cường tính an toàn bằng cách loại bỏ các hoạt động quản lý mật khẩu và nguy cơ từ nhiều mối đe dọa. Đây là một lĩnh vực mới trong quản lý định danh và truy cập.

Đánh cắp thông tin xác thực và lạm dụng mật khẩu ngày càng tiếp tục gia tăng. Do đó, các tổ chức đang hướng tới phương pháp xác thực không dùng mật khẩu để giải quyết khiếm khuyết của mật khẩu và bảo vệ quyền truy cập vào dữ liệu, hệ thống và mạng.

8.2.2 Các phương pháp xác thực không dùng mật khẩu

Xác thực không dùng mật khẩu là phương pháp tiếp cận tương đối mới. Có nhiều cách khác nhau để triển khai xác thực không dùng mật khẩu, bao gồm:

1. Xác thực sinh trắc học: Tính năng này sử dụng các đặc điểm cơ thể độc nhất để xác minh người dùng mà không yêu cầu mật khẩu.

2. Liên kết ma thuật (magic link): Phương pháp này yêu cầu địa chỉ email của người dùng trong cửa sổ đăng nhập. Người dùng nhận được một liên kết trong email của họ và khi nhấp vào sẽ có thể đăng nhập vào tài khoản. Liên kết hết hạn trong vòng vài giờ và người dùng nhận được một liên kết mới mỗi khi họ đăng nhập.

3. Thông báo đẩy (push notification): người dùng nhận được thông báo đẩy thông qua ứng dụng riêng của dịch vụ hoặc xác thực chuyên dụng trên thiết bị di động. Theo đó, nhà cung cấp dịch vụ sẽ gửi thông báo cho người dùng qua kênh liên lạc an toàn nhất hiện có. Người dùng trả lời xác thực bằng cách thực hiện một hành động đối với thông báo đẩy để xác minh danh tính của họ và sau đó có thể truy cập dịch vụ. Một số ứng dụng xác thực chuyên dụng trên thiết bị di động bao gồm: ESET Secure Authentication, Rapid Identity,...

4. Mật khẩu dùng một lần (OTP): Phương pháp này yêu cầu người dùng nhập mã nhận được qua email hoặc tin nhắn văn bản. Mã được gửi mỗi khi người dùng đăng nhập.

5. Đăng nhập một lần (Single-Sign-On) Một phương pháp xác thực không dùng mật khẩu hiệu quả khác là đăng nhập một lần. Phương pháp này cho phép nhân viên truy cập vào tất cả các tài khoản của họ mà không cần tạo hoặc ghi nhớ các mật khẩu phức tạp.

8.2.3 Nguyên lý hoạt động của xác thực không mật khẩu

Xác thực không mật khẩu là một loại xác thực đa yếu tố (MFA), nhưng thay thế mật khẩu bằng một yếu tố xác thực bảo mật hơn, chẳng hạn như sinh trắc học vân tay hoặc mã PIN.

Xác thực không mật khẩu hoạt động dựa trên nguyên lý giống như chứng thư số, đó là sử dụng một cặp khóa mã hóa bất đối xứng bao gồm một khóa riêng tư (Private key) và một khóa công khai (Public key). Cặp khóa Private-Public key được tạo ra khi đăng ký một tài khoản để đăng nhập ứng dụng. Khóa Private key được tạo ra và lưu trữ trực tiếp trên thiết bị của người dùng và không thể can thiệp để đọc hay truy xuất được thông tin này ra ngoài. Khóa Private key thường được bảo vệ và chỉ được truy xuất với một yếu tố xác thực bổ sung, chẳng hạn như vân tay, mã PIN hoặc nhận dạng giọng nói. Trong khi đó khóa Public key được cung cấp đến các ứng dụng, website, trình duyệt hoặc các hệ thống trực truyến mà người dùng muốn tạo tài khoản để đăng nhập không mật khẩu vào ứng dụng đó thay vì sử dụng tên đăng nhập/mật khẩu truyền thống.

8.2.4 Những lý do để sử dụng xác thực không mật khẩu

Những cuộc tấn công mạng liên tiếp trong thập kỷ qua đã chỉ ra rằng tên đăng nhập/mật khẩu (được phát minh từ năm 1964) đã không còn đủ bảo mật. Theo Báo cáo điều tra về lộ lọt dữ liệu năm 2019 của Verizon, hơn 80% các vụ tấn công lộ lọt dữ liệu xuất phát từ thông tin tài khoản bị đánh cắp. Việc sử dụng xác thực không mật khẩu chính là xu hướng của tương lai vì:

- Đơn giản và thuận tiện cho người dùng: Người dùng thường khó khăn trong việc ghi nhớ các mật khẩu phức tạp, chưa kể một người dùng phải ghi nhớ nhiều mật khẩu để truy cập đến các ứng dụng khác nhau. Kết quả là người dùng thường sử dụng các mật khẩu yếu, dễ nhớ, sử dụng cùng 01 mật khẩu để truy cập nhiều ứng dụng hoặc

chọn cách ghi mật khẩu ra giấy. Với xác thực không mật khẩu, người dùng không sử dụng mật khẩu để đăng nhập vào ứng dụng, do đó dễ dàng và thuận tiện hơn khi truy cập vào các ứng dụng, đồng thời giảm thiểu các nguy cơ an ninh mạng trong việc quản lý và sử dụng mật khẩu truyền thống.

- Ngăn chặn tấn công lừa đảo (Phishing): Sử dụng thông tin tên đăng nhập/mật khẩu để truy cập các ứng dụng, người dùng có thể bị tấn công lừa đảo để cung cấp thông tin tài khoản như kích vào đường dẫn, tệp đính kèm độc hại trong Email… Với xác thực không mật khẩu, không có thông tin đăng nhập để lừa đảo và đánh cắp.

- Ngăn chặn tấn công bằng mã độc Keylogger: Bằng phương thức tấn công Keylogger, hacker thu thập thông tin tài khoản của người dùng thông qua những gì mà người dùng gõ trên máy tính, thiết bị của họ. Với xác thực không mật khẩu, người dùng không cần phải nhập thông tin tài khoản để đăng nhập ứng dụng, do đó hacker không thể thu thập thông tin đăng nhập bằng hình thức tấn công này.

- Ngăn chặn tấn công sử dụng lại thông tin xác thực (Credential stuffing attacks): Hầu hết mọi người sử dụng cùng một mật khẩu cho nhiều tài khoản. Do đó hacker có thể sử dụng mật khẩu đã đánh cắp này để truy cập các tài khoản khác. Với xác thực không mật khẩu, không có mật khẩu để hacker đánh cắp và sử dụng lại.

- Ngăn chặn tấn công nghe lén trên môi trường mạng (Network sniffing attacks): Mật khẩu có thể dễ dàng bị hacker đánh cắp bằng cách bắt và phân tích lưu lượng mạng dưới dạng không mã hóa (cleartext) hoặc lưu lượng mã hóa không đủ mạnh có thể bị hacker bẻ gãy. Với xác thực không mật khẩu, thông tin đăng nhập không được truyền trên môi trường mạng, do đó loại trừ rủi ro liên quan đến tấn công nghe lén trên môi trường mạng.

- Ngăn chặn tấn công dò quét mật khẩu: Đây là hình thức tấn công sử dụng các chương trình, phần mềm với cơ sở dữ liệu dưới dạng từ điển mật khẩu để dò quét, khai thác các mật khẩu yếu hoặc mật khẩu mặc định của thiết bị, như 123456, admin, root,…

- Giảm công việc cho đội ngũ IT Helpdesk/Support: Với việc sử dụng tên đăng nhập/mật khẩu, người dùng thường hay quên hoặc nhập sai mật khẩu nhiều lần liên tiếp dẫn đến khóa tài khoản. Theo thống kê, khoảng 30%-50% các yêu cầu hỗ trợ liên quan đến khôi phục mật khẩu và mở khóa tài khoản bị khóa dẫn đến mất thời gian, giảm hiệu năng làm việc của đội ngũ IT.

- Xác thực hai hoặc đa yếu tố (MFA – MultiFactor Authentication) là phương pháp xác thực ngoài tên đăng nhập/mật khẩu còn sử dụng một yếu tố khác để định danh người dùng như sinh trắc học vân tay, khuôn mặt, mộng mắt, mã OTP, mã gửi về SMS, Email... Phương thức này bảo mật hơn so với phương thức xác thực sử dụng tên đăng nhập/mật khẩu truyền thống, tuy nhiên lại gây ra sự phức tạp và bất tiện cho người dùng.

Xác thực không mật khẩu giải quyết đồng thời cả 2 vấn đề về bảo mật và trải nghiệm của người dùng.

8.2.5 Những thách thức đối với phương pháp xác thực không dùng mật khẩu

Không dễ dàng triển khai

Xác thực không dùng mật khẩu dường như là một cách tiếp cận an toàn và dễ sử dụng, tuy nhiên lại có những thách thức trong việc triển khai nó. Vấn đề quan trọng nhất là ngân sách và sự phức tạp để chuyển đổi. Trong việc thiết lập ngân sách cho xác thực không dùng mật khẩu, doanh nghiệp nên tính toán cả chi phí mua phần cứng cũng như chi phí thiết lập và cấu hình.

Doanh nghiệp nên thực hiện xác thực không dùng mật khẩu dựa trên khóa công khai với các mức độ bảo vệ khác nhau. Họ cần sử dụng cả môđun mật mã dựa trên phần cứng và phần mềm ở phía máy khách. Tuy nhiên, đây sẽ là một thách thức thực sự đối với bất kỳ công ty phát triển phần mềm nào. Một bước triển khai sai có thể khiến tổ chức dễ bị tấn công chuỗi cung ứng phần mềm.

Khó khắc phục sự cố

Do phương pháp xác thực không dùng mật khẩu vẫn còn chưa quen thuộc với nhiều người, nên người dùng có thể gặp phải các vấn đề: Người dùng có thể gặp trục trặc khi muốn đăng nhập vào một thiết bị khác. Hơn nữa, nếu một người làm mất thiết bị đã xác thực của họ, có thể việc khắc phục sự cố và lấy lại tài khoản của họ sẽ mất thời gian. Do đó, một công ty sẽ cần một đội ngũ hỗ trợ có kinh nghiệm từ công ty quản lý định danh và truy cập để giúp đỡ khi các vấn đề như vậy phát sinh.

Tăng chi phí

Mặc dù nhiều doanh nghiệp có thể tiết kiệm được chi phí với xác thực không dùng mật khẩu, nhưng chi phí cài đặt của phương pháp xác thực này ban đầu có thể tốn kém. Một doanh nghiệp cần đầu tư ban đầu dựa trên hình thức triển khai mà doanh nghiệp đó mong muốn. Ví dụ, đối với phương pháp xác thực dựa trên điện thoại thông minh, doanh nghiệp sẽ cần phải xem xét chi phí phát triển để đảm bảo rằng nó được thực hiện trơn tru. Tuy nhiên, sau khi triển khai sẽ không phát sinh thêm chi phí nào khác.

Một số điện thoại thông minh không hỗ trợ sinh trắc học

Khi sử dụng sinh trắc học cho phương pháp xác thực không dùng mật khẩu, người dùng phải có điện thoại thông minh có máy quét. Đối với những người dùng sử dụng điện thoại thông minh không hỗ trợ sinh trắc học thì việc triển khai là không thể. Đây có thể là một hạn chế đáng kể nếu doanh nghiệp đang cung cấp dịch vụ thông qua ứng dụng dựa trên điện thoại thông minh.

8.2.6 Các giải pháp xác thực không mật khẩu trên thị trường

Hiện nay, công nghệ xác thực không mật khẩu được nghiên cứu và phát triển rộng rãi trên thế giới. Các giải pháp này có thể dựa trên nền tảng, phần mềm hoặc phần cứng khác nhau. Trong các hãng công nghệ lớn cung cấp giải pháp này có thể kể đến Microsoft với nền tảng Windows Hello for Business cho PC (hỗ trợ Windows 10, version 1511 trở lên và Azure AD), ứng dụng Microsoft Authenticator cho di động (thiết bị chạy iOS và Android từ version 6.0 trở lên) và phần cứng cho các khóa bảo mật tuân thủ theo chuẩn FIDO2 (hỗ trợ Windows 10, version 1809 trở lên và Azure AD). FIDO2 là một tập hợp các tiêu chuẩn mới do Liên minh Xác thực trực tuyến thế giới (FIDO Alliance) định nghĩa. Các dịch vụ tuân theo tiêu chuẩn này cho phép người dùng có thể dễ dàng xác thực các dịch vụ trực tuyến trên các môi trường khác nhau mà không cần sử dụng mật khẩu. Tham khảo đường dẫn sau để có thêm thông tin chi tiết: https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RE2KEup

Tại thị trường Việt Nam, vừa qua Công ty TNHH Dịch vụ và An ninh mạng VinCSS (Tập đoàn Vingroup) đã phát triển hệ sinh thái xác thực không mật khẩu VinCSS FIDO2 đạt chuẩn quốc tế. Thiết bị khóa bảo mật VinCSS FIDO2 Authenticator hoạt động được trên các hệ điều hành Windows, macOS, iOS, iPadOS, Android, nền tảng trình duyệt Microsoft Edge, Mozilla Firefox, Google Chrome, Apple Safari và tài khoản trực tuyến Microsoft, Google, Facebook, Dropbox, Twitter. Thông tin chi tiết tham khảo đường dẫn sau: https://passwordless.vincss.net/.

8.3 Công nghệ thực tại ảo (Virtual reality) [32]

8.3.1 Thực tại ảo (Virtual reality - VR) là gì?

Thực tại ảo (Virtual reality - VR) là thuật ngữ mô tả môi trường mô phỏng bằng máy tính, hình ảnh hiển thị trên màn hình thông qua kính nhìn ba chiều, cùng với các giác quan khác như âm thanh, xúc giác... để tạo ra một thế giới "như thật". Thế giới "nhân tạo" này lại phản ứng, thay đổi theo ý muốn (tín hiệu vào) của người sử dụng (hành động, lời nói...).

Đặc tính quan trọng nhất của công nghệ thực tế ảo là tương tác thời gian thực, tức là khả năng nhận biết thế giới thực gần trùng khít với thế giới ảo.

Hệ thống thực tế ảo gồm 5 thành phần: phần mềm, phần cứng, mạng liên kết, người dùng, các ứng dụng. Trong đó, phần mềm, phần cứng và các ứng dụng là quan trọng nhất.

- Phần mềm: Phần mềm luôn là linh hồn của VR cũng như đối với bất cứ một hệ thống máy tính hiện đại nào. Về mặt nguyên tắc có thể dùng bất cứ ngôn ngữ lập trình hay phần mềm đồ họa nào để mô hình hóa (modelling) và mô phỏng (simulation) các đối tượng của VR. Ví dụ như các ngôn ngữ OpenGL, C++,

Java3D, VRML, X3D,…hay các phần mềm thương mại như WorldToolKit, PeopleShop,… Phần mềm của bất kỳ VR nào cũng phải bảo đảm 2 công dụng chính: Tạo hình vào Mô phỏng. Các đối tượng của VR được mô hình hóa nhờ chính phần mềm này hay chuyển sang từ các mô hình 3D (thiết kế nhờ các phần mềm CAD khác như AutoCAD, 3D Studio,..). Sau đó phần mềm VR phải có khả năng mô phỏng động học, động lực học, và mô phỏng ứng xử của đối tượng.

- Phần cứng**:** Phần cứng của một hệ thống VR bao gồm: Máy tính (PC hay Workstation với cấu hình đồ họa mạnh), các thiết bị đầu vào (Input devices) và các thiết bị đầu ra (Output devices).

  + Các thiết bị đầu vào (Input devices): Chúng bao gồm những thiết bị đầu ra có khả năng kích thích các giác quan để tạo nên cảm giác về sự hiện hữu trong thế giới ảo. Chẳng hạn như màn hình đội đầu hiển thị HMD (Head mounted display), chuột, các tai nghe âm thanh nổi – và những thiết bị đầu vào có khả năng ghi nhận nơi người sử dụng đang nhìn vào hoặc hướng đang chỉ tới, như thiết bị theo dõi gắn trên đầu (head-trackers), găng tay hữu tuyến (wire-gloves).

  + Các thiết bị đầu ra (Output devices): gồm hiển thị đồ họa (như màn hình, HDM,..) để nhìn được đối tượng 3D. Thiết bị âm thanh (loa) để nghe được âm thanh vòm (như Hi-Fi, Surround,..). Bộ phản hồi cảm giác (Haptic feedback như găng tay,..) để tạo xúc giác khi sờ, nắm đối tượng. Bộ phản hồi xung lực (Force Feedback) để tạo lực tác động như khi đạp xe, đi đường xóc,…

Về cơ bản, VR có 3 đặc tính chính là Tương tác (Interactive), Nhập vai (Immersion) và Tưởng tượng (Imagination). Một hệ thống thực tế ảo thì tính tương tác, các đồ họa ba chiều thời gian thực và cảm giác đắm chìm được xem là các đặc tính then chốt.

- **Tương tác thời gian thực (real-time interactivity):** có nghĩa là máy tính có khả năng nhận biết được tín hiệu vào của người sử dụng và thay đổi ngay lập tức thế giới ảo. Người sử dụng nhìn thấy sự vật thay đổi trên màn hình ngay theo ý muốn của họ và bị thu hút bởi sự mô phỏng này.
- **Cảm giác đắm chìm:** là một hiệu ứng tạo khả năng tập trung sự chú ý cao nhất một cách có chọn lọc vào chính những thông tin từ người sử dụng hệ thống thực tế ảo. Người sử dụng cảm thấy mình là một phần của thế giới ảo, hòa lẫn vào thế giới đó. VR còn đẩy cảm giác này "thật" hơn nữa nhờ tác động lên các kênh cảm giác khác. Người dùng không những nhìn thấy đối tượng đồ họa 3D, điều khiển (xoay, di chuyển..) được đối tượng mà còn sờ và cảm thấy chúng như có thật. Các nhà nghiên cứu cũng đang tìm cách tạo những cảm giác khác như ngửi, nếm trong thế giới ảo.

- **Tính tương tác:** có hai khía cạnh của tính tương tác trong một thế giới ảo: sự du hành bên trong thế giới và động lực học của môi trường. Sự du hành là khả năng của người dùng để di chuyển khắp nơi một cách độc lập, cứ như là đang ở bên trong một môi trường thật. Nhà phát triển phần mềm có thể thiết lập những áp đặt đối với việc truy cập vào những khu vực ảo nhất định, cho phép có được nhiều mức độ tự do khác nhau (Người sử dụng có thể bay, xuyên tường, đi lại khắp nơi hoặc bơi lặn…). Một khía cạnh khác của sự du hành là sự định vị điểm nhìn của người dùng. Sự kiểm soát điểm nhìn là việc người sử dụng tự theo dõi chính họ từ một khoảng cách, việc quan sát cảnh tượng thông qua đôi mắt của một con người khác, hoặc di chuyển khắp trong thiết kế của một cao ốc mới như thể đang ngồi trong một chiếc ghế đẩy… Động lực học của môi trường là những quy tắc về cách thức mà người, vật và mọi thứ tương tác với nhau trong một trật tự để trao đổi năng lượng hoặc thông tin.

8.3.2 Một số công nghệ chính của thực tại ảo

Hiện tại đang có 03 hướng công nghệ chính: thực tế ảo (VR), thực tế ảo tăng cường (AR) và thực tế hỗn hợp (MR)

Hình 8.2: Một số công nghệ chính của thực tại ảo

8.3.2.1 Thực tế ảo (VR)?

Thực tế ảo (Virtual Reality – VR) là một trải nghiệm mô phỏng không gian, sự vật, sự việc có thể giống hoặc khác hoàn toàn với thế giới thực, nói cách khác là tạo ra một môi trường giả lập. Môi trường này được con người thiết kế qua các ứng dụng phần mềm chuyên dụng, và hiển thị trên màn hình máy tính hoặc qua kính thực tế ảo (Oculus Rift, HTC Vive, Playstation VR…). Trong thế giới đó, bạn có thể điều khiển và di chuyển đồ vật xung quanh bằng bộ điều khiển xúc

8.3.2.2 Thực tế ảo tăng cường (AR – Augumented Reality) khác gì so với thực tế ảo (VR)?

Hình 8.3: Công nghệ thực tế ảo tăng cường trong game pokemon

Trong khi VR đưa người dùng đến với một thế giới ảo tách biệt hoàn toàn với thực tại, thì công nghệ AR – thực tế ảo tăng cường lại giúp đưa các hình ảnh đồ họa (thông tin kỹ thuật số) vào trong thế giới thực để bạn có thể nhìn bằng mắt thường. Trò chơi *Pokémon Go*là một trong những ví dụ nổi tiếng nhất cho công nghệ này. Thực tế ảo tăng cường (AR) coi thế giới thực là trung tâm nhưng tăng cường thêm các chi tiết kỹ thuật số khác,chồng thêm các tầng nhận biết và bổ sung thực tế hoặc môi trường của bạn.Tóm lại, trong khi VR thay thế hoàn toàn thực tại bởi một thế giới mô phỏng, AR chỉ bổ sung các chi tiết vào thế giới thực tại.

Hình 8.4: AR bổ sung các thông tin số vào thực tại ảo

8.3.2.3 Thực tế hỗn hợp (Mixed Reality – MR) là sự kết hợp của VR và AR

Thực tế hỗn hợp (Mixed Reality – MR) nằm ở đâu đó ở giữa VR và AR. Nó pha trộn giữa thế giới thực và thế giới ảo để tạo ra các môi trường phức tạp – nơi các yếu tố vật lý và kỹ thuật số có thể tương tác với nhau trong thời gian thực.

Giống như VR, những nội dung kỹ thuật số (ảo) có tính tương tác và người dùng có thể thao tác với các đối tượng ảo này trong không gian vật lý (thực) của họ. Và giống như AR, nó có thể mang đến nhiều nội dung kỹ thuật số lên môi trường thế giới thực.

8.3.2.4 Thực tế mở rộng (Exetened Reality – XR) công nghê của tương lai

Thực tế mở rộng (Extended Reality – XR) là thuật ngữ bao trùm lên cả VR, AR và MR, cũng như tất cả các "thực tế" trong tương lai mà công nghệ có thể mang lại. XR là giao điểm của các loại công nghệ và cách chúng sẽ phối hợp với nhau để phá vỡ các nhiệm vụ hàng ngày của chúng ta.

8.3.3 ứng dụng của thực tại ảo

Từ lĩnh vực trò chơi điện tử đến điện ảnh hay y học, Thực tế ảo (VR), Thực tế ảo tăng cường (AR) và Thực tế hỗn hợp (MR) được sử dụng ngày càng nhiều.

- Y tế: Để đào tạo, chẳng hạn như mô phỏng các ca phẫu thuật
- Điện ảnh và truyền hình: Để tạo ra các trải nghiệm độc đáo trong các phim và chương trình thực tế
- Du lịch ảo: Để du lịch ảo đến viện bảo tàng nghệ thuật hoặc hành tinh khác tại chính nhà bạn
- Thể thao chuyên nghiệp: Dành cho các chương trình huấn luyện như STRIVR để hỗ trợ các vận động viên chuyên nghiệp và nghiệp dư
- Chơi game: Dành cho hơn 1.000 game hiện có, từ trò chơi nhập vai bắn súng hay thám hiểm đến trò chơi chiến lược

Hình 8.5:Một số úng dụng của thực tại ảo trong các ngành nghề

8.3.4 lập trình VR

Lập trình VR hay thực tế ảo đang hiện diện nhiều hơn trong các lĩnh vực cuộc sống hiện nay. Lập trình VR và Game VR là hướng đi mới cho lập trình viên.

Các chương trình thực tế ảo có thể chạy trên các loại kính VR cardboad đơn giản, như video, các game đơn giản. Trên kính cao cấp hơn thì chương trình chạy trên đó cũng phong phú hơn: như game VR, video VR360, các ứng dụng VR cho du lịch, nội thất, dạy nghề, giáo dục ,bất động sản, huấn luyện … Như vậy để tạo ra 1 sản phẩm VR chính là tạo ra các game, video, ứng dụng đó để đưa vào các thiết bị kính thực tế ảo sao cho nó chạy được để người khác đeo kính vào có thể chơi, sử dụng.

8.3.4.1 Các bước lập trình VR để tạo ta một sản phẩm thực tế ảo

- Đầu tiên là lên ý tưởng, nội dung sẽ làm 1 sản phẩm thực tế ảo: Xác định làm game VR hay ứng dụng VR, làm cho lĩnh vực gì, nội dung sẽ chạy như thế nào, bối cảnh, đồ vật, các môi trường tương tác trong sản phẩm VR sẽ gồm những gì, càng chi tiết càng tốt.
- Xác định loại kính thực tế ảo VR nào sẽ dùng để triển khai cài đặt game hay ứng dụng đó lên: Việc này quan trọng để sẵn sàng cho việc thiết kế, dựng 3D, bối cảnh, nhân vật phù hợp, và sau đó các lập trình viên sẽ lập trình VR sao cho tương thích thiết bị kính đã chọn đó để triển khai khả thi nhất.
- Tiếp đến là thiết kế dựng bối, cảnh nhận vật, không gian, đồ đạc, đối tượng trong game theo kịch bản game VR đã lên.
- Sau khi công đoạn thiết kế tương đối ổn thì chuyển sang lập trình VR, nó là lập trình hoàn thiện game VR hoặc ứng dụng VR đó thành một sản phẩm hoàn chỉnh.
- Trong quá trình làm trên phải test các chức năng, bối cảnh, hành động của game, sản phẩm đạt tiêu chuẩn theo kịch bản game.
- Sau khi test xong sản phẩm bản cuối cùng của game Ok thì chuyển qua cài đặt, tích hợp lên loại kính VR đã lựa chọn.
- Sau đó là hướng dẫn sử dụng, vận hành, nâng cấp, phát hành chuyển giao.

8.3.4.2 Thiết kế và lập trình là trọng tâm của vấn đề?

Để tạo thành game thì việc thiết kế và lập trình vẫn là trọng tâm của việc tạo ra sản phẩm VR. Vậy chúng ta hãy tìm hiểu các công cụ thông dụng nào đang được dùng, đang thịnh hành phục vụ cho 2 việc đó. Việc tạo ra các sản phẩm đơn giản có vẻ chỉ cần thể là đủ, nhưng các ứng dụng phức tạp thì cần nhiều hơn thế, nó còn là giải pháp tích hợp, triển khai và các vấn đề khác. Nhưng hãy tập trung vào các vấn đề tối thiêu trước như sau.

a) Thiết kế cho lập trình VR:

Thiết kế là môt bước đầu rất quan trọng tạo ra phần mềm thực tế ảo. Về Thiết kế đồ họa, đối tượng, nhân vật: Hiện nay thường dùng các công cụ thiết kế 3D chuyên dụng có sẵn như 3DMAX, SKETCHUP, MAYA, BLENDER, SOLIDWORK… để tạo ra các mô hình 3D phù hợp và đầy đủ theo kịch bản

b) Lập trình VR cho game:

Tiếp theo là việc lập trình, hiện nay các game VR và AR nói chung đang sử dụng 2 nên tảng phổ biến để lập trình game và các ứng dụng thực tế ảo đó là Unity và Unreal. Cả 2 Engine này rất mạnh mẽ trong việc lập trình, xử lý các môi trường, đối tượng 3D rất tốt, và hầu như các game, ứng dụng 3D, hay các phần mềm thực tế ảo, thực tế tăng cường AR hiện nay đều tạo ra chủ yếu bằng 2 engine này.

Lập trình với Unity 3D Engine sẽ hỗ trợ rất tốt cho mảng game 2D, 3D, thực tế là platform hỗ trợ trên cả các môi trường đa nền tảng rất tốt như window, IOS, android.. với ngôn ngữ lập trình phổ biến.

Lập trình VR với Unreal: Tương tự như Unity thì Unreal cũng được nhiều lập trình viên lựa chọn cũng vì sự tiện lợi như trên, hỗ trợ đa nền tảng và ngôn ngữ lập trình phổ biến.

c) So sánh unity và unreal:

Việc chọn lựa Unreal và Unity làm engine cho sản phẩm của mình – lập trình VR còn phụ thuộc vào các yêu cầu, kịch bản, và các đặc trưng chi tiết sản phẩm để lựa chọn cái nào phù hợp nhất.

Các sản phẩm thực thế ảo được ứng dụng ngày càng phong phú trong nhiều lĩnh vực khác nhau, việc chọn giải pháp phù hợp thực hiện, giải quyết một vấn đề nào đó dựa vào công nghệ này được xem như một giải pháp mới. Sự thành công hay thất bại phụ thuộc vào các làm để lôi cuốn được lượng người dùng đông nhất và sự hài lòng vượt trội so với các công nghệ truyền thống khác thì vẫn chờ đợi các lập trình viên và các công ty chuyên làm trong lĩnh vực này.

8.4 Tự động hóa quy trình bằng Robot (Robotic Process Automation)

8.4.1 Khái niệm về tự động hóa quy trình

RPA hiểu đơn giản là một con Robot xử lý công việc tự động trên máy tính. Quy trình tự động hóa RPA(Robotic Process Automation) là công nghệ mới nổi gần đây dựa trên robot software và AI. Tự động hóa quy trình là một chuỗi các hành động diễn ra tự động, tạo ra sự xuyên suốt trong quá trình quản lý, kinh doanh của công việc. Từ đó đảm bảo mọi luồng công việc đều diễn ra một cách thuận lợi, trôi chảy. Đây được xem là bước tiến hoàn toàn mới của công nghệ, giúp doanh nghiệp tháo gỡ toàn bộ nút thắt thường gặp trong quá trình triển khai quy trình quản lý và điều hành.

RPA có thể làm việc với các phần mềm khác như Excel, phần mềm kế toán, ngân hàng… RPA ứng dụng trí thông minh để thực hiện lượng công việc lớn, lặp lại theo chu kỳ. Điển hình như: Nhập liệu, tạo đơn hàng, cấp quyền truy cập; các công việc đòi hỏi sự liên tục giao tiếp với nhiều hệ thống khác nhau.

8.4.2 Tính năng nổi bật của Robotic Process Automation

- Phần mềm có giao diện thân thiện, dễ dàng thao tác và sử dụng. Thiết kế quy trình tự động hóa phù hợp cả với những cá nhân không có kiến thức lập trình.
- RPA được tích hợp đầy đủ các tính năng hỗ trợ người dùng như: thiết kế quy trình – lập lịch giám sát, theo dõi – thực thi quy trình được thiết lập.
- Có thể quản lý từ xa nhiều bot trong cùng một thời điểm, theo dõi theo thời gian thực hiện.
- Độ bảo mật cao, có thể để ẩn trong PC giúp người dùng vừa có thể làm việc, vừa chạy bot đồng thời.
- Người dùng có thể tương tác, trao đổi với nhau thông qua chatbot.
- Kết hợp với một số giải pháp số hóa khác như: xác minh hóa đơn điện tử, nhận diện văn bản điện tử, hỗ trợ AI thu thập và phân tích dữ liệu…

8.4.3 Ứng dụng của giải pháp tự động hoá quy trình RPA trong từng lĩnh vực cụ thể

- ***Human Resources (HR):*** Giúp quản trị nhân sự tự động hóa các quy trình tuyển dụng, trả lương, yêu cầu đào tạo, xử lý đơn xin việc, nghỉ việc và thai sản,...
- ***Sản xuất & phân phối dược phẩm:*** Kiểm soát chất lượng trong hoạt động kiểm tra nguyên liệu, sản xuất thuốc, đóng gói, kiểm tra sản phẩm sau sản xuất và chuẩn bị xuất xưởng.
- ***Dịch vụ:*** Quy trình chăm sóc khách hàng trước, trong và sau bán hàng, quy trình xử lý khiếu nại,…
- ***Du lịch:*** Quản lý các chuyến bay, đặt phòng khách sạn, chuyến tham quan và đặt chỗ thuê ô tô của khách hàng cho đại lý du lịch.
- ***Sức khỏe:*** Tự động hóa quản lý việc nhận đơn thuốc từ bác sĩ thông qua quá trình lấy thuốc và đến tay bệnh nhân.
- ***Công nghệ thông tin:*** Chỉ định cách giải quyết cho từng loại vấn đề phần mềm/ phần cứng cụ thể mà nhân viên gọi đến.
- ***Các CIO tập đoàn:*** Tổng hợp dữ liệu thông tin, báo cáo từ nhiều nguồn hàng ngày, hàng tuần..
- ***Ngân hàng, tổ chức tài chính:*** Hợp nhất hệ thống quản lý dữ liệu nhiều hệ thống quản lý dữ liệu. Đơn giản hóa yêu cầu xử lý lượng thông tin liên tục và lặp đi lặp lại…

- ***Các bộ phận sắm, cung ứng, chăm sóc khách hàng:*** Giải quyết công việc, báo cáo trên nền Office lặp đi lặp lại, theo một quy tắc nhất định trên phần mềm.

8.4.4 Lợi ích của RPA

Tiết kiệm thời gian, chi phí

Người dùng có thể gửi mail, cập nhật báo cáo, thao tác toàn bộ tác vụ ngay trên phần mềm. Theo đó, có thể thực hiện các bước tiếp theo hoặc quay lại dễ dàng. Tiết kiệm tối đa thời gian, cải thiện được hiệu suất làm việc.

Thay vì phải gửi email, cập nhật báo cáo tại nhiều công cụ để chuyển tiếp cho cá nhân khác, toàn bộ tác vụ đều được tự động hóa trên phần mềm. Tiết kiệm thời gian cho các hoạt động thủ công, cải thiện hiệu suất làm việc

Linh hoạt, cải thiện năng suất

Phần mềm quản lý RPA đều có thể làm việc được mọi lúc mọi nơi trên tất cả các thiết bị. Điều này cho phép nhân viên làm việc bên ngoài nhưng vẫn có thể tương tác, trao đổi công việc qua di động. Như vậy, mọi công việc sẽ không bị gián đoạn bởi khoảng cách, không gian.

Thay vì lập nhiều nhóm chat, tạo nhiều file dự án, email, nhà quản lý có thể thiết lập nhiều quy trình khác nhau trong một hệ thống trên phần mềm. Dễ dàng phân bổ, kiểm soát và đánh giá chất lượng công việc từng cá nhân, phòng ban trên một giao diện duy nhất.

Đảm bảo độ chính xác cao

Một doanh nghiệp hoạt động không thể tránh khỏi sai sót. Thế nhưng chủ động trong việc xác định đúng lỗi và ngăn ngừa chúng ngay từ đầu là điều hoàn toàn có thể làm được.

Với hệ thống quản lý quy trình làm việc, lãnh đạo sẽ dễ dàng nhận thấy được vấn đề mà doanh nghiệp đang gặp phải. Từ đó có thể đưa ra phương án xử lý kịp thời, đó cũng là bài học để không xảy ra lỗi sai tương tự.

Nâng cao tinh thần trách nhiệm

Khi áp dụng công cụ tự động hóa quản lý trong doanh nghiệp sẽ bắt buộc tất cả nhân viên phải làm theo một quy trình thống nhất. Điều này đem đến sự minh bạch, phân định rõ trách nhiệm của từng cá nhân đối với công việc được phân công…

Nhân viên sẽ không thể đưa ra lý do "không nhận được email" hay "không đọc email" để rũ bỏ trách nhiệm đối với công việc được lãnh đạo giao phó. Vì chỉ cần nhìn vào quy trình, người quản lý sẽ biết được ai phải làm việc gì, tiến độ công việc trong từng giai đoạn.

Lưu trữ và xử lý dữ liệu tốt

Tự động hóa quản lý bằng các phần mềm RPA giúp các Giám đốc dễ dàng hơn trong việc quản lý công ty. Đặc biệt là tránh nguy cơ lộ thông tin, mất dữ liệu.

Nếu việc sao lưu dữ liệu ra ổ cứng, sao lưu ra nhiều máy tính khác nhau gây mất thời gian, tốt kém thì việc quản lý bằng phần mềm sẽ giúp doanh nghiệp giải quyết được vấn đề này. Bởi mọi dữ liệu đều được lưu trữ trên đám mây, không bị ảnh hưởng của việc mất máy tính, nhiễm virus hay các trở ngại khác,…

Quản lý đa nhiệm vụ, đa phòng ban chính xác, tức thời là đặc điểm nổi bật phần mềm tự động hóa quy trình RPA.

8.5 Công nghệ AI (Artificial Intelligence)

8.5.1. Công nghệ AI là gì?

AI (viết tắt của Artifical Intelligence - Trí Thông Minh Nhân Tạo) là công nghệ mô phỏng các quá trình suy nghĩ và học tập của con người cho máy móc, đặc biệt là các hệ thống máy tính. Các quá trình này bao gồm việc học tập (thu thập thông tin và các quy tắc sử dụng thông tin), lập luận (sử dụng các quy tắc để đạt được kết luận gần đúng hoặc xác định), và tự sửa lỗi. Các ứng dụng đặc biệt của AI bao gồm các hệ thống chuyên gia, nhận dạng tiếng nói và thị giác máy tính (nhận diện khuôn mặt, vật thể hoặc chữ viết).

Khái niệm về công nghệ AI xuất hiện đầu tiên bởi John McCarthy (một nhà khoa học máy tính Mỹ) vào năm 1956 tại hội nghị The Dartmouth. Ngày nay, công nghệ AI là một thuật ngữ bao gồm tất cả mọi thứ từ quá trình tự động hoá robot đến người máy thực tế.

Công nghệ AI gần đây trở nên nổi tiếng, nhận được sự quan tâm của nhiều người là nhờ Big Data. Mối quan tâm của các doanh nghiệp về tầm quan trọng của dữ liệu cùng với công nghệ phần cứng đã phát triển mạnh mẽ hơn, cho phép xử lý công nghệ AI với tốc độ nhanh hơn bao giờ hết.

Công nghệ AI đã được áp dụng vào một loạt các lĩnh vực trong thực tế như: chăm sóc sức khỏe, kinh doanh, giáo dục, tài chính, pháp luật, ngân hàng, các ngành sản xuất, đặc biệt là AI trong bảo mật.

Hiện nay, AI được ứng dụng rất nhiều trên các sản phẩm công nghệ tiên tiến như: Điện thoại, các thiết bị âm thanh và các thiết bị điện gia dụng.

Công nghệ AI trên điện thoại giúp cải thiện đáng kể khả năng chụp hình, xử lý hình ảnh, tiết kiệm điện năng, nâng cao bảo mật,… Ngoài ra AI còn được tích hợp trên các vi xử lí giúp điện thoại có thể xử lý những thuật toán phức tạp, tăng cường hiệu năng, khả năng chụp ảnh, xử lí đồ họa cũng như mang lại nhiều tính năng độc đáo trên điện thoại.

Các dòng smartphone hiện nay đang được trang bị rất nhiều tính năng thú vị được hỗ trợ bởi trí tuệ nhân tạo AI, có thể kể tới như Google Translate, các trợ lí ảo như Google Assistant, Siri, Bixby,…

Đi kèm với những lợi ích ủa AI, cũng có những mặt trái mà chúng ta cần quan tâm như chi phí cao, không có tính linh hoạt, sáng tạo và đặc biệt là khả năng gây ra tình trạng thất nghiệp hàng loạt cho người lao động.

8.5.2. Phân loại công nghệ AI

Dựa vào mức độ phức tạp của công nghệ AI, có thể chia nó thành 4 loại sau đây:

Loại 1: Công nghệ AI phản ứng (Reactive Machine)

Một trong những thành công đầu tiên có thể kể đến trong lĩnh vực nghiên cứu AI là chương trình Deep Blue do IBM tạo ra, chương trình đã đánh bại kỳ thủ cờ vua Garry Kasparov bằng cách xác định và dự đoán những nước đi của đối thủ, từ đó lập luận để đưa ra những bước đi phù hợp nhất. Tuy nhiên do những hạn chế về công nghệ của những năm 90 mà Deep Blue của IBM không có ký ức cũng như không thể sử dụng những kinh nghiệm trong quá khứ trong tương lai để tiếp tục phát triển. Tuy nhiên đây cũng được xem là một thành công lớn trong lĩnh vực nghiên cứu về AI của IBM.

Khái niệm về AI phản ứng là việc máy có thể phân tích những động thái khả thi nhất của mình và của đối thủ, đưa ra giải pháp tối ưu nhất.

Bên cạnh đó, một sản phẩm khác của "gã khổng lồ" Google là AlphaGO được thiết kế để chơi cờ vây, tuy nhiên nó còn tồn tại những hạn chế giống như Deep Blue, đây chỉ là những nghiên cứu ban đầu về AI nên còn nhiều hạn chế và không thể áp dụng rộng rãi.

Hình 8.6: AlphaGO – Chương trình "chơi cờ vây" của Google

Loại 2: Công nghệ AI với bộ nhớ hạn chế

Đây được xem là một trong những thành công lớn khi ứng dụng thành công AI trong một số lĩnh vực và sản phẩm công nghệ khác như xe không người lái, máy bay drone hoặc những tàu ngầm hiện đại. Công nghệ AI này khắc phục được những nhược điểm của của AI phản ứng, chúng có thể sử dụng những kinh nghiệm trong quá khứ để đưa ra những quyết định trong tương lai. **Công nghệ AI** này thường được kết hợp với nhiều cảm biến môi trường xung quanh để dự đoán những tình huống có thể xảy ra và đưa ra những quyết định tốt nhất cho thiết bị. Vì dụ như các xe hơi không người lái, chúng được phát triển với nhiều cảm biến xung quanh xe, một cảm biến ở đầu xe có thể tính toán được khoảng cách của xe với xe phía trước, AI sẽ dự đoán nguy có thể xảy ra va chạm, từ đó điều chính tốc độ xe để đảm bảo an toàn và tránh gây tai nạn giao thông.

Hình 8.7: AI được phát triển và ứng dụng trên các phương tiện không người lái

Loại 3: Lý thuyết về trí tuệ nhân tạo

Đây là một thuật ngữ tâm lý. Công nghệ AI này có thể tự mình suy nghĩ và học hỏi những thứ xung quanh để áp dụng cho chính bản thân nó lên một việc cụ thể. Loại công nghệ AI này chưa khả thi trong thời gian hiện tại.

Loại 4: Tự nhận thức

Lúc này cả hệ thống AI có ý thức về bản thân, có ý thức và hành xử như con người. Chúng thậm chí còn có cảm xúc và hiểu được cảm xúc của những người khác. Tất nhiên, loại công nghệ AI này vẫn chưa khả thi.

Một trong các ứng dụng rõ rệt và có hiệu quả nhất của AI là nhà thông minh. Bằng việc kết nối nhiều loại sản phẩm có khả năng học hỏi thói quen của chủ sở hữu nhờ vào trợ lí ảo như Google Assistant, lúc này AI sẽ tận dụng mọi thông tin mà nó ghi nhớ được từ chủ nhân để phục vụ các nhu cầu được đưa ra một cách nhanh chóng và thuận tiện nhất.

8.6 Hệ thống nhúng

Hệ thống nhúng (*embedded system*) là một thuật ngữ để chỉ một hệ thống có khả năng tự trị được nhúng vào trong một môi trường hay một hệ thống mẹ. Đó là các hệ thống tích hợp cả phần cứng và phần mềm phục vụ các bài toán chuyên dụng trong nhiều lĩnh vực công nghiệp, tự động hoá điều khiển, quan trắc và truyền tin. Đặc điểm của các hệ thống nhúng là hoạt động ổn định và có tính năng tự động hoá cao.

Hệ thống nhúng thường được thiết kế để thực hiện một chức năng chuyên biệt nào đó. Khác với các máy tính đa chức năng, chẳng hạn như máy tính cá nhân, một hệ thống nhúng chỉ thực hiện một hoặc một vài chức năng nhất định, thường đi kèm với những yêu cầu cụ thể và bao gồm một số thiết bị máy móc và phần cứng chuyên dụng mà ta không tìm thấy trong một máy tính đa năng nói chung. Vì hệ thống chỉ được xây dựng cho một số nhiệm vụ nhất định nên các nhà thiết kế có thể tối ưu hóa nó nhằm giảm thiểu kích thước và chi phí sản xuất. Các hệ thống nhúng thường được sản xuất hàng loạt với số lượng lớn. Hệ thống nhúng rất đa dạng, phong phú về chủng loại. Đó có thể là những thiết bị cầm tay nhỏ gọn như đồng hồ kĩ thuật số và máy chơi nhạc MP3, hoặc những sản phẩm lớn như đèn giao thông, bộ kiểm soát trong nhà máy hoặc hệ thống kiểm soát các máy năng lượng hạt nhân. Xét về độ phức tạp, hệ thống nhúng có thể rất đơn giản với một vi điều khiển hoặc rất phức tạp với nhiều đơn vị, các thiết bị ngoại vi và mạng lưới được nằm gọn trong một lớp vỏ máy lớn.

Các thiết bị PDA hoặc máy tính cầm tay cũng có một số đặc điểm tương tự với hệ thống nhúng như các hệ điều hành hoặc vi xử lý điều khiển chúng nhưng các thiết bị này không phải là hệ thống nhúng thật sự bởi chúng là các thiết bị đa năng, cho phép sử dụng nhiều ứng dụng và kết nối đến nhiều thiết bị ngoại vi.

Hệ thống nhúng thường có một số đặc điểm chung như sau:

- ✓ Các hệ thống nhúng được thiết kế để thực hiện một số nhiệm vụ chuyên dụng chứ không phải đóng vai trò là các hệ thống máy tính đa chức năng. Một số hệ thống đòi hỏi ràng buộc về tính hoạt động thời gian thực để đảm bảo độ an toàn và tính ứng dụng; một số hệ thống không đòi hỏi hoặc ràng buộc chặt chẽ, cho phép đơn giản hóa hệ thống phần cứng để giảm thiểu chi phí sản xuất.
- ✓ Một hệ thống nhúng thường không phải là một khối riêng biệt mà là một hệ thống phức tạp nằm trong thiết bị mà nó điều khiển.
- ✓ Phần mềm được viết cho các hệ thống nhúng được gọi là firmware và được lưu trữ trong các chip bộ nhớ ROM hoặc bộ nhớ flash chứ không phải là trong một ổ đĩa. Phần mềm thường chạy với số tài nguyên phần cứng hạn chế: không có bàn phím, màn hình hoặc có nhưng với kích thước nhỏ, dung lượng bộ nhớ thấp.

Sau đây, ta sẽ đi sâu, xem xét cụ thể đặc điểm của các thành phần của hệ thống nhúng.

- Giao diện

Các hệ thống nhúng có thể không có giao diện (đối với những hệ thống đơn nhiệm) hoặc có đầy đủ giao diện giao tiếp với người dùng tương tự như các hệ điều hành trong các thiết bị để bàn. Đối với các hệ thống đơn giản, thiết bị nhúng sử dụng nút bấm, đèn LED và hiển thị chữ cỡ nhỏ hoặc chỉ hiển thị số, thường đi kèm với một hệ thống menu đơn giản.

Còn trong một hệ thống phức tạp hơn, một màn hình đồ họa, cảm ứng hoặc có các nút bấm ở lề màn hình cho phép thực hiện các thao tác phức tạp mà tối thiểu hóa được khoảng không gian cần sử dụng; ý nghĩa của các nút bấm có thể thay đổi theo màn hình và các lựa chọn. Các hệ thống nhúng thường có một màn hình với một nút bấm dạng cần điều khiển (joystick button). Sự phát triển mạnh mẽ của mạng toàn cầu đã mang đến cho những nhà thiết kế hệ nhúng một lựa chọn mới là sử dụng một giao diện web thông qua việc kết nối mạng. Điều này có thể giúp tránh được chi phí cho những màn hình phức tạp nhưng đồng thời vẫn cung cấp khả năng hiển thị và nhập liệu phức tạp khi cần đến, thông qua một máy tính khác. Điều này là hết sức hữu dụng đối với các thiết bị điều khiển từ xa, cài đặt vĩnh viễn. Ví dụ, các router là các thiết bị đã ứng dụng tiện ích này.

- Kiến trúc CPU

Các bộ xử lý trong hệ thống nhúng có thể được chia thành hai loại: vi xử lý và vi điều khiển. Các vi điều khiển thường có các thiết bị ngoại vi được tích hợp trên chip nhằm giảm kích thước của hệ thống. Có rất nhiều loại kiến trúc CPU được sử dụng trong thiết kế hệ nhúng như ARM, MIPS, Coldfire/68k, PowerPC, x86, PIC, 8051, Atmel AVR, Renesas H8, SH, V850, FR-V, M32R, Z80, Z8 … Điều này trái ngược với các loại máy tính để bàn, thường bị hạn chế với một vài kiến trúc máy tính nhất định. Các hệ thống nhúng có kích thước nhỏ và được thiết kế để hoạt động trong môi trường công nghiệp thường lựa chọn PC/104 và PC/104++ làm nền tảng. Những hệ thống này thường sử dụng DOS, Linux, NetBSD hoặc các hệ điều hành nhúng thời gian thực như QNX hay VxWorks. Còn các hệ thống nhúng có kích thước rất lớn thường sử dụng một cấu hình thông dụng là hệ thống on chip (System on a chip – SoC), một bảng mạch tích hợp cho một ứng dụng cụ thể (an application-specific integrated circuit – ASIC). Sau đó nhân CPU được mua và thêm vào như một phần của thiết kế chip. Một chiến lược tương tự là sử dụng FPGA (field-programmable gate array) và lập trình cho nó với những thành phần nguyên lý thiết kế bao gồm cả CPU.

- Thiết bị ngoại vi

Hệ thống nhúng giao tiếp với bên ngoài thông qua các thiết bị ngoại vi, ví dụ như:

- Serial Communication Interfaces (SCI): RS-232, RS-422, RS-485...

- Synchronous Serial Communication Interface: I2C, JTAG, SPI, SSC và ESSI
- Universal Serial Bus (USB)
- Networks: Controller Area Network, LonWorks...
- Bộ định thời: PLL(s), Capture/Compare và Time Processing Units
- Discrete IO: General Purpose Input/Output (GPIO)

➢ Công cụ phát triển

Tương tự như các sản phẩm phần mềm khác, phần mềm hệ thống nhúng cũng được phát triển nhờ việc sử dụng các trình biên dịch (compilers), chương trình dịch hợp ngữ (assembler) hoặc các công cụ gỡ rối (debuggers). Tuy nhiên, các nhà thiết kế hệ thống nhúng có thể sử dụng một số công cụ chuyên dụng như:

- Bộ gỡ rối mạch hoặc các chương trình mô phỏng (emulator)
- Tiện ích để thêm các giá trị checksum hoặc CRC vào chương trình, giúp hệ thống nhúng có thể kiểm tra tính hợp lệ của chương trình đó.
- Đối với các hệ thống xử lý tín hiệu số, người phát triển hệ thống có thể sử dụng phần mềm workbench như MathCad hoặc Mathematica để mô phỏng các phép toán.
- Các trình biên dịch và trình liên kết (linker) chuyên dụng được sử dụng để tối ưu hóa một thiết bị phần cứng.
- Một hệ thống nhúng có thể có ngôn ngữ lập trình và công cụ thiết kế riêng của nó hoặc sử dụng và cải tiến từ một ngôn ngữ đã có sẵn.

Các công cụ phần mềm có thể được tạo ra bởi các công ty phần mềm chuyên dụng về hệ thống nhúng hoặc chuyển đổi từ các công cụ phát triển phần mềm GNU. Đôi khi, các công cụ phát triển dành cho máy tính cá nhân cũng được sử dụng nếu bộ xử lý của hệ thống nhúng đó gần giống với bộ xử lý của một máy PC thông dụng.

➢ Độ tin cậy

Các hệ thống nhúng thường nằm trong các cỗ máy được kỳ vọng là sẽ chạy hàng năm trời liên tục mà không bị lỗi hoặc có thể khôi phục hệ thống khi gặp lỗi. Vì thế, các phần mềm hệ thống nhúng được phát triển và kiểm thử một cách cẩn thận hơn là phần mềm cho máy tính cá nhân. Ngoài ra, các thiết bị rời không đáng tin cậy như ổ đĩa, công tắc hoặc nút bấm thường bị hạn chế sử dụng. Việc khôi phục hệ thống khi gặp lỗi có thể được thực hiện bằng cách sử dụng các kỹ thuật như watchdog timer – nếu phần mềm không đều đặn nhận được các tín hiệu watchdog định kì thì hệ thống sẽ bị khởi động lại.

Một số vấn đề cụ thể về độ tin cậy như:

- Hệ thống không thể ngừng để sửa chữa một cách an toàn, ví dụ như ở các hệ thống không gian, hệ thống dây cáp dưới đáy biển, các đèn hiệu dẫn đường,…

Giải pháp đưa ra là chuyển sang sử dụng các hệ thống con dự trữ hoặc các phần mềm cung cấp một phần chức năng.

- Hệ thống phải được chạy liên tục vì tính an toàn, ví dụ như các thiết bị dẫn đường máy bay, thiết bị kiểm soát độ an toàn trong các nhà máy hóa chất,… Giải pháp đưa ra là lựa chọn backup hệ thống.
- Nếu hệ thống ngừng hoạt động sẽ gây tổn thất rất nhiều tiền của ví dụ như các dịch vụ buôn bán tự động, hệ thống chuyển tiền, hệ thống kiểm soát trong các nhà máy …

Các kiến trúc phần mềm hệ thống nhúng

Một số loại kiến trúc phần mềm thông dụng trong các hệ thống nhúng như sau:

- Vòng lặp kiểm soát đơn giản

Theo thiết kế này, phần mềm được tổ chức thành một vòng lặp đơn giản. Vòng lặp gọi đến các chương trình con, mỗi chương trình con quản lý một phần của hệ thống phần cứng hoặc phần mềm.

- Hệ thống ngắt điều khiển

Các hệ thống nhúng thường được điều khiển bằng các ngắt. Có nghĩa là các tác vụ của hệ thống nhúng được kích hoạt bởi các loại sự kiện khác nhau. Ví dụ, một ngắt có thể được sinh ra bởi một bộ định thời sau một chu kỳ được định nghĩa trước, hoặc bởi sự kiện khi cổng nối tiếp nhận được một byte nào đó.

Loại kiến trúc này thường được sử dụng trong các hệ thống có bộ quản lý sự kiện đơn giản, ngắn gọn và cần độ trễ thấp. Hệ thống này thường thực hiện một tác vụ đơn giản trong một vòng lặp chính. Đôi khi, các tác vụ phức tạp hơn sẽ được thêm vào một cấu trúc hàng đợi trong bộ quản lý ngắt để được vòng lặp xử lý sau đó. Lúc này, hệ thống gần giống với kiểu nhân đa nhiệm với các tiến trình rời rạc.

- Đa nhiệm tương tác

Một hệ thống đa nhiệm không ưu tiên cũng gần giống với kỹ thuật vòng lặp kiểm soát đơn giản ngoại trừ việc vòng lặp này được ẩn giấu thông qua một giao diện lập trình API. Các nhà lập trình định nghĩa một loạt các nhiệm vụ, mỗi nhiệm vụ chạy trong một môi trường riêng của nó. Khi không cần thực hiện nhiệm vụ đó thì nó gọi đến các tiến trình con tạm nghỉ (bằng cách gọi "pause", "wait", "yield" …).

Ưu điểm và nhược điểm của loại kiến trúc này cũng giống với kiểm vòng lặp kiểm soát đơn giản. Tuy nhiên, việc thêm một phần mềm mới được thực hiện dễ dàng hơn bằng cách lập trình một tác vụ mới hoặc thêm vào hàng đợi thông dịch (queue-interpreter).

- Đa nhiệm ưu tiên

Ở loại kiến trúc này, hệ thống thường có một đoạn mã ở mức thấp thực hiện việc chuyển đổi giữa các tác vụ khác nhau thông qua một bộ định thời. Đoạn mã này thường nằm ở mức mà hệ thống được coi là có một hệ điều hành và vì thế cũng gặp phải tất cả những phức tạp trong việc quản lý đa nhiệm.

Bất kỳ tác vụ nào có thể phá hủy dữ liệu của một tác vụ khác đều cần phải được tách biệt một cách chính xác. Việc truy cập tới các dữ liệu chia sẻ có thể được quản lý bằng một số kỹ thuật đồng bộ hóa như hàng đợi thông điệp (message queues), các phương thức truyền tin thị giác (semaphores)… Bởi những phức tạp nói trên, giải pháp thường được đưa ra đó là sử dụng một hệ điều hành thời gian thực. Lúc đó, các nhà lập trình có thể tập trung vào việc phát triển các chức năng của thiết bị chứ không cần quan tâm đến các dịch vụ của hệ điều hành nữa.

- Vi nhân (Microkernel) và nhân ngoại (Exokernel)

Khái niệm vi nhân (microkernel) là một bước tiếp cận gần hơn tới khái niệm hệ điều hành thời gian thực. Lúc này, nhân hệ điều hành thực hiện việc cấp phát bộ nhớ và chuyển CPU cho các luồng thực thi. Còn các tiến trình người dùng sử dụng các chức năng chính như hệ thống file, giao diện mạng lưới,… Nói chung, kiến trúc này thường được áp dụng trong các hệ thống mà việc chuyển đổi và giao tiếp giữa các tác vụ là nhanh.

Còn nhân ngoại (exokernel) tiến hành giao tiếp hiệu quả bằng cách sử dụng các lời gọi chương trình con thông thường. Phần cứng và toàn bộ phần mềm trong hệ thống luôn đáp ứng và có thể được mở rộng bởi các ứng dụng.

- Nhân khối (monolithic kernels)

Trong kiến trúc này, một nhân đầy đủ với các khả năng phức tạp được chuyển đổi để phù hợp với môi trường nhúng. Điều này giúp các nhà lập trình có được một môi trường giống với hệ điều hành trong các máy để bàn như Linux hay Microsoft Windows và vì thế rất thuận lợi cho việc phát triển. Tuy nhiên, nó lại đòi hỏi đáng kể các tài nguyên phần cứng làm tăng chi phí của hệ thống. Một số loại nhân khối thông dụng là Embedded Linux và Windows CE. Mặc dù chi phí phần cứng tăng lên nhưng loại hệ thống nhúng này đang tăng trưởng rất mạnh, đặc biệt là trong các thiết bị nhúng mạnh như Wireless router hoặc hệ thống định vị GPS. Lý do của điều này là:

  - Hệ thống này có cổng để kết nối đến các chip nhúng thông dụng
  - Hệ thống cho phép sử dụng lại các đoạn mã sẵn có phổ biến như các trình điều khiển thiết bị, Web Servers, Firewalls, …
  - Việc phát triển hệ thống có thể được tiến hành với một tập nhiều loại đặc tính, chức năng còn sau đó lúc phân phối sản phẩm, hệ thống có thể được cấu hình để

loại bỏ một số chức năng không cần thiết. Điều này giúp tiết kiệm được những vùng nhớ mà các chức năng đó chiếm giữ.

- Hệ thống có chế độ người dùng để dễ dàng chạy các ứng dụng và gỡ rối. Nhờ đó, quy trình phát triển được thực hiện dễ dàng hơn và việc lập trình có tính linh động hơn.
- Có nhiều hệ thống nhúng thiếu các yêu cầu chặt chẽ về tính thời gian thực của hệ thống quản lý. Còn một hệ thống như Embedded Linux có tốc độ đủ nhanh để trả lời cho nhiều ứng dụng. Các chức năng cần đến sự phản ứng nhanh cũng có thể được đặt vào phần cứng.

Câu hỏi

Câu 1. Một hệ thống IoT gồm có ba thành phần nào?

a) Thiết bị thông minh, Ứng dụng IoT, Giao diện đồ họa người dùng
b) Thiết bị thông minh, Ứng dụng IoT, người sử dụng
c) Thiết bị thông minh, Giao diện đồ họa người dùng, người sử dụng
d) Ứng dụng IoT, Giao diện đồ họa người dùng, người sử dụng

Câu 2. Công nghệ xác thực không dùng mật khẩu là quá trình xác minh danh tính của người dùng bằng một yếu tố khác không phải là mật khẩu?

a) Đúng
b) Sai

Câu 3. Thực tại ảo (Virtual reality - VR) có 3 đặc tính chính nào?

a) Tương tác (Interactive), Nhập vai (Immersion), Tưởng tượng (Imagination)
b) Tương tác, Nhập vai, lập trình
c) Tương tác, Tưởng tượng, lập trình
d) Nhập vai (Immersion), Tưởng tượng (Imagination), lập trình

Câu 4. Khái niệm về công nghệ AI xuất hiện đầu tiên vào thời gian nào?

a) Năm 1955
b) Năm 1956
c) Năm 1957
d) Năm 1958

Câu 5. Dựa vào mức độ phức tạp của công nghệ AI, có thể chia thành mấy loại?

a) 2 loại
b) 3 loại
c) 4 loại
d) 5 loại

Bài tập cuối chương

Bài 8.1. IoT giúp cải thiện cuộc sống của con người như thế nào? Lấy một vài ví dụ minh hoạ.

Bài 8.2. Hãy nêu nguyên lý hoạt động của công nghệ xác thực không mật khẩu

Bài 8.3. Tại sao cần sử dụng công nghệ xác thực không mật khẩu?

Bài 8.4. Hãy so sánh 03 hướng công nghệ chính của thực tại ảo: thực tế ảo (VR), thực tế ảo tăng cường (AR) và thực tế hỗn hợp (MR).

Bài 8.5. Hãy trình bày **c**ác bước lập trình VR để tạo ta một sản phẩm thực tế ảo.

Bài 8.6. Hãy nêu ứng dụng của giải pháp tự động hoá quy trình RPA trong các lĩnh vực: Du lịch, y tế, công nghệ thông tin, xây dựng.

Bài 8.7. Hãy trình bày các tính năng nổi bật của các hệ thống phần mềm sử dụng giải pháp RPA

Bài 8.8. Dựa vào mức độ phức tạp của công nghệ AI, có thể chia thành 4 loại nào? Trình bày tóm tắt về 4 loại đó.

Bài 8.9. Hãy nêu một số loại kiến trúc phần mềm thông dụng trong các hệ thống nhúng. Nêu ưu, nhược điểm của các kiến trúc đó.

Bài 8.10. Trình bày các đặc điểm chung của các hệ thống nhúng

Tài liệu tham khảo

- Giáo trình chính

[1] Ian Sommerville (2015), *Software Engineering*, 9th Edition, Addison – Wesley.

[2] Bộ môn Công nghệ Phần mềm, Khoa Công nghệ thông tin, Đại học Công nghệ Thông tin và Truyền thông, Đại học Thái Nguyên (2022), *Bài giảng Nhập môn công nghệ phần mềm (Lưu hành nội bộ)*.

- Tài liệu tham khảo

[3] Ivan Marsic (2012), *Software Engineering,* Rutgers University, New Brunswick, New Jersey.

[4] Rajib Mall (2014), *Fundamentals of Software Engineering, Fourth Edition*, PHI Learning Private Limi ted, Delhi.

[5] Eric J. Braude and Michael E. Bernstein (2016), *Software Engineering - Modern Approaches, Second Edition,* Waveland Press, Inc.

[6] Len Bass, Paul Clements, Rick Kaman (2015) *Software Architecture in Practice* (3rd), Addison - Wesley.

[7] https://en.wikipedia.org/wiki/History_of_software_engineering

[8] http://catless.ncl.ac.uk/Risks/).

[9]https://www.techsignin.com/3-xu-huong-dinh-hinh-cong-nghe-phan-mem-nam-2020/

[10] https://en.wikipedia.org/wiki/IEEE_Standards_Association

[11] https://shecancode.io/blog/my-software-engineer-roadmap-by-joana-carneiro

[12] https://itviec.com

[13] https://github.com/MunGell/awesome-for-beginners

[14] https://personal.utdallas.edu/~chung/RE/IEEE830-1993.pdf

[15]https://sceweb.uhcl.edu/helm/RUP_course_example/courseregistrationproject/indexcourse.htm

[16]https://sceweb.uhcl.edu/helm/RUP_course_example/courseregistrationproject/indexcourse.htm

[17] VADAPALLI, Sricharan. *DevOps: continuous delivery, integration, and deployment with DevOps: dive into the core DevOps strategies*. Packt Publishing Ltd, 2018.

[18] Võ Đình Hiếu (2016), *Giáo trình kiến trúc hướng dịch vụ*, NXB Đại học Quốc gia Hà Nội

[19] http://www.softwaretestinghelp.com/what-is-stlc-v-model/

[20] https://co-well.vn/nhat-ky-cong-nghe/7-nguyen-tac-kiem-thu-phan-mem-co-ban/

[21]http://www.softwaretestingstuff.com/2010/09/unit-testing-best-practices-techniques.html

[22] http://softwaretestingfundamentals.com/unit-testing/

[23] https://www.slideshare.net/guest45ac48/unit-test

[24] https://professionalqa.com/system-testing

[25] https://www.educba.com/system-testing

[26] https://professionalqa.com/acceptance-testing

[27] https://softwaretestingfundamentals.com/acceptance-testing/

[28] https://en.wikipedia.org/wiki/Acceptance_testing

[29] Phạm Ngọc Hùng, Trương Anh Hoàng, Đặng Văn Hưng (2014), Giáo trình kiểm thử phần mềm.

[30] Nguyễn Văn Vỵ, Nguyễn Việt Hà (2009), Giáo trình kỹ nghệ phần mềm.

[31] https://aws.amazon.com/vi/what-is/iot/

[32] Steven M. LaValle (2019), *Virtual Reality*, Cambridge University Press.

[33] Capstone Project Document - SPCS GROUP (2018), Sharing platform for cleaning services, FPT University.

Phụ lục

I. PHỤ LỤC 1 – TÀI LIỆU ĐẶC TẢ PHẦN MỀM

Cấu trúc chung của tài liệu đặc tả yêu cầu dựa theo chuẩn IEEE 839 - 1984:

1. Giới thiệu
    - *1.1. Mục đích của tài liệu yêu cầu*
    - *1.2. Phạm vi của sản phẩm*
    - *1.3. Các định nghĩa, từ viết tắt*
    - *1.4. Các tham chiếu*
    - *1.5. Tổng quan về tài liệu yêu cầu (mô tả cấu trúc tài liệu)*
2. Mô tả chung
    - *2.1. Tổng quan về sản phẩm*
    - *2.2. Các chức năng của sản phẩm*
    - *2.3. Đối tương người dùng*
    - *2.4. Các ràng buộc tổng thể*
    - *2.5. Giả thiết và các phụ thuộc*
3. Đặc tả yêu cầu (yêu cầu chi tiết)
    - *3.1 Yêu cầu chức năng*
        - *3.1.1 Yêu cầu chức năng 1*
            - *3.1.1.1 Giới thiệu*
            - *3.1.1.2 Dữ liệu vào*
            - *3.1.1.3 xử lý*
            - *3.1.1.4 Kết quả*
        - *3.1.2 Yêu cầu chức năng 2*
        - *...*
        - *3.1.n Yêu cầu chức năng n*
    - *3.2 Các yêu cầu phi chức năng*
        - *3.2.1 Các thuộc tính của hệ thống*
        - *3.2.2 Các ràng buộc của hệ thống*
        - *3.2.3 Các yêu cầu khác.*

4. Phụ lục

5. Chỉ mục

II. PHỤ LỤC 2 - TÀI LIỆU THIẾT KẾ HỆ THỐNG

Mục Lục:

Theo dõi phiên bản tài liệu

1. Giới thiệu
    1.1. Mục đích
    1.2. Phạm vi
    1.3. Bảng chú giải thuật ngữ
    1.4. Tài liệu tham khảo
    1.5. Tổng quan về tài liệu
2. Tổng quan hệ thống
3. Kiến trúc hệ thống
    3.1 Thiết kế kiến trúc
    3.2. Mô tả sự phân rã
    3.3. Cơ sở thiết kế
4. Thiết kế dữ liệu
    4.1. Mô tả dữ liệu
    4.2. Từ điển dữ liệu
5. Thiết kế theo chức năng
    5.1. Chức năng 1
    5.2. Chức năng 2
    5.3. Chức năng 3
    ……
6. Bảng tham khảo tới các yêu cầu
7. Các phụ lục

III. PHỤ LỤC 3 – BÁO CÁO TỔNG KẾT SAU DỰ ÁN

1. Giới thiệu chung về dự án
    1.1. Mục đích
    1.2. Phạm vi
2. Tình hình/ hiện trạng trước thực hiện dự án
3. Tóm tắt nội dung công việc của dự án
4. Những điểm đã đạt được/ thành công

4.1. Các thành công

4.2. Thảo luận, bài học từng thành công

5. Các vấn đề gặp phải khi thực hiện dự án

5.1. Thảo luận từng vấn đề

5.2. Cách khắc phục, rút kinh nghiệm

6. Cơ hội cho công việc tương lai

Các câu hỏi thường gặp

1. Phần cứng là gì?

Câu trả lời: Là các thiết bị, cấu kiện mang tính vật lý có thể tiếp xúc bằng tay được như máy in, ổ đĩa, máy quét, các mạch tích hợp, ...

2. Phần mềm là gì?

Câu trả lời: Theo nghĩa đơn giản nhất, phần mềm là tệp dữ liệu và các chương trình chứa các dòng lệnh chỉ thị cho máy tính thực hiện một công việc nào đó

3. Phần mềm có những đặc trưng gì?

Câu trả lời: Phần mềm có các đặc trưng sau:

- Không mòn cũ, nhưng thoái hóa theo thời gian
- Khó được lắp giáp từ các mẫu có sẵn
- Phức tạp, khó hiểu, vô hình (tư duy logic, không nhìn thấy)
- Thay đổi là bản chất
- Nhu cầu thay đổi, môi trường vận hành thay đổi
- Cần phát triển theo nhóm

4. Phân loại phần mềm theo những tiêu chí nào?

Câu trả lời: Phần mềm có thể được phân loại theo 3 tiêu chí chính: (1) theo mức độ hoàn thiện; (2) theo chức năng thực hiện; (3) theo lĩnh vực áp dụng.

5. Công nghệ phần mềm là gì?

Câu trả lời: Là một chuyên ngành nghiên cứu thuộc lĩnh vực công nghệ thông tin. Theo quan điểm của các nhà nghiên cứu, SE ngày nay có thể xem như một mô hình gồm 4 tầng*: "SE có thể xem như một mô hình gồm 4 tầng: tầng quy trình, tầng phương pháp, tầng công cụ và tầng chất lượng hướng tới mục tiêu xây dựng các sản phẩm phần mềm đáp ứng các yêu cầu người dùng, đảm bảo chất lượng, không vượt quá kinh phí, thời hạn cho trước".*

6. CASE tool là gì?

Câu trả lời: Là các sản phẩm phần mềm nhằm mục đích hỗ trợ tự động hoặc bán tự động cho các hoạt động trong quy trình phần mềm

7. Vòng đời phần mềm là gì?

Câu trả lời: Là một tập hợp các hoạt động có cấu trúc, được thực hiện theo trình tự nhất định (song song, tuần tự) nhằm xây dựng mới hoặc nâng cấp phần mềm đang tồn tại.

8. Cách thức xác định kích cỡ phần mềm?

Câu trả lời: Ước lượng kích cỡ phần mềm là phần công việc quan trọng trong quản lý dự án phần mềm. Nó giúp người quản lý có thể dự án thời gian cũng như các nỗ lực cần thiết để triển khai dự án. Một số cách thức ước lượng kích cỡ phần mềm

- Số dòng mã (LOC - Lines of Code)
- Số thực thể trong biểu đồ quan hệ
- Tổng số quy trình trong luồng dữ liệu chi tiết
- Số điểm chức năng (Function points)

……

9. Kiểm thử phần mềm là gì?

Câu trả lời: Kiểm thử là tiến trình vận hành hệ thống hoặc thành phần dưới những điều kiện xác định, quan sát hoặc ghi nhận kết quả và đưa ra đánh giá về hệ thống hoặc thành phần đó. Mục đích của kiểm thử phầm mềm là tìm lỗi

10. Quy trình phần mềm gồm 4 giai đoạn cơ bản nào?

Câu trả lời: Trong quy trình phần mềm gồm 4 giai đoạn cơ bản sau: Đặc tả phần mềm, Thiết kế và cài đặt phần mềm, Đánh giá phần mềm, Cải tiến phần mềm.

11. Nghiên cứu tính khả thi trong kỹ nghệ phần mềm nhằm mục đích gì?

Câu trả lời: Là hoạt động nhằm phân tích và đánh giá xem dự án có khả thi về mặt tài chính và về mặt công nghệ không.

12. SRS là gì?

Câu trả lời: SRS - Software Requirement Specification là tài liệu đặc tả yêu cầu phần mềm.

13. Biểu đồ Use case là gì?

Câu trả lời: Biểu đồ UC là biểu đồ hành vi, giúp trực quan hoá các tương tác giữa các tác nhân và hệ thống được phát triển. Biểu đồ gồm hệ thống, các use case liên quan, các tác nhân và các mối quan hệ giữa:

- System: cái gì đang được mô tả?
- Actor: người sử dụng hệ thống, tham gia vào các hoạt động của hệ thống?
- Use case: những gì actor cần làm?

14. Thiết kế phần mềm là gì?

Câu trả lời: Thiết kế phần mềm là hoạt động chuyển đặc tả yêu cầu thành mô hình thiết kế mà người lập trình có thể chuyển thành chương trình với một ngôn ngữ lập trình cụ thể, chương trình có thể vận hành được, đáp ứng được yêu cầu đặt ra.

15. Quản lý yêu cầu phần mềm là gì?

Câu trả lời: Là tập các hoạt động nhằm quản lý; thiết lập; duy trì các thương lượng với khách hàng & người dùng về các yêu cầu phần mềm.

16. Mẫu thử phần mềm trong kỹ nghệ yêu cầu là gì?

Câu trả lời: Mẫu thử phần mềm là phiên bản sản phẩm được phát triển nhanh chóng để chuyển giao đến khách hàng dùng thử nhằm thẩm định các yêu cầu cần thu thập.

17. Kỹ nghệ yêu cầu là gì?

Câu trả lời: Là quy trình xác định mục tiêu dự án; định nghĩa; lập tài liệu yêu cầu; & quản lý các yêu cầu trong suốt tiến trình dự án

18. Yêu cầu phần mềm là gì?

Câu trả lời: Là một phát biểu/mô tả/đặc tả về dịch vụ mà hệ thống cung cấp, hoặc mô tả về một ràng buộc/điều kiện/phụ thuộc mà hệ thống phải thoả mãn; hoặc cũng có thể là một mục tiêu mà hệ thống phần mềm phải đạt được

19. Bảo trì phần mềm là gì?

Câu trả lời: Là việc sửa đổi phần mềm sau khi phát hành nhằm chỉnh sửa các lỗi phát sinh, cải thiện hiệu năng, các thuộc tính của phần mềm hoặc làm cho phần mềm thích ứng với môi trường vận hành mới

20. Stakeholders là gì?

Câu trả lời: Stakeholder tạm dịch là các bên liên quan, hoặc những cá nhân/tổ chức liên quan đến dự án xây dựng hệ thống, hoặc những tổ chức/cá nhân sẽ cung cấp yêu cầu cho dự án, hoặc bị tác động bởi dự án. Họ có thể là những người sử dụng cuối; khách hàng; đối tác đầu tư; các đối thủ cạnh tranh; các tác nhân thị trường; các kỹ sư phần mềm; người quản lý; hoặc các chuyên gia lĩnh vực; ….

21. Kỹ nghệ ngược (reverse engineering) là gì?

Câu trả lời: Là quy trình phân tích hệ thống để xác định các thành phần và mối quan hệ giữa chúng từ đó tạo dạng biểu diễn mới của hệ thống ở mức độ trừu tượng hoá cao hơn

22. Có những công cụ hỗ trợ quá trình kỹ nghệ ngược nào?

Câu trả lời: Kỹ nghệ ngược nếu được tiến hành thủ công sẽ mất rất nhiều thời gian và sức người. Do đó nó cần được hỗ trợ bởi các công cụ tự động. Ví dụ về một số công cụ kỹ nghệ ngược:

- **CIAO and CIA:** A graphical navigator for software and web repositories along with a collection of Reverse Engineering tools.
- **Rigi:** A visual software understanding tool.

- **Bunch:** A software clustering/modularization tool.
- **GEN++:** An application generator to support development of analysis tools for the C++ language.
- **PBS:** Software Bookshelf tools for extracting and visualizing the architecture of programs.

23. Bảo trì phần mềm có thể được chia thành những loại nào?

Câu trả lời: Bốn loại bảo trì phần mềm bao gồm:

- Bảo trì sửa lỗi: sửa chữa các thất bại phần mềm
- Bảo trì thích nghi: thích nghi phần mềm với môi trường thay đổi mới
- Bảo trì hoàn thiện: sửa đổi hoặc nâng cấp hệ thống đáp ứng các yêu cầu mới
- Bảo trì phòng ngừa: tạo các thay đổi để cải tiến khả năng bảo trì của hệ thống trong tương lai.

24. Quản lý cấu hình phần mềm là gì?

Câu trả lời: Là quy trình theo dõi và điều khiển các thay đổi xảy ra trong suốt vòng đời phát triển phần mềm. Bất cứ thay đổi nào được tạo trong quy trình phát triển phần mềm phải được theo dõi qua một quy trình rõ ràng có giám sát một cách chặt chẽ, đây là nhiệm vụ chính của hoạt động quản lý cấu hình phần mềm.

25. Thế nào là lỗi phần mềm?

Câu trả lời: Có rất nhiều định nghĩa khác nhau về lỗi phần mềm, nhưng tựu chung, có thể phát biểu một cách tổng quát: *"Lỗi phần mềm là sự không khớp giữa chương trình và đặc tả của nó".*

Bài tập thảo luận

Yêu cầu chung:

- Sinh viên chia nhóm thực hiện đề tài, thảo luận (5-6 sinh viên/1 nhóm), lập bảng phân công công việc chi tiết cho từng thành viên trong nhóm.
- Các nhóm có thể đề xuất đề tài thực hiện hoặc theo sự phân công của giảng viên.
- Sau khi kết thúc học phần mỗi nhóm sinh viên phải: Xây dựng được một phần mềm mô phỏng, các bước thực hiện theo đúng qui trình. Hoàn thiện toàn bộ các tài liệu, giấy tờ theo đúng biểu mẫu, tương ứng với từng giai đoạn phát triển phần mềm. Hoặc tìm hiểu về một lĩnh vực mới trong Công nghệ phần mềm.

1. <u>Thảo luận 1:</u> Lập kế hoạch dự án phần mềm (Số tiết: 03 tiết)

a) Mục tiêu

- Sinh viên nắm được những kiến thức tổng quan về công nghệ phần mềm. Biết vận dụng những kiến thức đã học vào cuộc sống nói chung và thực tế nghề nghiệp nói riêng.

- Nắm vững kiến thức: Vòng đời phát triển phần mềm, quy trình phát triển phần mềm, các mô hình phát triển phần mềm như mô hình thác nước, mô hình mẫu thử, mô hình xoắn ốc, mô hình Aglie,.... Có kỹ năng lựa chọn được mô hình hệ thống phù hợp với yêu cầu bài toán.

- Sinh viên nắm vững những kiến thức cơ bản về: Quản lý dự án phần mềm, lập kế hoạch dự án, ước lượng dự án. Nghiên cứu và sử dụng thành thạo 1 công cụ để hỗ trợ việc quản lý dự án.

b) Nội dung sinh viên phải biết

- Sinh viên nắm vững và tổng hợp được những kiến thức cơ bản trong lĩnh vực công nghệ phần mềm như: Các khái niệm, vòng đời phần mềm, qui trình phát triển phần mềm, các mô hình phát triển phần mềm, quản lý dự án phần mềm ...

- Nghiên cứu các mô hình quy trình phần mềm khác nhau. Lựa chọn mô hình phù hợp cho bài toán.

- Nắm được các nội dung chính trong bộ tiêu chuẩn CMM, CMMI. Cách áp dụng.

- Có hiểu biết về xu hướng mới của lĩnh vực công nghệ phần mềm.

- Phát biểu bài toán cần giải quyết

- Lý thuyết về tổ chức dự án và lập kế hoạch quản lý dự án, sử dụng công cụ hỗ trợ việc quản lý dự án

- Kỹ thuật xác định phạm vi dự án, xây dựng bản WBS.

- Phân rã các chức năng đã xác định thành các nhiệm vụ (Task) để lập một kế hoạch dự án đơn giản.

c) Nội dung chủ đề thảo luận

Các nhóm thảo luận/ trình bày các vấn đề chính theo yêu cầu của giảng viên bao gồm:

1. Tìm hiểu các nội dung chính trong bộ tiêu chuẩn CMM, CMMI. Cách áp dụng các mô hình này trên thực tế. Dẫn chứng bằng các công ty ở Việt Nam đã áp dụng thành công các mô hình này.
2. Các xu hướng mới của lĩnh vực công nghệ phần mềm.
3. Nghiên cứu, lựa chọn sử dụng một mô hình quy trình phần mềm để phát triển phần mềm trong đề tài của nhóm.
4. Lập kế hoạch quản lý dự án
5. Xây dựng được bản phân rã cấu trúc công việc (WBS).
6. Tìm hiểu về các công cụ hỗ trợ việc quản lý dự án (Các nhóm tự chọn công cụ quản lý dự án).

2. <u>Thảo luận 2:</u> Đặc tả phần mềm (Số tiết: 03 tiết)

a) Mục tiêu

Sinh viên nắm vững những kiến thức tổng quan về kỹ nghệ yêu cầu. Đứng trước một yêu cầu trong thực tế sinh viên có kỹ năng xác định được bài toán, và đưa ra được hướng giải quyết vấn đề.

b) Nội dung sinh viên phải biết

- Sinh viên cần biết cách phát biểu bài toán, tìm hiểu về các chức năng nghiệp vụ liên quan đến bài toán.

- Có kỹ năng xây dựng: Tài liệu đặc tả phần mềm.

- Tìm hiểu và chuẩn bị các công cụ cần thiết để làm bài tập.

c) Nội dung chủ đề thảo luận

Các nhóm thảo luận/ trình bày các vấn đề chính theo yêu cầu của giảng viên bao gồm:

1. Giới thiệu bài toán (Case Study).
2. Tìm hiểu, thu thập và phân tích các yêu cầu về chức năng nghiệp vụ của bài toán.
3. Tìm hiểu và cài đặt các công cụ cần thiết để thực hiện dự án.
4. Viết tài liệu đặc tả phần mềm *(Tham khảo phụ lục 1)*
    a. Phương pháp đặc tả
    b. Xác định rõ ràng các yêu cầu phần mềm
    c. Tài liệu đặc tả phần mềm
    d. Bản đánh giá yêu cầu đã xác định được ở trên

e. Lập kế hoạch quản lý yêu cầu

3. <u>Thảo luận 3:</u> Phân tích thiết kế và cài đặt phần mềm (Số tiết: 03 tiết)

a) Mục tiêu

Biết phân tích thiết kế hệ thống, và tiến hành cài đặt các yêu cầu cơ bản của phần mềm. Thực hành với công cụ tổ chức, quản lý và chia sẻ mã nguồn.

b) Nội dung sinh viên phải biết

- Hiểu được mô hình hoá hệ thống là gì? Và tại sao phải mô hình hoá hệ thống, phân biệt được các mô hình hệ thống, có khả năng lựa chọn và ứng dụng các mô hình hệ thống vào từng trường hợp cụ thể.

- Biết cách áp dụng các phương pháp phân tích thiết kế hệ thống: theo hướng cấu trúc, hướng đối tượng, hướng cấu phần...

- Bước đầu cài đặt được các chức năng cơ bản.

- Thực hành với công cụ GIT**:** Cài đặt/đăng ký tài khoản và biết cách sử dụng Git & GitHub.

c) Nội dung chủ đề thảo luận

Các nhóm thảo luận/ trình bày các vấn đề chính theo yêu cầu của giảng viên bao gồm:

1. Báo cáo tài liệu phân tích thiết kế phần mềm *(Tham khảo phụ lục 2).*
2. Lựa chọn ngôn ngữ lập trình, tiến hành cài đặt phần mềm: CSDL, giao diện người dùng, Cài đặt được ½ các chức năng của phần mềm.
3. Thực hành với công cụ GIT: Cài đặt/đăng ký tài khoản và biết cách sử dụng Git&GitHub.

4. <u>Thảo luận 4</u>: Kiểm thử phần mềm (Số tiết: 03 tiết)

a) Mục tiêu

Nắm được quy trình kiểm thử phần mềm. Tìm hiểu chi tiết về kiểm thử thành phần và kiểm thử hệ thống; các phương pháp được sử dụng. Có kỹ năng thiết kế kịch bản kiểm thử, các trường hợp kiểm thử và sử dụng các công cụ giúp kiểm thử tự động.

b) Nội dung sinh viên phải biết

- Nắm vững quy trình, các phương pháp kiểm thử phần mềm.

- Có hiểu biết về các công cụ kiểm thử tự động.

- Tiến hành kiểm thử trên phần mềm mà nhóm xây dựng.

c) Nội dung chủ đề thảo luận

Các nhóm thảo luận/ trình bày các vấn đề chính theo yêu cầu của giảng viên bao gồm:

1. Lập kế hoạch kiểm thử
2. Trình bày phương pháp kiểm thử nhóm lựa chọn để kiểm thử phần mềm
3. Giới thiệu tổng quan về công cụ kiểm thử mà nhóm sẽ sử dụng
4. Thiết kế kịch bản kiểm thử và một số TestCase cơ bản
5. Thực thi test: có sử dụng các công cụ kiểm thử tự động
6. Báo cáo kết quả kiểm thử thực tế trên phần mềm, đánh giá kết quả và giải pháp khắc phục (nếu có).

5. <u>Thảo luận 5:</u> Quản lý cấu hình & Bảo trì, đào tạo, hoàn thiện phần mềm (Số tiết: 03 tiết)

a) Mục tiêu

Nắm vững kiến thức về Quản lý cấu hình phần mềm, Lập kế hoạch bảo trì phần mềm, Lập kế hoạch đào tạo phần mềm. Hoàn thiện các chức năng trên phần mềm cùng các mẫu biểu giấy tờ để kết thúc dự án.

b) Nội dung sinh viên phải biết

- Có kỹ năng quản lý cấu hình, lập kế hoạch bảo trì, đào tạo người sử dụng cho phần mềm.

- Nắm được các vấn đề liên quan đến bảo trì: phân loại, phương pháp, chi phí bảo trì …

- Hiểu được một số quy trình và các chiến lược cải tiến phần mềm

- Hoàn thiện phần mềm với đầy đủ các chức năng cơ bản.

- Cài đặt/ đăng ký sử dụng thành công công cụ hỗ trợ việc quản lý cấu hình, bảo trì phần mềm. Sử dụng thành thạo công cụ.

c) Nội dung chủ đề thảo luận

Các nhóm thảo luận/ trình bày các vấn đề chính theo yêu cầu của giảng viên bao gồm:

1. Kế hoạch quản lý cấu hình phần mềm.
2. Báo cáo bản kế hoạch bảo trì phần mềm.
3. Báo cáo bản kế hoạch đào tạo phần mềm cho người sử dụng.
4. Hoàn thiện phần mềm với đầy đủ các chức năng cơ bản.

2.6 <u>Thảo luận 6:</u> Tài liệu hướng dẫn sử dụng phần mềm, kết thúc dự án (Số tiết: 03 tiết)

a) Mục tiêu

Hoàn thiện tài liệu hướng dẫn sử dụng phần mềm cùng các mẫu biểu giấy tờ để kết thúc dự án.

b) Nội dung sinh viên phải biết

- Viết tài liệu hướng dẫn sử dụng phần mềm
- Báo cáo tổng kết kết thúc dự án phần mềm.

c) Nội dung chủ đề thảo luận

Các nhóm thảo luận/ trình bày các vấn đề chính theo yêu cầu của giảng viên bao gồm:

1. Tài liệu hướng dẫn sử dụng phần mềm
2. Báo cáo tổng kết kết thúc dự án phần mềm *(Xem phụ lục 3)*

Hướng dẫn/ Gợi ý:

❖ Điều kiện kết thúc dự án:

- Đã hoàn thành các yêu cầu dự án
- Chưa hoàn thành các yêu cầu, nhưng có các yếu tố sau:
    - Kinh phí hết, không được cấp thêm
    - Thời hạn hết, không cho phép gia hạn
    - Ban quản lý và nhà tài trợ quyết định dừng dự án
    - Những lý do đặc biệt khác

❖ Các công việc cần thực hiện khi kết thúc dự án:

- Đóng gói dự án
    - Đánh giá thành viên tham gia và kiến nghị lợi ích
    - Hoàn thiện tài liệu, chứng từ, dữ liệu
    - Cảm ơn người tham gia, giúp đỡ
    - Xử lý vấn đề tổ chức, nguồn lực liên quan
- Tổng kết sau dự án
    - Xác định mặt mạnh, mặt yếu của sản phẩm
    - Đánh giá sự hài lòng của khách hàng
    - Đánh giá mặt được, chưa được của công tác quản lý
    - Bài học kinh nghiệm, bàn giao
- Thanh lý hợp đồng với khách, đối tượng khác.