Biên soạn: TS. HUỲNH XUÂN HIỆP
Th.S. PHAN PHƯƠNG LAN

GIÁO TRÌNH

Nhập môn
CÔNG NGHỆ PHẦN MỀM

NHÀ XUẤT BẢN ĐẠI HỌC CẦN THƠ
2011

LỜI GIỚI THIỆU

Hiện nay, khoa học công nghệ đã và đang chiếm nhiều ưu thế vượt trội. Nó là một cuộc cách mạng lớn góp phần thay đổi toàn xã hội. Để góp phần làm phong phú thêm nguồn tư liệu phục vụ nghiên cứu, học tập cho bạn đọc trong và ngoài ngành Công nghệ thông tin và truyền thông, Nhà Xuất bản Đại học Cần Thơ xin được phép ấn hành và giới thiệu cùng bạn đọc giáo trình "Nhập môn công nghệ phần mềm" do TS. Huỳnh Xuân Hiệp biên soạn. Giáo trình bao gồm 05 chương với 132 trang; Nội dung các chương giới thiệu khái quát công nghệ phầm mềm, tiến trình phần mềm, chu trình sống của phần mềm, cách ước lượng chi phí để tính giá trị của các phần mềm. Thêm vào đó, cuối mỗi chương còn có rất nhiều tài liệu tham khảo hữu ích cho bạn đọc. Giáo trình là tài liệu tham khảo có giá trị cho sinh viên các ngành Công nghệ thông tin và bạn đọc muốn tham khảo những vấn đề liên quan đến phần mềm.

Nhà Xuất bản Đại học Cần Thơ chân thành cám ơn các Tác giả và sự đóng góp ý kiến của quý Thầy Cô trong Hội đồng thẩm định trường Đại học Cần Thơ để giáo trình "Nhập môn công nghệ phần mềm" được ra mắt bạn đọc.

Nhà Xuất bản Đại học Cần Thơ trân trọng giới thiệu đến giảng viên, sinh viên và bạn đọc giáo trình này.

Chân thành cám ơn!

NHÀ XUẤT BẢN ĐẠI HỌC CẦN THƠ

LỜI NÓI ĐẦU

Quyển giáo trình *Nhập môn Công nghệ phần mềm* mang đến cái nhìn tổng quan về quá trình thực hiện phần mềm cũng như những giai đoạn đặc trưng của quá trình này. Trong nội dung giáo trình cũng đề cập đến một quy trình công nghệ phần mềm tiêu biểu là quy trình RUP (Rational Unified Process). Các kỹ thuật tính toán ước lượng về kích thước phần mềm, chi phí, nhân lực và thời gian thực hiện được đề cập đến một cách khá chi tiết. Kỹ thuật ước lượng trị giá phần mềm theo công văn 3364/UDCNTT-BTTTT của Bộ Thông tin & Truyền thông cũng được đưa vào giảng dạy như là một nội dung thực tiễn mà sinh viên chuyên ngành máy tính/công nghệ thông tin cần nắm bắt được khi ra trường.

Việc biên soạn giáo trình nhằm mục tiêu giảng dạy cho các sinh viên trình độ kỹ sư máy tính/công nghệ thông tin (các chuyên ngành kỹ thuật phần mềm, hệ thống thông tin,...) trên cơ sở *Bài giảng Nhập môn Công nghệ phần mềm* được tác giả giảng dạy tại Trường Đại học Cần Thơ từ nhiều năm qua và tham khảo các kiến thức, các nguồn tài liệu được công nhận và ứng dụng rộng rãi của các nhà chuyên môn, các chuyên gia và các đồng nghiệp.

Những đóng góp chuyên môn sâu của các đồng nghiệp thuộc Bộ môn Công nghệ phần mềm, Khoa Công nghệ Thông tin & Truyền thông và Hội đồng thẩm định giáo trình của Trường Đại học Cần Thơ đã giúp giáo trình được hoàn thiện hơn rất nhiều.

Trong quá trình biên soạn giáo trình chắc chắn không tránh khỏi có những thiếu sót. Rất mong nhận được sự góp ý, phản hồi từ các bạn sinh viên, các đồng nghiệp và các chuyên gia trong lĩnh vực công nghệ phần mềm để quyển giáo trình này có chất lượng ngày càng tốt hơn.

Cần Thơ, ngày 22 tháng 06 năm 2011

HUỲNH XUÂN HIỆP
PHAN PHƯƠNG LAN

MỤC LỤC

DANH SÁCH BẢNG

DANH SÁCH HÌNH

MỘT SỐ THUẬT NGỮ

Artifacts	Tác vụ
Capability Maturity Models	Mô hình khả trưởng
Code-and-Fix Model	Mô hình xây dựng và hiệu chỉnh
Computer-Aided Software Engineering	Công cụ hỗ trợ phát triển phần mềm
Configuration Control	Kiểm soát cấu hình
Constructive Cost Model (COCOMO)	Mô hình chi phí kiến tạo
Cohesion Level	Mức độ gắn kết
Coupling Level	Mức độ nối kết
Data Dictionary	Từ điển dữ liệu
Egoless Programmer	Lập trình viên bản ngã
Evolution-Tree Model	Mô hình cây tiến hóa
Extreme Programming and Agile Process	Mô hình phát triển nhanh
Files – Flows – Processes	Tập tin – Luồng – Tiến trình
Fountain Model	Mô hình vòi phun nước
Function Point	Điểm chức năng
Incremental Model	Mô hình tăng trưởng
Iterative-and-Incremental Model	Mô hình lặp và tăng trưởng
Kilo or Thousand Delivered Source Instructions (KDSI/KLOC)	Số dòng mã lệnh tính theo đơn vị ngàn
Lines of Code	Dòng mã lệnh
Maintenance Programmer	Lập trình viên bảo trì
Object Point	Điểm đối tượng
Open Source Model	Mô hình mã nguồn mở
Rational Unified Process® (RUP)	Tiến trình hợp nhất Rational
Rapid Application Development	Mô hình phát triển ứng dụng nhanh
Rapid Prototyping Model	Mô hình bản mẫu
Requirements Analysis Team (RAT)	Nhóm phân tích yêu cầu
Risk Incremental Model	Mô hình tăng trưởng nhiều rủi ro

Role/Worker	Vai trò
Self –Documenting Code	Chú thích tự thân
Synchronize-and-Stabilize Model	Mô hình đồng bộ và ổn định
Spiral Model	Mô hình xoắn ốc
Software Development	Phát triển phần mềm
Software Engineering	Công nghệ phần mềm, kỹ thuật phần mềm, kỹ nghệ phần mềm
Software Process	Tiến trình phần mềm
Software Project Management Plan	Kế hoạch quản lý dự án phần mềm
Software Quality Assurance (SQA)	Bảo đảm chất lượng phần mềm
Unified Modeling Language (UML)	Ngôn ngữ mô hình hóa hợp nhất
Use Case Point	Điểm trường hợp sử dụng
Validation	Công nhận hợp lệ
Verification	Thẩm tra
Version	Phiên bản
Waterfall Model	Mô hình thác nước

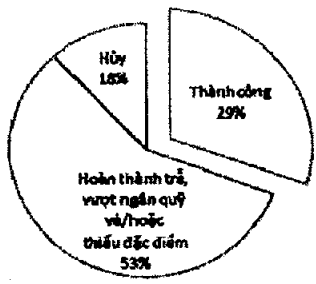
CHƯƠNG 1

TỔNG QUAN

1. PHẠM VI

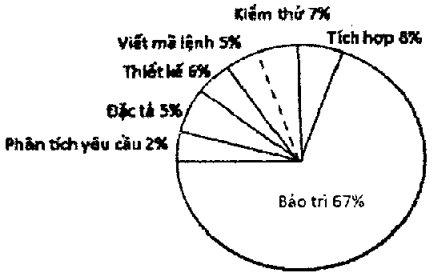
Thuật ngữ công nghệ phần mềm (Software Engineering - SE) được đề xuất đầu tiên vào năm 1967. Từ đó đến nay, sự phát triển của phần mềm đã có những bước phát triển mạnh mẽ. Sự phát triển của phần mềm luôn chịu ảnh hưởng của sự phát triển về phần cứng và hệ điều hành: hệ điều hành đa nhiệm (1960s), bộ nhớ ảo (1970s), đa xử lý (multiprocessor), hệ điều hành phân tán (mạng), hệ điều hành cho di động, hệ điều hành cho hệ thống nhúng,v.v....Song song đó, vấn đề bảo trì phần mềm luôn đặt ra những thách thức không nhỏ.

Khía cạnh kinh tế trong quá trình phát triển phần mềm luôn là vấn đề quan trọng, nhất là sự lựa chọn kỹ thuật thực hiện nhanh hơn để giảm giá thành, sự ảnh hưởng của kỹ thuật mới lên công ty phần mềm (khó bảo trì, thời gian huấn luyện, kinh nghiệm làm việc trên kỹ thuật mới chưa nhiều), sự phụ thuộc vào lựa chọn của khách hàng, luật bản quyền. Hình 1 cho thấy tỉ lệ thành công một cách trọn vẹn chỉ chiếm khoảng 29% số lượng các dự án phần mềm.



Hình 1: *Kết quả thực hiện của hơn 9000 dự án hoàn thành trong năm 2004 [6].*

Khía cạnh bảo trì trong chu trình sống của phần mềm (phân tích yêu cầu, phân tích hệ thống/đặc tả, thiết kế, cài đặt, tích hợp, bảo trì, kết thúc hoạt động) ảnh hưởng rất lớn lên giá thành phần mềm. Trên Hình 2 ta thấy giai đoạn bảo trì, về nguyên tắc, chiếm đến 67% thời gian thực hiện phần mềm (xem thêm Bảng 1).

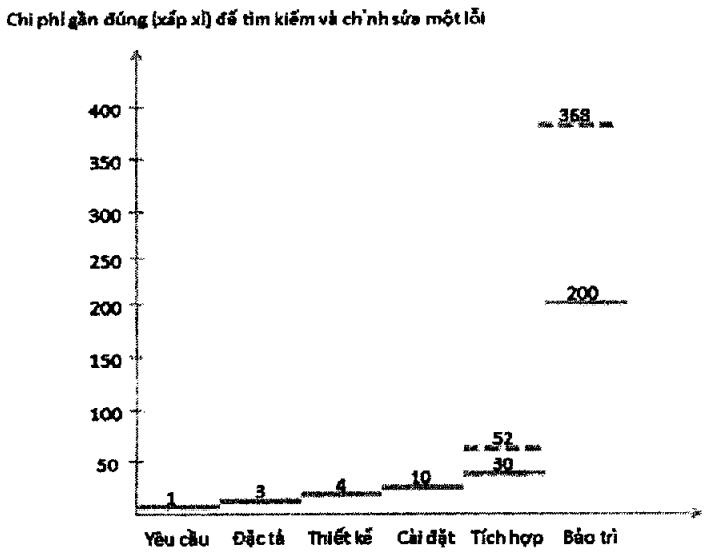


HInh 2: Giá thành của các giai đoạn trong chu trình sống của phần mềm.

Bảng 1: So sánh chi phí các dự án phần mềm.

	Các dự án khác nhau từ 1976 đến 1981	132 dự án gần đây nhất của Hewlett Packard
Giai đoạn yêu cầu và đặc tả	22%	18%
Giai đoạn thiết kế	18%	19%
Giai đoạn cài đặt	36%	34%
Giai đoạn tích hợp	24%	29%
Phát triển	33%	25%
Bảo trì	67%	75%

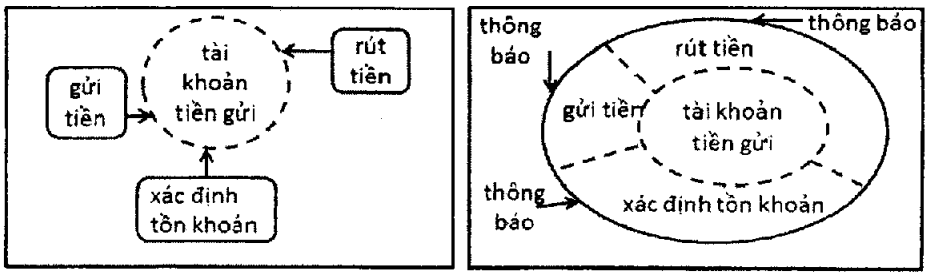
Cần chú ý là sai sót tại các giai đoạn trước sẽ ảnh hưởng rất lớn đến các giai đoạn sau, tạo ra các lỗi. Công việc sửa chữa các lỗi càng sớm càng tốt là rất quan trọng (xem Hình 3).



***Hình 3:** Chi phí phải trả để điều chỉnh lỗi.*

Đội ngũ lập trình là một vấn đề cần được quan tâm sát sao trong quá trình thực hiện phần mềm trên cơ sở hình thành từng nhóm làm việc chuyên biệt trong từng lĩnh vực. Một số vấn đề nảy sinh cần giải quyết như cách chia sẻ các phần công việc, mối quan hệ, sự giao tiếp giữa các thành viên với nhau, kỹ thuật tổ chức và quản lý đội ngũ phát triển phần mềm (lập trình viên, đặc tả viên, thiết kế viên,...), sự ràng buộc lẫn nhau giữa các thành viên (cùng nhóm, khác nhóm,...), cách đánh giá thời gian làm việc, cách đánh giá hiệu quả công việc, cách đánh giá về kinh nghiệm thực hiện công việc.

Một trong những tiếp cận xây dựng phần mềm quan trọng hiện nay là hướng đối tượng xem dữ liệu và tác động có vai trò quan trọng như nhau, hướng kỹ thuật này còn được gọi là thiết kế hướng trách nhiệm hay thiết kế theo hợp đồng. Hình 4 cho thấy sự khác nhau cơ bản giữa hướng cài đặt theo cấu trúc trên cơ sở can thiệp trực tiếp vào dữ liệu và theo hướng đối tượng chỉ can thiệp vào dữ liệu thông qua việc gửi các thông điệp.



(a) (b)

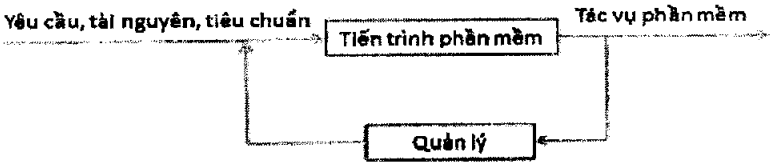
Hình 4: *So sánh hai phương pháp cài đặt (a) cấu trúc và (b) hướng đối tượng.*

Một số thuật ngữ cần quan tâm như: phần mềm phải bao gồm mã lệnh dưới dạng máy có thể đọc được; các dạng tài liệu đặc tả, thiết kế, luật và sổ sách về chi phí; kế hoạch quản lý dự án phần mềm và các tài liệu quản lý khác; các dạng tài liệu hướng dẫn sử dụng. Bên cạnh đó chương trình là một đoạn mã lệnh có thể tự thực thi được trong khi hệ thống là tập hợp các chương trình liên quan với nhau còn sản phẩm là một mẫu bình thường của phần mềm và là kết quả đạt được sau một tiến trình phát triển phần mềm. Một vài thuật ngữ khác như sản xuất phần mềm bao gồm hai giai đoạn phát triển phần mềm và bảo trì, đồng thời cần phân biệt lỗi (error) hay có lỗi (bug) là hai vấn đề khác nhau.

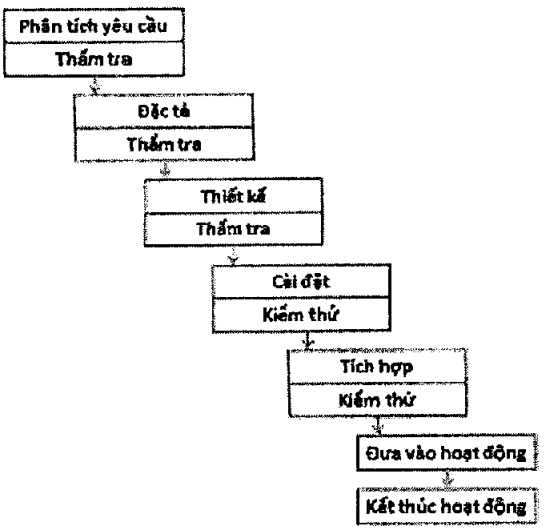
2. TIẾN TRÌNH PHẦN MỀM

Tiến trình phần mềm (software process) là cách thức tạo ra phần mềm, mỗi công ty có tiến trình phần mềm riêng (xem Hình 5, Hình 6). Khách hàng (client) là cá nhân hay công ty đặt hàng sản phẩm, nhà phát triển (developer) là các thành viên của công ty có trách nhiệm phát triển phần mềm đã được đặt hàng nhằm quán

xuyến toàn bộ các công việc hoặc có trách nhiệm một phần như thiết kế, cài đặt,... Các dạng quan hệ giữa khách hàng và nhà phát triển bao gồm dạng phần mềm nội bộ (internal software) và phần mềm hợp đồng (contract software). Người sử dụng (user) là một hay nhiều cá nhân thay mặt khách hàng để sử dụng sản phẩm. Phát triển phần mềm (software development) bao gồm tất cả các công việc tạo ra sản phẩm trước khi nó được chuyển sang giai đoạn bảo trì.



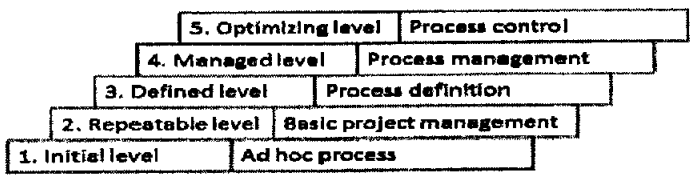
Hình 5: Tiến trình phần mềm với kiểm soát quản lý.



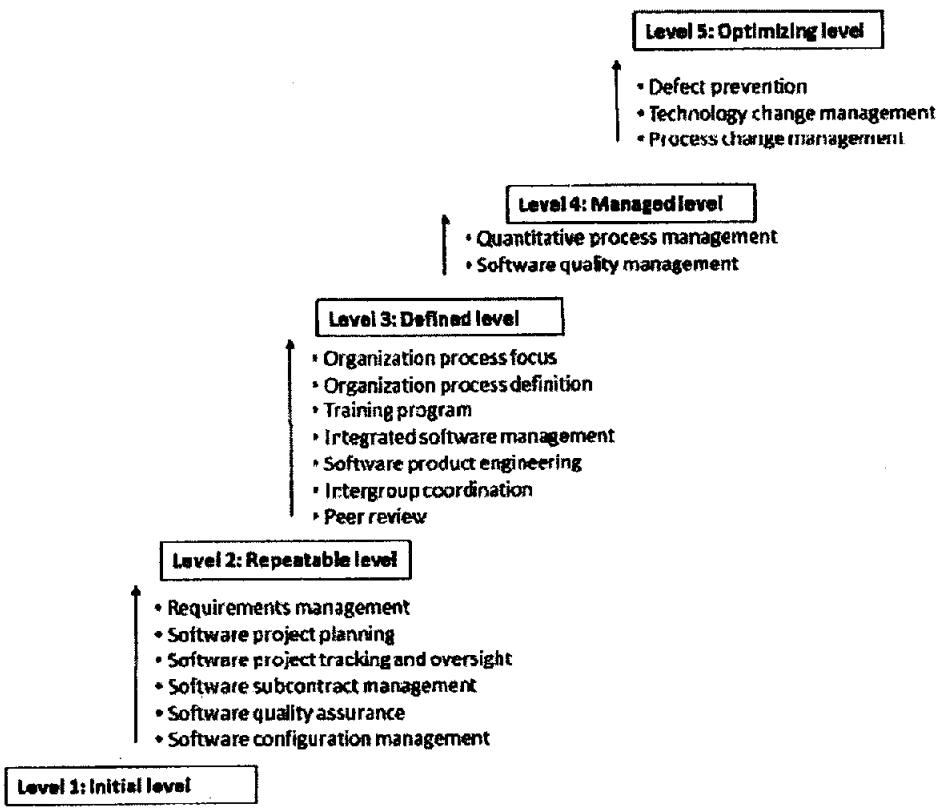
Hình 6: Các giai đoạn trong chu trình sống của phần mềm.

Cần quan tâm đến một số khía cạnh trong sản xuất phần mềm. Độ phức tạp (complexity) là một thuộc tính của phần mềm tác động trên tiến trình phần mềm và cả công tác quản lý tiến trình, có thể biểu diễn bằng toán học và vật lý và ảnh hưởng đến công tác bảo trì. Sự thích ứng (conformity) xác định phần mềm phải thích ứng được với các thiết bị sẵn có (không phải các thiết bị đáp ứng phần mềm), thích ứng tốt với phần cứng phục vụ phần mềm. Dễ chuyển đổi (changeability) xác định phần mềm phải thay đổi theo thực tiễn, mở rộng các chức năng ban đầu với thực tế là thay đổi phần mềm dễ hơn thay đổi về phần cứng và sau đó phần cứng thay đổi theo sự phát triển của phần mềm hoặc công

nghệ. Tính vô hình (invisibility) nhằm giấu các công đoạn phức tạp khi thực hiện phần mềm, dễ dàng thuyết minh, thuyết phục khách hàng, dễ dàng giao tiếp giữa các bộ phận thực hiện phần mềm, sử dụng các phương pháp, công cụ trực quan sinh động. Việc nhanh chóng tạo phần mềm mới từ các bộ phận hay công cụ có sẵn (silver bullet) là nhằm nhanh chóng chuyển đổi chức năng của sản phẩm, giảm thời gian cũng như chi phí thực hiện phần mềm, sử dụng mô hình chuyển đổi nhanh. Cần chú ý thuật ngữ lỗi trên 1000 dòng mã lệnh tương đương hợp ngữ (faults per million equivalent assembler source – MEASL).



Hình 7: Năm mức của CMM.



Hình 8: Nội dung chi tiết của năm mức trong CMM.

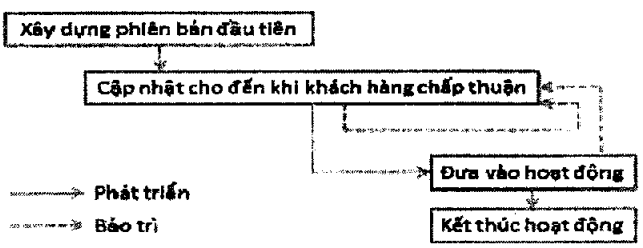
Cải tiến tiến trình phần mềm với mô hình khả trưởng CMM (Capability Maturity Models) là nhóm các chiến lược liên quan với nhau nhằm cải tiến tiến trình phần mềm. Có các nội dung cho phần mềm SW-CMM (Software), cho quản lý nguồn nhân lực P-CMM (People), cho công nghệ hệ thống SE-CMM (System Engineering), cho phát triển sản phẩm tích hợp IPD-CMM (Integrated Product Development) và cho đạt được sản phẩm SA-CMM (Software Aquisition).

CMMs là tiêu chuẩn khởi điểm cho các tiêu chuẩn quốc tế về sau như ISO 9000 (International Standards Organization 9000-series standards) gồm 5 chuẩn áp dụng rộng rãi cho các hoạt động công nghiệp: thiết kế (design), phát triển (development), sản xuất (production), cài đặt (installation) và bảo dưỡng (servicing). ISO 9001 dành cho chất lượng sản phẩm [ISO 9001, 1987], áp dụng ISO 9001 cho phần mềm ISO 9000-3 [ISO 9000-3, 1991]; SPICE (Software Process Improvement Capability dEtermination) tương tự như SW-CMM và ISO 9000 thống nhất hai chuẩn từ 06/1997: ISO/IEC 15504 hay 15504.

3. CÁC MÔ HÌNH CHU TRÌNH SỐNG CỦA PHẦN MỀM

3.1 Mô hình xây dựng và hiệu chỉnh

Mô hình xây dựng và hiệu chỉnh (code-and-fix model) không có đặc tả hay thiết kế, chỉ đơn giản là làm đi làm lại cho đến khi nào đáp ứng được yêu cầu của khách hàng. Thường sử dụng trong các bài tập lập trình từ 100 đến 200 dòng mã lệnh.

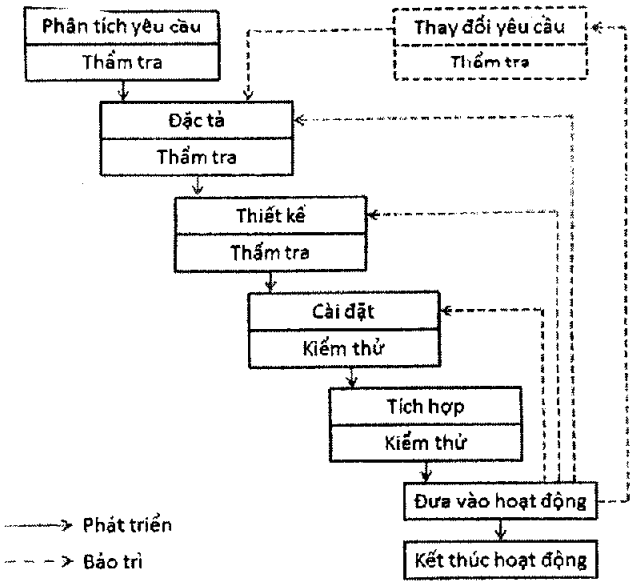


Hình 9: Mô hình xây dựng và hiệu chỉnh.

3.2 Mô hình thác nước

Mô hình thác nước (waterfall model) cho phép các lỗi ở một số giai đoạn trước được phản hồi bởi các giai đoạn sau và mỗi giai đoạn chỉ được xem là hoàn thành sau khi đã có đầy đủ tài liệu cho giai đoạn đó và được nhóm đảm bảo chất lượng phần mềm (SQA) chấp thuận. Các bước tiến hành bắt đầu với các yêu cầu được xác định và kiểm chứng bởi khách hàng và nhóm SQA, các đặc tả được kiểm chứng bởi nhóm SQA và gửi cho khách hàng, khung kế hoạch quản lý dự án phần mềm (SPMP) và bảng thời gian làm việc chi tiết được lập. Giai đoạn thiết kế bắt đầu sau khi khách hàng đồng ý về giá thành và thời gian thực hiện, thực

hiện cài đặt và tích hợp, khách hàng cho hoạt động thử; chấp nhận sản phẩm, chuyển sang giai đoạn bảo trì. Mô hình này có ưu điểm là tính kỷ luật cao, quy định tốt về tài liệu cho mỗi giai đoạn, kiểm chứng cẩn thận bởi nhóm SQA, được ứng dụng rộng rãi. Nhược điểm của mô hình là quá nhiều kiểm thử, thẩm tra và tài liệu nên khó hình dung và khó hiểu đối với khách hàng.



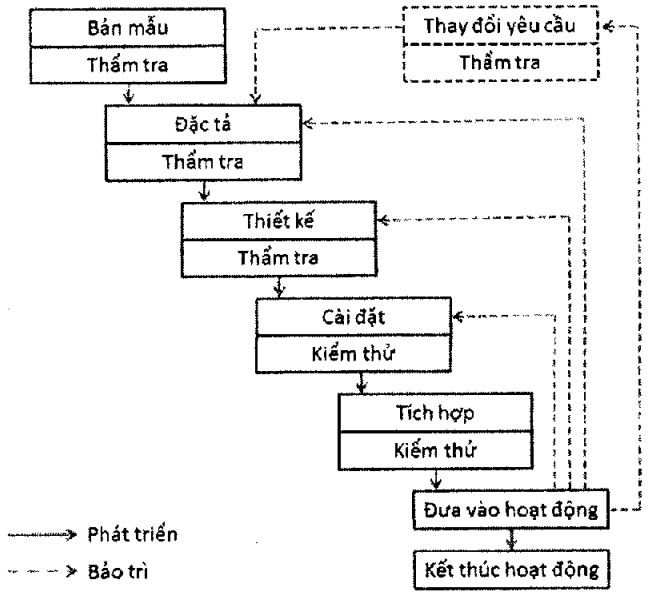
Hình 10: *Mô hình thác nước.*

3.3 Mô hình bản mẫu

Mô hình bản mẫu (rapid prototyping model) là mô hình hoạt động có chức năng tương đương với một tập hợp con của sản phẩm, được sử dụng khi cần thực hiện một cách nhanh chóng một sản phẩm phần mềm trên cơ sở phần mềm tương tự hay phần mềm mà nhóm thực hiện đã có nhiều kinh nghiệm. Chẳng hạn như nếu chức năng sản phẩm là trả tiền tài khoản, nhận tiền từ tài khoản và xếp hàng vào kho thì việc tạo nhanh bản mẫu có thể bao gồm các công việc của sản phẩm như: màn hình nhập liệu, in các báo cáo nhưng không có các công việc như cập nhật tập tin hay bắt các lỗi xuất hiện.

Đầu tiên là xây dựng bản mẫu của mô hình, tạo điều kiện cho khách hàng và người sử dụng tương lai tương tác với mô hình và thử nghiệm, chuyển sang giai đoạn đặc tả sau khi khách hàng đã chấp thuận rằng các yêu cầu cần thiết đã có trong quá trình xây dựng nhanh bản mẫu. Yêu cầu của mô hình là thực hiện càng nhanh càng tốt để tăng tốc độ của tiến trình phát triển phần mềm.

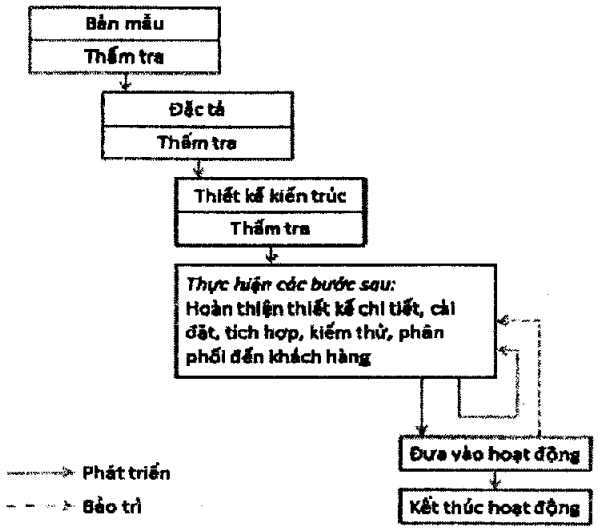
Có thể tích hợp hai mô hình thác nước và bản mẫu bằng cách xem việc xây dựng mô hình bản mẫu là đầu vào của mô hình thác nước. Có thể xảy ra một số hiệu ứng lề và có thể có rủi ro xuất hiện do sử dụng nhiều mô hình (số lượng ở đây là 2).



Hình 11: Mô hình bản mẫu.

3.4 Mô hình tăng trưởng

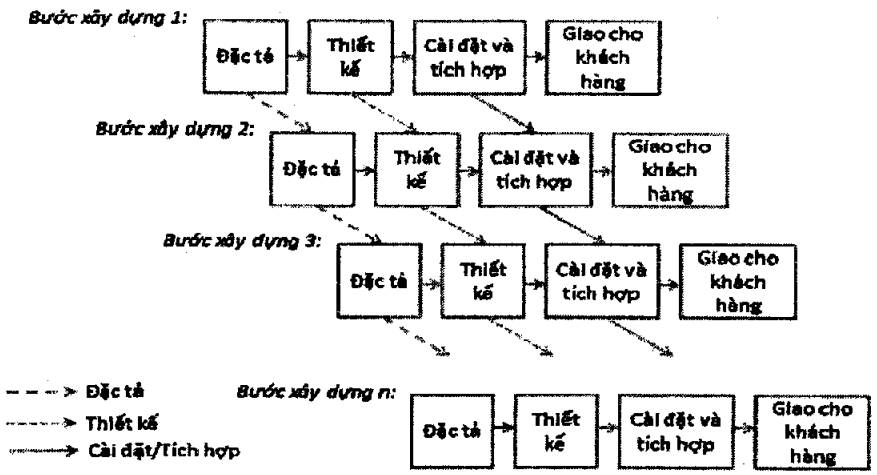
Mô hình tăng trưởng (incremental model) bao gồm chuỗi các bước thiết kế, cài đặt, tích hợp và kiểm thử được thực hiện liên tục (tăng). Mô hình này thường sử dụng trong một số dự án về không gian.



Hình 12: Mô hình tăng trưởng.

3.5 Mô hình tăng trưởng nhiều rủi ro

Mô hình tăng trưởng nhiều rủi ro (risk incremental model) sẽ giao sản phẩm cho khách hàng sau mỗi bước xây dựng. Mỗi bước xây dựng tương đương với một tập con các yêu cầu của khách hàng. Một sản phẩm điển hình thường bao gồm khoảng 10-50 bước xây dựng. Ưu điểm là giảm khó chịu cho khách hàng khi phải thay đổi một sản phẩm hoàn chỉnh. Cần chú ý việc phá bỏ các cấu trúc của bước xây dựng trước đó.



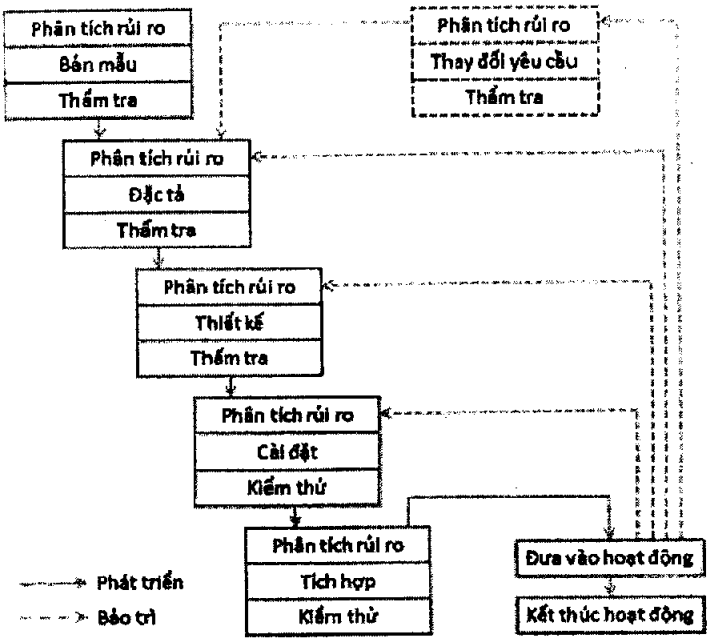
Hình 13: Mô hình tăng trưởng nhiều rủi ro.

3.6 Mô hình đồng bộ và ổn định

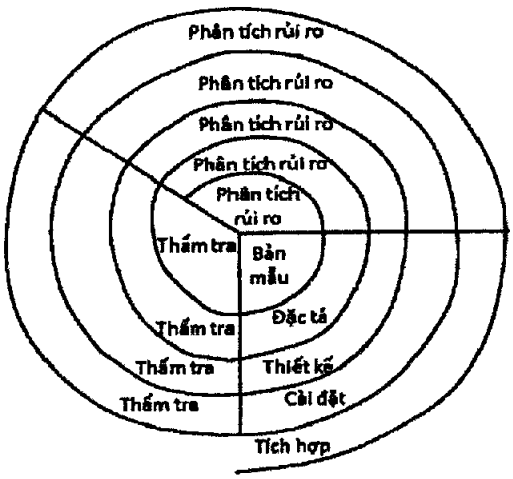
Mô hình đồng bộ và ổn định (synchronize-and-stabilize model) là một dạng khác của mô hình tăng trưởng được triển khai sử dụng tại công ty Microsoft, Inc. Các bước thực hiện bao gồm việc dẫn dắt các phân tích yêu cầu bằng cách phỏng vấn đông đảo các khách hàng tiềm năng (potential customers), rút ra tài liệu đặc tả, công việc được chia thành 3 hay 4 bước xây dựng (đặc điểm cấp thiết nhất, đặc điểm cấp thiết nhì,....), mỗi bước xây dựng được thực hiện cùng lúc bởi nhiều nhóm nhỏ, cuối mỗi ngày các nhóm thực hiện đồng bộ với nhau bằng cách ghép các phần việc của nhóm lại với nhau, kiểm thử và lần vết trên sản phẩm kết quả, hiệu chỉnh các lỗi và bước xây dựng chuyển sang trạng thái ổn định, sẽ không có bất kỳ thay đổi nào nữa trên tài liệu đặc tả. Lặp lại bước đồng bộ.

3.7 Mô hình xoắn ốc

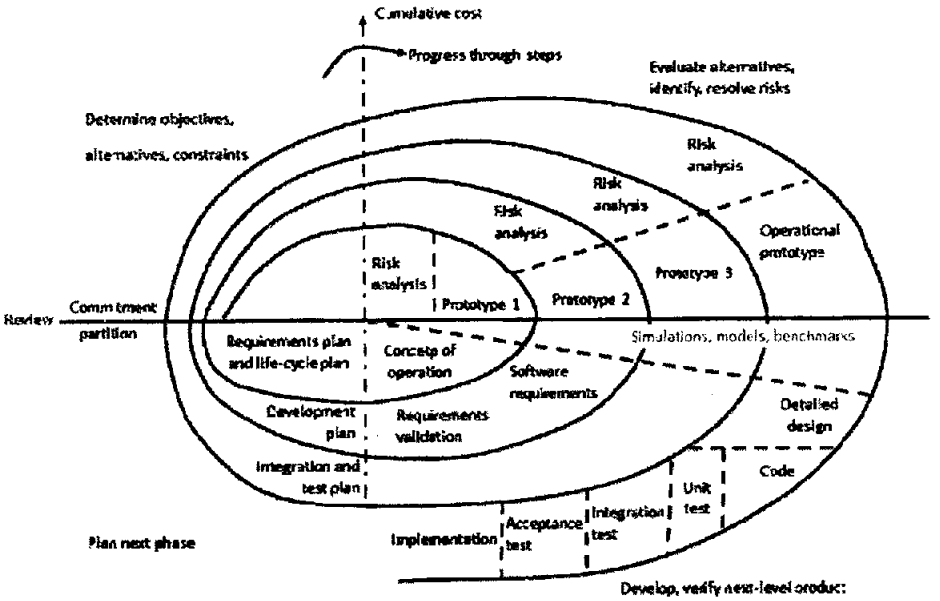
Mô hình xoắn ốc (spiral model) được hình thành trên cơ sở xác định yếu tố rủi ro hầu như luôn tồn tại trong sự phát triển của phần mềm và nhằm nhằm giảm thiểu sự rủi ro trong quá trình phát triển. Được sử dụng rộng rãi cho một lớp rộng các sản phẩm và gặt hái nhiều thành công. Điểm mạnh của mô hình hướng rủi ro (risk-driven) là các công việc luân phiên và chịu các ràng buộc đã hỗ trợ cho việc tái sử dụng phần mềm hiện có, đánh giá được mức độ rủi ro, xác định mục tiêu quan trọng luôn là chất lượng phần mềm đồng thời giảm nhẹ kiểm thử và nhanh chóng sửa chữa những lỗi xảy ra. Bảo trì đơn giản chỉ là một vòng tròn trong xoắn ốc, như vậy không có sự phân biệt giữa phát triển và bảo trì. Nhược điểm của mô hình hướng rủi ro xảy ra khi tiến hành như thế nào nếu có một thành phần có độ rủi ro cao hoặc là kích thước sản phẩm ảnh hưởng đến giá thành việc phân tích rủi ro? Mô hình này thường dành riêng cho các phần mềm nội bộ có kích thước lớn.



Hình 14: Mô hình xoắn ốc đơn giản.



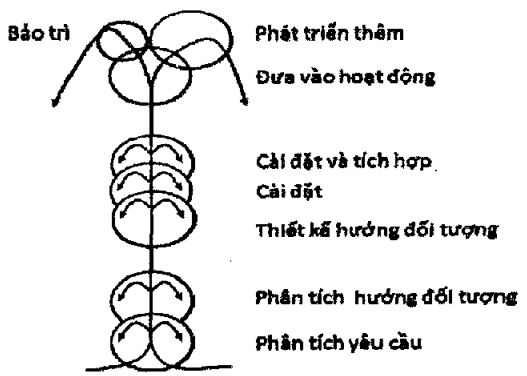
Hình 15: Mô hình xoắn ốc đơn giản (trình bày lại một phần).



Hình 16: Mô hình xoắn ốc đầy đủ.

3.8 Mô hình vòi phun nước

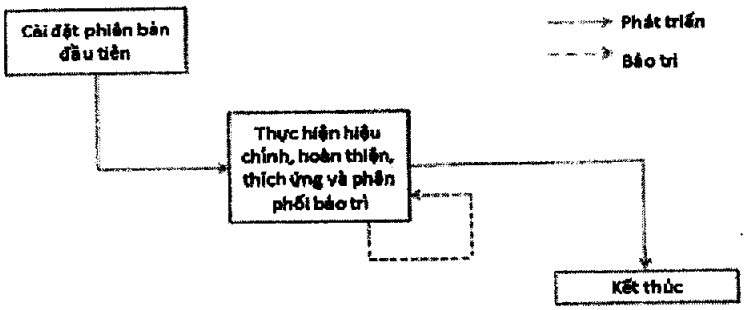
Mô hình vòi phun nước có đặc tính quan trọng nhất là lặp giữa các giai đoạn và lặp một phần trong giai đoạn. Các vòng tròn thể hiện các giai đoạn gối đầu lên nhau, phần thấy được phản ánh sự gối lên trên giữa các hoạt động, mũi tên bên trong một giai đoạn thể hiện sự lặp lại bên trong giai đoạn đó, vòng tròn bảo trì nhỏ hơn tượng trưng cho việc giảm bớt nhân lực cho công tác bảo trì. Đây là mô hình đại diện cho các mô hình chu trình sống phần mềm dạng hướng đối tượng.



Hình 17: Mô hình vòi phun nước.

3.9 Mô hình mã nguồn mở

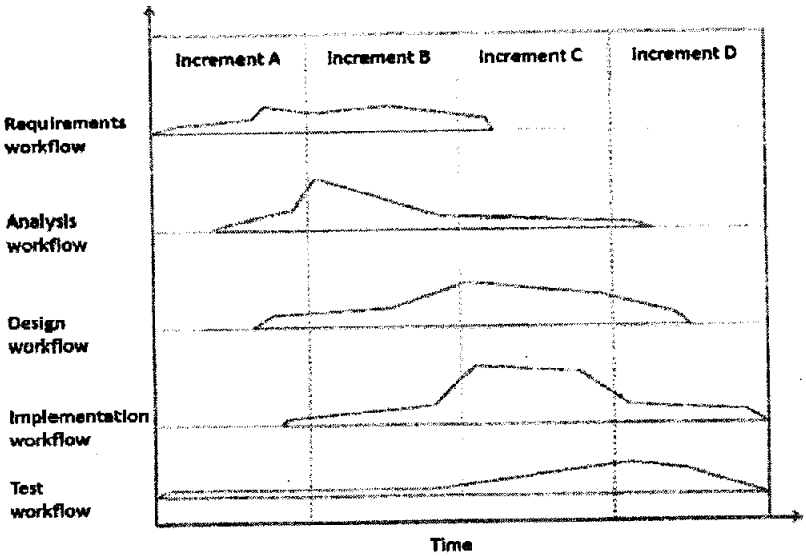
Mô hình mã nguồn mở (open source model) thường được hình thành trên hai giai đoạn không hình thức. Ở giai đoạn đầu tiên một cá nhân đưa ra ý kiến cho một chương trình và xây dựng phiên bản đầu tiên và phiên bản này được phát hành miễn phí. Với vai trò là người đồng phát triển (co-developer), người sử dụng của các phiên bản đầu tiên này sẽ chỉ ra các nhược điểm và đề xuất hướng phát triển, sữa chữa các lỗi và mở rộng chương trình. Những người khác dựa trên các ý tưởng đã được đề xuất sẽ cài đặt tạo nên sự phát triển của sản phẩm phần mềm.



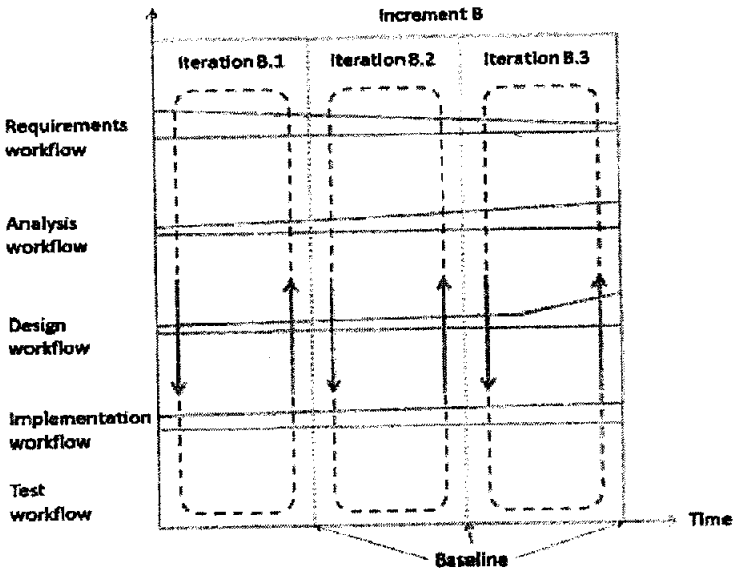
Hình 18: Mô hình mã nguồn mở.

3.10 Mô hình lặp và tăng trưởng

Mô hình lặp và tăng trưởng (iterative-and-incremental model) tạo điều kiện phát triển phần mềm có thể được chia ra thành các thành phần tăng trưởng trong khi các công việc (tác vụ) có thể được phát triển theo quy trình lặp. Xuất phát điểm ban đầu với các yêu cầu, các yêu cầu này có thể được mở rộng và cập nhật trong quá trình phát triển còn lại của phần mềm. Trong mỗi giai đoạn tăng trưởng, các dòng công việc (phân tích, thiết kế, cài đặt và kiểm thử) được thực hiện một cách nghiêm chỉnh. Trong mỗi quá trình tăng trưởng khác nhau thì mức độ quan trọng của các công việc trong dòng công việc sẽ có sự thay đổi khác nhau tùy theo tính chất, tình hình của giai đoạn tăng trưởng đó.



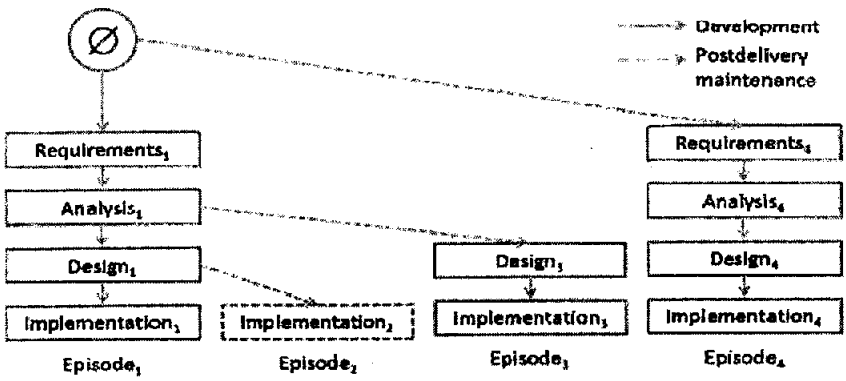
Hình 19: Mô hình lặp và tăng trưởng.



Hình 20: Ba bước lặp của quá trình tăng trưởng B.

3.11 Mô hình cây tiến hóa

Mô hình cây tiến hóa (evolution-tree model) thực hiện các bước phát triển phần mềm đồng thời trong quá trình phần mềm được phát triển và bảo trì. Mô hình này được phát triển dựa trên mô hình lặp và tăng trưởng. Quá trình phát triển dựa trên các tập tác vụ cơ sở (baseline), hay còn được gọi là các mốc công việc theo thời đoạn (episode). Mô hình này thường được sử dụng trong giai đoạn bảo trì.



Hình 21: Mô hình cây tiến hóa.

3.12 Mô hình phát triển nhanh

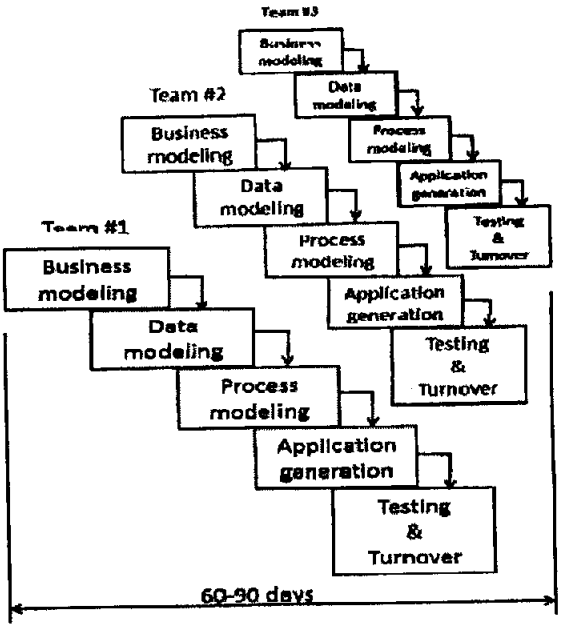
Mô hình phát triển nhanh (extreme programming and agile processes) là một tiếp cận mới về phát triển phần mềm đang được thảo luận rộng rãi. Tiếp cận này dựa trên mô hình lặp và tăng trưởng. Nhóm phát triển phần mềm đầu tiên sẽ xác định các đặc điểm khác nhau mà khách hàng mong muốn sản phẩm hỗ trợ (stories). Đối với mỗi đặc điểm như vậy nhóm phát triển sẽ cài đặt và xác định chi phí cần thiết. Bước đầu tiên này tương ứng với giai đoạn yêu cầu và phân tích (tham khảo quy trình RUP ở chương 3).

Đây là một trong những cách tiếp cận mới được tập hợp lại dưới tên "tiến trình nhanh" (agile process) trên cơ sở định hướng ít nhấn mạnh đến giai đoạn phân tích và thiết kế so với các giai đoạn khác và giai đoạn cài đặt được bắt đầu sớm hơn trong chu trình sống của phần mềm bởi vì làm việc trên phần mềm được xem trọng hơn tài liệu chi tiết. Ứng phó với thay đổi là một mục tiêu quan trọng khác của mô hình này đồng thời nâng cao tầm quan trọng của việc phối hợp với khách hàng.

3.13 Mô hình phát triển ứng dụng nhanh

Mô hình phát triển ứng dụng nhanh (rapid application development - RAD) là một tiến trình phát triển phần mềm dạng tăng trưởng nhấn mạnh đến chu trình phát triển cực ngắn. Mô hình này dựa trên việc sử dụng các thành phần

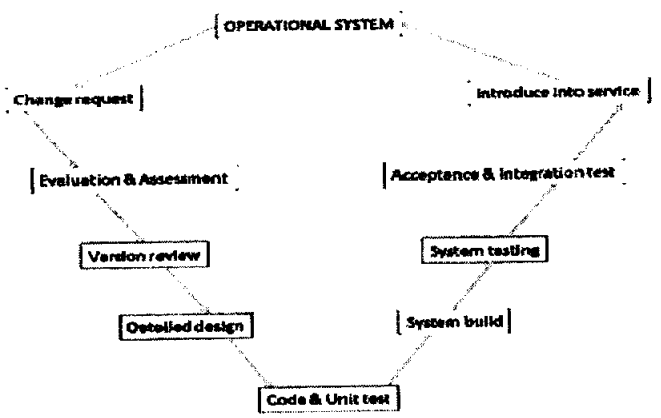
(component-based) để xây dựng phần mềm tương đối đầy đủ các chức năng với thời gian khoảng từ 60 đến 90 ngày.



Hình 22: Mô hình phát triển ứng dụng nhanh.

3.14 Mô hình V

Mô hình V (V model) là một mô hình cải tiến từ mô hình thác nước với các khái niệm rất gần nhau. Điểm khác biệt chính là chu trình phát triển phần mềm không có điểm bắt đầu và điểm kết thúc mà tạo thành một chu trình.



Hình 23: Mô hình V.

4. TỔ CHỨC NHÓM LÀM VIỆC

Sản phẩm phần mềm chỉ có thể thành công khi người tham gia thực hiện thành thạo, hiểu biết về công nghệ phần mềm và được đào tạo tốt về công nghệ phần mềm. Ngoài con người tốt, các nhóm làm việc cũng phải được tổ chức nhằm làm cho các thành viên làm việc hiệu quả và kết hợp chặt chẽ với nhau. Các sản phẩm tương đối lớn trở đi phải do những người chuyên nghiệp thực hiện và những người này được tổ chức thành nhóm làm việc (team). Vấn đề đặt ra là sản phẩm nếu do một người thực hiện sẽ hoàn thành trong một năm, bốn người thực hiện sẽ hoàn thành trong ba tháng? Cần chú ý luật Brooks đã chỉ ra rằng thêm nhân lực cho một sản phẩm phần mềm đang thực hiện sẽ làm chậm tiến độ thực hiện của nó. Các giai đoạn đều có nhóm làm việc riêng nhưng vai trò đặc biệt thuộc về nhóm cài đặt (mỗi người làm việc trên một mô đun riêng).

4.1 Nhóm làm việc dân chủ

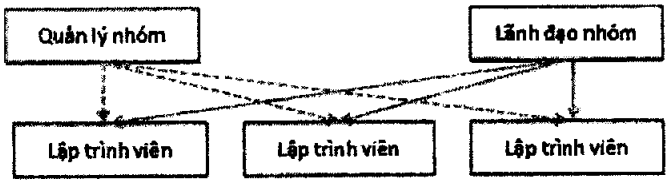
Tiếp cận về *nhóm làm việc dân chủ* với các thành viên có vai trò như nhau trên cơ sở tiếp cận về lập trình viên bản ngã. Đặc điểm của lập trình viên bản ngã là sự gắn bó cao với mã lệnh của họ, các mô đun như là sự mở rộng của chính bản thân nên khó phát hiện lỗi. Do đó cần thiết cấu trúc lại môi trường xã hội theo các giá trị của lập trình viên và khuyến khích các thành viên khác trong nhóm tìm kiếm lỗi trong các mã lệnh của mình nhằm thể hiện tinh thần tập thể cao. Nhóm làm việc dân chủ (democratic team) nên có tối đa là 10 lập trình viên bản ngã.

4.2 Nhóm làm việc cổ điển

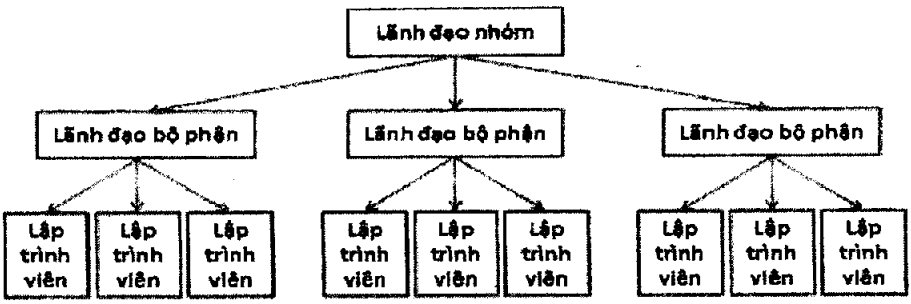
Tiếp cận về *nhóm làm việc cổ điển* bao gồm trưởng nhóm lập trình (quản lý tốt, giỏi lập trình, xử lý các công việc khó khăn khác), lập trình viên hỗ trợ (sẵn sàng thay thế trưởng nhóm quán xuyến các công việc khi cần), thư ký lập trình (bảo trì thư viện, tài liệu, danh sách mã nguồn, dữ liệu kiểm thử,...) và các lập trình viên. Nhóm lập trình sẽ yêu cầu sự trợ giúp của các chuyên gia luật, tài chính,...trong các vấn đề liên quan. Hiện nay rất khó tìm được trưởng nhóm có khả năng "tuyệt vời" như cấu trúc nhóm cổ điển.

4.3 Nhóm làm việc hiện đại

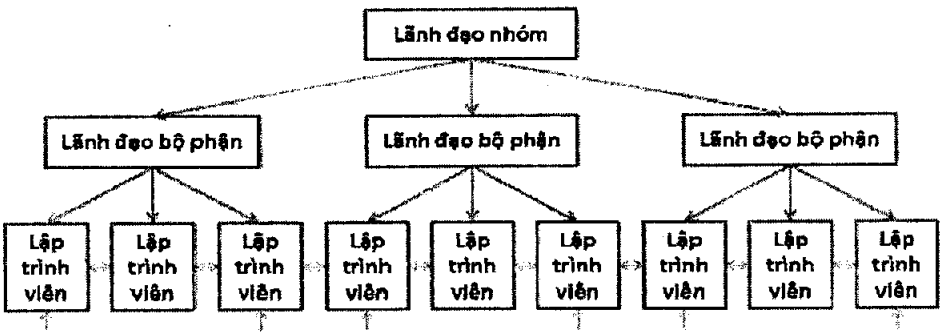
Tiếp cận về *nhóm làm việc hiện đại*, tách trưởng nhóm thành hai cá thể là lãnh đạo (leader) về các vấn đề kỹ thuật và quản lý (manager) về các vấn đề không thuộc về kỹ thuật.



Hình 24: Cấu trúc nhóm lập trình hiện đại.



Hình 25: Cấu trúc tổ chức quản lý kỹ thuật cho các dự án lớn.



Hình 26: Cấu trúc tổ chức quản lý kỹ thuật cho các dự án lớn với việc phi tập trung hóa các quyết định và các kênh giao tiếp kỹ thuật.

4.4 Nhóm làm việc đồng bộ hóa và ổn định

Tiếp cận về *nhóm làm việc dạng đồng bộ hóa và ổn định* được xây dựng trên cơ sở cứ mỗi 3-4 bước xây dựng được thực hiện song song bởi nhiều nhóm nhỏ. Tổ chức của một nhóm nhỏ gồm có một quản lý chương trình, 3-8 nhà phát triển, 3-8 kiểm thử viên (tương ứng 1-1 với số lượng nhà phát triển). Công việc của một nhóm nhỏ được hình thành với tài liệu đặc tả được cung cấp cho toàn bộ công việc của nhóm và mỗi thành viên trong nhóm được tự do thiết kế và cài đặt phần làm việc của mình. Với cách tổ chức này các lập trình viên luôn sáng tạo và đổi

mới, hướng cùng mục đích nhưng cần phải tôn trọng triệt để thời gian đưa mã nguồn vào cơ sở dữ liệu của sản phẩm để đồng bộ hóa.

4.5 Nhóm làm việc nhanh

Tiếp cận về *nhóm làm việc nhanh* được tổ chức trên cơ sở hai lập trình viên làm việc trên cùng một máy tính đơn. Cách tổ chức này còn được gọi là lập trình cặp (pair programming), thường được sử dụng trong các mô hình phát triển phần mềm cực độ (extreme programming) hay mô hình phát triển nhanh (agile processes).

4.6 Nhóm làm việc mã nguồn mở

Tiếp cận về *nhóm làm việc mã nguồn mở* được thực hiện một cách tự nguyện không có quản lý cũng như không có hội họp và liên lạc với nhau một cách không đồng bộ. Hơn nữa không có đặc tả cũng như thiết kế, tài liệu cũng rất ít. Các thành viên làm việc trên cơ sở tìm kiếm kinh nghiệm cho bản thân và làm việc vì công việc có lợi ich chung.

4.7 Nhóm làm việc P-CMM

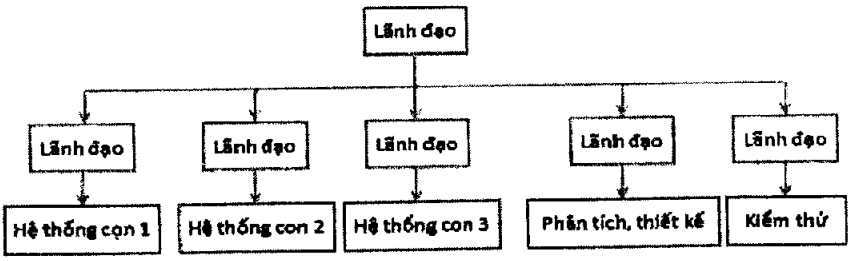
Tiếp cận về *nhóm làm việc P-CMM* sẽ giúp cho thực hiện tốt việc quản lý và phát triển năng lực của một tổ chức thông qua việc thực hiện năm mức tăng trưởng.

4.8 Nhóm làm việc SWAT

Tiếp cận *nhóm SWAT (Skilled With Advanced Team)* được thiết lập với 4 hoặc 5 thành viên với trách nhiệm cao trong công việc và cả mối quan hệ, thường được bố trí cùng làm việc trong cùng một phòng làm việc. Nhóm SWAT thường được hình thành khi phát triển các phần mềm với mô hình sống dạng tăng trưởng, trưởng nhóm làm việc kiêm cả công việc quản lý.

4.9 Nhóm làm việc phân cấp

Tiếp cận về *nhóm làm việc phân cấp* với các bộ phận là các hệ thống con và các bộ phận chuyên môn (kiểm thử, phân tích,...).



Hình 27: Cấu trúc tổ chức quản lý phân cấp.

4.10 Nhóm làm việc ma trận

Tiếp cận về *nhóm làm việc dạng ma trận* với các phần mềm khác nhau.

	Cài đặt	Đồ họa	Cơ sở dữ liệu	Phân tích, thiết kế	Kiểm thử
Phần mềm 1	X	X		X	X
Phần mềm 2		X	X	X	X
Phần mềm 3			X	X	X

Hình 28: Cấu trúc tổ chức quản lý dạng ma trận.

4.11 Nhóm làm việc cấu trúc mở

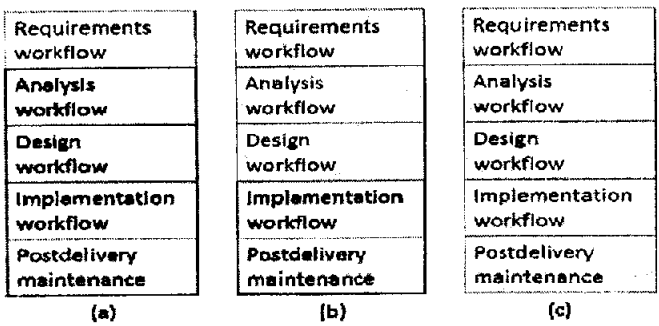
Tiếp cận *nhóm làm việc cấu trúc mở* được hình thành khi thực hiện các phần mềm phức tạp với cấu trúc quản lý mở và quan tâm nhiều đến các chiều quan hệ. Nhóm làm việc cố gắng tổ hợp sự quan tâm đến từng thành viên và cam kết của thành viên đó với mục tiêu của dự án, tập trung vào các hoạt động để đảm bảo đạt được các mục tiêu của sản xuất phần mềm đúng thời gian và hiệu quả.

5. CÔNG CỤ PHẦN MỀM

Công nghệ phần mềm cần hai dạng công cụ: (i) phân tích - dùng trong phát triển phần mềm (công cụ phân tích giá thành, lợi nhuận; công cụ phân tích mịn dần,...); (ii) phần mềm, các sản phẩm trợ giúp các nhóm công nghệ phần mềm trong phát triển và bảo trì phần mềm, thường gọi là các công cụ CASE (Computer-Aided Software Engineering tools - CASE tools). Việc sử dụng công cụ hỗ trợ trong quá trình phát triển phần mềm sẽ trợ giúp một mặt nào đó trong sản xuất phần mềm.

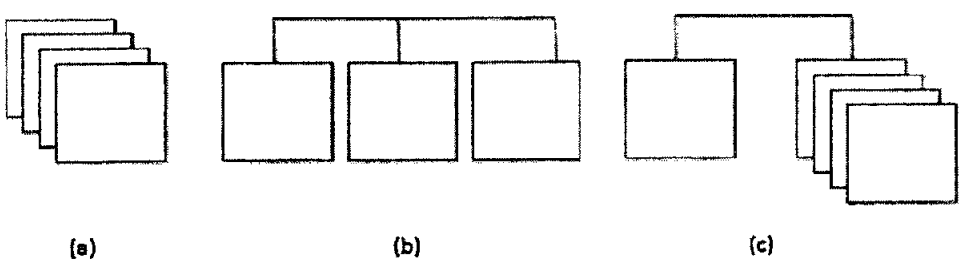
Công cụ (tools) nhằm trợ giúp một giai đoạn trong các giai đoạn đầu tiên (upperCASE hay front-end) như phân tích yêu cầu, đặc tả và thiết kế; hoặc trợ giúp trong các giai đoạn cuối (lowerCASE hay back-end) như cài đặt, tích hợp và bảo trì. Nhóm công cụ (workbench) là các công cụ thể hiện bằng đồ họa với từ điển dữ liệu, kiểm chứng tính chắc chắn, sinh báo cáo, sinh màn hình cho các giai

đoạn đặc tả và thiết kế (PowerBuilder, System Architect,...) hỗ trợ một hoặc hai hoạt động. Hoạt động bao gồm nhiều công việc (chẳng hạn như hoạt động mã hóa có các công việc: viết, nối kết, kiểm thử và dò lỗi). Môi trường (environment) là công cụ hỗ trợ đầy đủ tiến trình phần mềm. Từ điển dữ liệu (data dictionary) là danh sách các định nghĩa dữ liệu có trong sản phẩm.



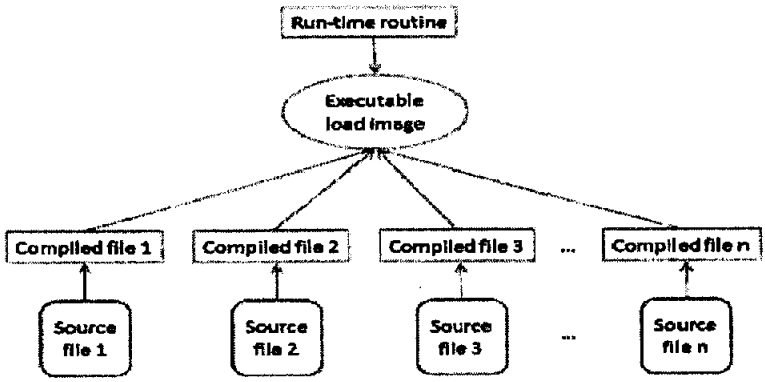
Hình 29: *(a) Công cụ, (b) nhóm công cụ, (c) môi trường.*

Một phiên bản mới của phần mềm (new version) sẽ ra đời mỗi khi bảo trì sản phẩm. Phiên bản đã sửa chữa (revisions) là phiên bản mới sau khi đã được hiệu chỉnh ít nhất một lỗi. Gọi phiên bản đã sửa chữa trước là n thì phiên bản đã sửa chữa sau sẽ là n+1. Phiên bản khác nhau (variations) là các phần mềm được cài đặt trên các hệ điều hành hay các loại phần cứng khác nhau,...(chẳng hạn như các trình điều khiển thiết bị cho cùng một loại máy in được xây dựng cho các hệ điều hành khác nhau, phần mềm được triển khai trên các hệ điều hành khác nhau).



Hình 30: *Các dạng phiên bản khác nhau (a) revisions (b) variations (c) mixte.*

Kiểm soát cấu hình (configuration control) là kiểm soát các phiên bản khác nhau của từng tác vụ. Phiên bản cụ thể của mỗi tác vụ tham gia vào phiên bản cho trước của sản phẩm khi biên dịch được gọi là cấu hình (configuration) của phiên bản đó.



Hình 31: Các thành phần cấu hình của một chương trình thực thi.

6. KIỂM THỬ

Đầu tiên cần phân biệt lỗi (fault) là thiếu sót về mặt kỹ thuật và lỗi (error) là tạo ra bởi người lập trình [IEEE 610.12, 1990]. Các khái niệm khác cần nắm rõ như thẩm tra (verification) và công nhận hợp lệ (validation).

Một sản phẩm được gọi là chất lượng (quality) nếu như sản phẩm đáp ứng chính xác đặc tả của nó. Đảm bảo chất lượng phần mềm (Software Quality Assurance - SQA) bao gồm việc thành lập nhóm đảm bảo chất lượng phần mềm SQA nhằm đảm bảo sản phẩm hoạt động đúng chức năng và kiểm tra mỗi khi các nhà phát triển hoàn thành một giai đoạn nào đó. Nhóm SQA cần phải được độc lập về quản lý trên cơ sở nhóm SQA và nhóm phát triển phải được quản lý độc lập với nhau, không can thiệp vào công việc của nhau. Các tiêu chí về chất lượng bao gồm: chính xác (correctness), tin cậy (reliability), hiệu quả (efficiency), toàn vẹn (integrity), khả năng sử dụng (usability), bảo trì (maintainability), khả năng kiểm thử (testability), mềm dẻo (flexibility), dễ chuyển đổi (portability), khả năng sử dụng lại (reusability) và vận hành tương tác (interoperability).

6.1 Kiểm thử không dựa trên thực thi

Kiểm thử không dựa trên thực thi với tiếp cận nhóm kiểm thử walkthrough khoảng 4-6 người với thành phần có ít nhất một đại diện thuộc nhóm đặc tả, một nhà quản lý chịu trách nhiệm về nhóm đặc tả, một đại diện khách hàng, một đại diện của nhóm thực hiện giai đoạn kế tiếp, một đại diện của nhóm SQA, làm trưởng nhóm walkthrough. Nhóm walkthrough nên chọn những người già dặn kinh nghiệm kỹ thuật. Có hai cách thực hiện quản lý nhóm walkthrough theo hướng thành viên (mỗi thành viên trong nhóm ra danh sách chất vấn có các mục không rõ ràng hoặc không chính xác theo quan điểm của mình, đại diện nhóm đặc tả giải trình) hoặc theo hướng tài liệu (người có trách nhiệm về tài liệu giải trình từng phần trong tài liệu cho nhóm đưa ra ý kiến).

Thanh tra (inspection) cũng là một hướng kiểm thử không dựa trên thực thi. Nhóm thanh tra thường có khoảng 4 người với nhóm trưởng, người thiết kế, người cài đặt và người kiểm thử thuộc nhóm SQA. Nếu nhóm thanh tra có khoảng 3-6 người thì vai trò sẽ phân bổ như sau với nhóm trưởng, người dẫn dắt nhóm phần thiết kế và người viết báo cáo lỗi. Cần chú ý thiết lập danh sách các lỗi tiềm tàng.

Việc thực hiện thanh tra nhằm kiểm thử các thiết kế và mã lệnh, gồm 5 bước: xem xét khái quát (các tài liệu sẽ được thanh tra như đặc tả, thiết kế, mã lệnh, kế hoạch,... được đưa ra bởi chính người viết tài liệu đó; tất cả các thành viên trong nhóm sẽ nhận đầy đủ các tài liệu), chuẩn bị (các thành viên tìm hiểu các tài liệu một cách chi tiết; danh sách các lỗi trong các lần thanh tra gần nhất), thanh tra (một thành viên duyệt qua tất cả các mục và các nhánh trong tài liệu; phát hiện các lỗi; lãnh đạo nhóm thanh tra viết báo cáo về lỗi), làm lại (các cá nhân phụ trách các tài liệu sẽ sửa các lỗi được mô tả trong báo cáo về lỗi ở bước thanh tra), tiếp tục (nhóm trưởng đảm bảo rằng toàn bộ các tài liệu đã được điều chỉnh, giới thiệu lỗi).

Công tác thanh tra được đánh giá thông qua phương pháp tính mật độ lỗi (số lỗi trên một trang đặc tả hay thiết kế, số lỗi trên KDSI/một ngàn dòng mã lệnh), tính tỷ lệ phát hiện lỗi (số lượng lỗi quan trọng/không quan trọng trên một giờ) và tính hiệu suất dò tìm lỗi (số lượng lỗi quan trọng/không quan trọng trên người-giờ).

6.2 Kiểm thử dựa trên thực thi

Kiểm thử dựa trên thực thi là tiến trình suy xét dựa vào cách thức xử lý của sản phẩm trên cơ sở thực thi sản phẩm trong một môi trường đã biết với các đầu vào chọn lọc. Hệ mô phỏng là một mô hình hoạt động của môi trường sản phẩm. Các khái niệm cần quan tâm như tiện ích, độ tin cậy, sự mạnh mẽ, hiệu suất và sự đúng đắn (một sản phẩm được xem là đúng nếu như nó đáp ứng được những đặc tả đầu ra của nó, độc lập với tài nguyên máy tính và vận hành với những điều kiện cho phép).

7. MÔ ĐUN/ĐỐI TƯỢNG

Mô đun (module) là tập hợp của một hay nhiều câu lệnh kế tiếp nhau được đặt tên, các phần khác trong chương trình có thể kích hoạt với tên được đặt và có tập hợp các tên biến riêng biệt. Mô đun nhìn một cách đơn giản thì đó là một khối đơn các mã lệnh có thể kích hoạt giống như thủ tục, hàm hay phương thức.

7.1 Độ gắn kết

Độ gắn kết (cohesion level) là mức độ tương tác bên trong một mô đun. Các thể loại hay mức gắn kết theo mức độ từ "xấu" đến "tốt" của một mô đun là trùng

khớp (thực hiện nhiều hành động không liên quan đến nhau), luận lý (thực hiện chuỗi các hành động có liên quan với nhau, một trong số đó được chọn bởi mô đun gọi đến), thời gian (thực hiện chuỗi các hành động liên quan với nhau theo thời gian), thủ tục (thực hiện chuỗi các hành động liên quan với nhau theo các bước đúng trình tự phát triển sản phẩm), truyền thông (thực hiện chuỗi các hành động liên quan với nhau theo các bước đúng trình tự phát triển sản phẩm và nếu như mọi hành động đều được thực hiện trên dữ liệu giống nhau), chức năng (thực hiện một hành động hoặc nhận lấy một kết quả) và thông tin (thực hiện một số lượng các hành động, mỗi hành động có đầu vào riêng, mã lệnh độc lập và thực hiện trên dữ liệu giống nhau).

7.2 Độ nối kết

Độ nối kết (coupling level) là mức độ tương tác giữa hai mô đun, rất quan trọng trong đánh giá. Các dạng hay mức nối kết thường gặp theo mức độ từ "xấu" đến "tốt" là nội dung (có thể tham khảo trực tiếp nội dung của nhau), chung (truy xuất đến các dữ liệu toàn cục giống nhau), điều khiển (có thể gửi phần tử điều khiển cho nhau hay có thể điều khiển lẫn nhau), nhãn hiệu (tham số được gửi đi là một cấu trúc dữ liệu và mô đun được gọi chỉ thao tác trên một vài thành phần của cấu trúc dữ liệu đó) và dữ liệu (tất cả các tham số đều là các mục dữ liệu thuần nhất).

Các vấn đề cần quan tâm khác đối với một mô đun là bao gói dữ liệu nhằm để trừu tượng hóa, ẩn thông tin, hỗ trợ thừa kế, đa hình và liên kết động.

8. SỬ DỤNG LẠI, DỄ DI CHUYỂN VÀ VẬN HÀNH TƯƠNG TÁC

Sử dụng lại là việc lấy một bộ phận của sản phẩm này để phát triển thuận lợi sản phẩm khác (với chức năng khác). Bộ phận được sử dụng lại của một phần mềm có thể là một mô đun, một đoạn mã lệnh, một thiết kế, một phần hướng dẫn sử dụng, một tập dữ liệu kiểm thử, một ước lượng về thời gian và giá thành,.... Có hai dạng sử dụng lại là ngẫu nhiên (một số bộ phận của sản phẩm cũ vẫn được sử dụng cho sản phẩm mới) và thảo luận (bộ phận đang được thực hiện sẽ được sử dụng lại trong tương lai). Cần chú ý đến vấn đề bản ngã vì các nhà chuyên nghiệp thường viết các bộ phận từ đầu chứ không thích sử dụng lại của người khác, chất lượng của bộ phận sử dụng lại, việc phục hồi lại các bộ phận cũ hữu ích và giá thành cao khi sử dụng lại.

Một sản phẩm phần mềm được cho là dễ di chuyển nếu với chi phí không lớn lắm có thể thực thi được trên một máy tính mới thay vì phải viết lại từ đầu. Một số vấn đề cần quan tâm như việc không tương thích phần cứng, không tương thích hệ điều hành, không tương thích về số hoá phần mềm (16 bits hay 32 bits chẳng hạn), không tương thích trình biên dịch.

Dễ di chuyển là sự hợp tác qua lại trên các đối tượng mã lệnh từ nhiều nhà sản xuất phần mềm khác nhau, được viết trên nhiều ngôn ngữ lập trình khác nhau và thực thi trên nhiều hệ điều hành khác nhau. Một số kỹ thuật nhằm đạt được tính dễ di chuyển như hệ thống phần mềm dễ di chuyển, hệ thống phần mềm ứng dụng dễ di chuyển, dữ liệu dễ di chuyển. Các thành phần như OLE, COM, ActiveX, DCOM, COM+, COM3, CORBA,... là những ví dụ về hỗ trợ các phần mềm có được thuộc tính dễ di chuyển.

9. ĐÁNH GIÁ PHẦN MỀM

Còn gọi là đo (measurement) phần mềm hay tiến trình đánh giá (process metrics) sản phẩm phần mềm và thường được đánh giá theo từng giai đoạn. Có nhiều phương pháp đánh giá, nhìn chung có 5 yếu tố cơ bản sau là quan trọng nhất: kích thước (size) [LOC,...], giá thành (cost) [đơn vị tiền tệ], thời gian thực hiện (duration) [tháng], nhân lực (effort) [người-tháng], chất lượng (quality) [số lượng lỗi phát hiện được].

Ngoài ra cần quan tâm đến việc ước tính giá thành và lợi nhuận trong tương lai trên các nội dung về sự thay đổi trị giá của đồng tiền, giá thành phần cứng/phần mềm, chi phí thuê chuyên gia công nghệ phần mềm, cách tính giá trị sản phẩm như thế nào,... Hướng giải pháp nên tính theo mệnh giá của đồng tiền mạnh hoặc sử dụng chiến lược thích hợp để đánh giá.

Bên cạnh đó, cần quan tâm mạnh mẽ đến tiếp cận làm mịn dần trong quá trình thực hiện phần mềm. Đây là cách thức, quá trình trì hoãn các quyết định chi tiết càng lâu càng tốt nhằm có thể tập trung vào những vấn đề trọng tâm nhất.

TÀI LIỆU THAM KHẢO

[1] Julia H. Allen *et al.*, *Software Security Engineering*, Pearson Education, 2008.

[2] Barry W. Boehm, *Software Engineering*, IEEE Computer Society - Wiley, 2007.

[3] Alphonse Carlier, *Le développement du logiciel*, Hermes, 1995.

[4] Scott E. Donaldson and Stanley G. Siegel, *Successful Software Development (2nd edition)*, Prentice Hall, 2000.

[5] Christopher Harris-Jones, *Knowledge Based Systems Methods: A Practitioners' Guide*, Prentice Hall, 1995.

[6] Frank Hayes, "Chaos is back", Computerworld, http://www.computerworld.com/s/article/97283/Chaos_Is_Back, 2008.

[7] IEEE Std 828, *IEEE Standard for Software Configuration Management Plans*, The Institute of Electrical and Electronics Engineers, Inc., 1998.

[8] IEEE, *Guide to the Software Engineering Body of Knowledge - SWEBOK®*, IEEE Computer Society, 2004.

[9] IEEE Std 610.12-1990, *IEEE Standard Glossary of Software Engineering Terminilogy*, IEEE, 1990.

[10] Stephen H. Kan, *Metrics and Models in Software Quality Engineering*, Addison-Wesley, 2002.

[11] Per Kroll and Philippe Kruchten, *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*, Addison Wesley, 2003.

[12] Philippe Kruchten, *The Rational Unified Process: An Introduction (2^{nd}, 3^{rd} editions)*, Addison Wesley, 2000, 2003.

[13] Craig Larman, *Agile and Iterative Development: A Manager's Guide*, Addison Wesley, 2003.

[14] Timothy C. Lethbridge and Robert Laganière, *Object-Oriented Software Engineering: Practical Software Development Using UML and Java*, McGraw-Hill, 2002.

[15] Raymond J. Madachy, *Software Process Dynamics*, IEEE Press – Wiley, 2008.

[16] Mario E. Moreira, *Software Configuration Management Implementation Roadmap*, Wiley, 2004.

[17] Rational Software White Paper, *Reaching CMM Levels 2 and 3 with the Rational Unified Process*, Rational Software Corporation, 2000.

[18] John W. Rittinghouse, *Managing Software Deliverables: A Software Development Management Methodology*, Digital Press – Elsevier, 2004.

[19] Robert E. Park, *Software Size Measurement: A Framework for Counting Source Statements*, Technical Report CMU/SEI-92-TR-020 ESC-TR-92-020, 1996.

[20] Roger S. Pressman, *Software Engineering: A Practitioner's Approach (6^{th} edition)*, McGraw-Hill, 2005.

[21] Stephen R. Schach, *Object-Oriented and Classical Software Engineering (5^{th},6^{th},7^{th} editions)*, McGraw-Hill, 2002, 2005, 2007.

[22] Stephen R. Schach, *Object-Oriented and Classical Software Engineering (5^{th} editions)*, McGraw-Hill, 2002.

[23] Ian Sommerville, *Software Engineering (6^{th},8^{th} editions)*, Addison-Wesley, 2001, 2006.

[24] Jeff Tian, *Software Quality Engineering: Testing Quality Assurance and Quantifiable Improvement*, IEEE Computer Society - Wiley, 2005.

[25] Hans van Vliet, *Software Engineering: Principals and Practice (2^{nd} edition)*, Wiley, 2000.

[26] http://www.rspa.com/

[27] http://www.sei.cmu.edu/

[28] http://computingcareers.acm.org/

CÂU HỎI HƯỚNG DẪN ÔN TẬP

1. Sự khác nhau giữa trừu tượng hóa thủ tục và trừu tượng hóa dữ liệu?
2. Giải thích các khái niệm gắn kết và nối kết?
3. Mô tả các dạng phần mềm thích hợp với chu trình sống mã nguồn mở, phát triển nhanh (agile), xoắn ốc và thác nước?
4. Phân biệt giai đoạn phân tích/đặc tả và thiết kế? Có cần thiết gộp hai giai đoạn này thành một giai đoạn duy nhất hay không?
5. Phân biệt các dạng tổ chức nhóm làm việc?
6. Tầm quan trọng của việc cải tiến tiến trình phần mềm?
7. Phân biệt các cách thức tổ chức kiểm thử?
8. Khái niệm lỗi trong công nghệ phần mềm được hiểu như thế nào?
9. Phân biệt sự khác nhau giữa sử dụng lại và dễ di chuyển?

ĐỊNH HƯỚNG THẢO LUẬN

1. Xác định những tình huống phát triển phần mềm phù hợp với chu trình sống của phần mềm?
2. Mô hình chu trình sống của phần mềm nào (theo bạn) là hiệu quả nhất?
3. Những sản phẩm phần mềm nào được phát triển dựa trên mô hình bản mẫu?
4. Các công việc chính và vai trò của quản lý cấu hình?
5. Những vấn đề chính của mô hình tổ chức nhóm dạng phân cấp, dạng ma trận?
6. Hoạt động của nhóm đảm bảo chất lượng phần mềm như thế nào (theo bạn) sẽ là hiệu quả nhất?
7. Thảo luận về luật Brooks?
8. Vấn đề mấu chốt của ẩn thông tin là gì?
9. Mô tả tình huống trong đó khách hàng, nhà phát triển và người sử dụng là một?
10. Sự khác nhau giữa nhóm làm việc dạng dân chủ và mã nguồn mở?
11. Thảo luận sự giống và khác nhau trong việc ẩn thông tin giữa C++ và Java?
12. Ảnh hưởng của gắn kết và nối kết lên việc sử dụng lại?
13. Kỹ thuật Cleanroom?

BÀI TẬP THỰC HÀNH

1. Tìm kiếm và viết một tổng kết về các công cụ phần mềm hỗ trợ mô hình chu trình sống của phần mềm RAD.
2. Tím kiếm và thống kê các dạng công cụ phần mềm.
3. Tìm kiếm và xác định mức CMM hiện có của các công ty phần mềm tại Việt Nam.
4. Cài đặt công cụ tính toán độ nối kết và độ gắn kết.
5. Một phần mềm có chi phí phát triển khoảng 540 triệu, vậy chi phí cần bổ sung cho giai đoạn phân phối/bảo trì cần là bao nhiêu?

CHƯƠNG 2

TIẾN TRÌNH PHẦN MỀM

Tiến trình phần mềm thường trải qua các giai đoạn phân tích yêu cầu, phân tích hệ thống (hay còn được gọi là đặc tả), thiết kế, cài đặt, tích hợp và bảo trì.

1. PHÂN TÍCH YÊU CẦU

Đây là giai đoạn xác định cái mà khách hàng cần (needs) chứ không phải cái mà khách hàng muốn (wants) đồng thời phải phân tích càng chính xác càng tốt thực trạng hiện nay của khách hàng. Một số khó khăn chính thường gặp phải như khách hàng không biết họ cần gì và ngay cả khi khách hàng biết rõ mình cần gì thì cũng sẽ khó khăn khi chuyển tải những thông tin này cho nhà phát triển theo hướng tin học hóa!

Việc khởi động giai đoạn phân tích yêu cầu bắt đầu khi các thành viên của nhóm phân tích yêu cầu (RAT) tiếp xúc với khách hàng. Thông thường thì khách hàng sẽ sắp xếp những buổi phỏng vấn đầu tiên và các buổi phỏng vấn thêm sẽ được xếp lịch trong tiến trình phỏng vấn. Tiến trình phỏng vấn kết thúc khi nhóm RAT nhận thấy đã nắm bắt được các thông tin liên quan từ khách hàng và những người sử dụng tương lai của sản phẩm.

1.1 Phỏng vấn theo cấu trúc

Phỏng vấn theo cấu trúc (structured interview) cần chuẩn bị sẵn các câu hỏi cụ thể dạng đóng để nêu ra. Chẳng hạn như khách hàng có thể được hỏi "có bao nhiêu nhân viên bán hàng trong công ty?" "khoảng thời gian giới hạn cho một đáp ứng yêu cầu là bao nhiêu?". Sau đó người đi phỏng vấn viết một báo cáo cho biết các nội dung chính của buổi phỏng vấn và gửi một bản sao cho khách hàng để hiệu chỉnh.

1.2 Phỏng vấn không theo cấu trúc

Phỏng vấn không theo cấu trúc (unstructured interview) cần đặt các câu hỏi dạng mở nhằm khuyến khích khách hàng nói rõ các thông tin. Chẳng hạn như bằng cách hỏi khách hàng "tại sao không vừa ý với sản phẩm hiện hiện nay?" có thể cho biết được nhiều khía cạnh trong cách thức kinh doanh của khách hàng. Với người phỏng vấn nhiều kinh nghiệm có thể đặt các câu hỏi mở rộng sau khi đã lắng nghe cẩn thận và dẫn dắt cuộc nói chuyện đi xa hơn, do đó sẽ có nhiều thông tin tốt. Sau đó người đi phỏng vấn viết một báo cáo cho biết các nội dung chính của buổi phỏng vấn và gửi một bản sao cho khách hàng để hiệu chỉnh.

1.3 Gửi bản câu hỏi

Gửi bản câu hỏi (send a questionnaire) là cách thực hiện nhằm gửi một bản câu hỏi đến các thành viên liên quan trong cơ quan khách hàng. Cách làm này rất hữu dụng vì tập hợp được ý kiến của nhiều cá nhân khác nhau, các ý kiến phản hồi sẽ được suy nghĩ cẩn thận và xác đáng hơn. Khuyết điểm chính là khó mở rộng các câu hỏi và ít thông tin mở rộng.

1.4 Khảo sát các biểu bảng

Khảo sát các biểu bảng (examine the various forms) thường được sử dụng trong môi trường kinh doanh và tiến hành khảo sát toàn bộ các biểu bảng được khách hàng sử dụng. Chẳng hạn như việc khảo sát một mẫu biểu được in trong một cửa hàng có thể phản ánh được các thông tin về số trang in, kích thước khổ giấy, độ ẩm, nhiệt độ mực in, áp lực trên giấy, các trường khác nhau trang mẫu biểu sẽ chỉ rõ sự chuyển tiếp giữa các tác vụ in và các giai đoạn tương đối quan trọng. Việc thấu hiểu các thông tin qua việc quan sát cách thức kinh doanh của khách hàng là cách cực kỳ hữu ích nhằm xác định những cái mà khách hàng cần.

1.5 Quay phim

Quay phim (set up video/cameras) là phương pháp mới được sử dụng gần đây và được tiến hành tại nơi làm việc nhằm ghi lại chính xác mọi diễn biến. Khi thực hiện kỹ thuật này nhóm RAT phải có được sự hợp tác của tất cả các thành viên. Cần chú ý là sẽ cực kỳ khó khăn để nắm bắt các thông tin cần thiết khi người được quay phim cảm thấy mình bị xâm phạm đời tư, lo sợ hay không thích bị quấy rầy. Cần dự kiến trước các rủi ro trước khi giới thiệu máy quay phim vì có thể việc này sẽ gây ra những tức giận cho thành viên được quay phim.

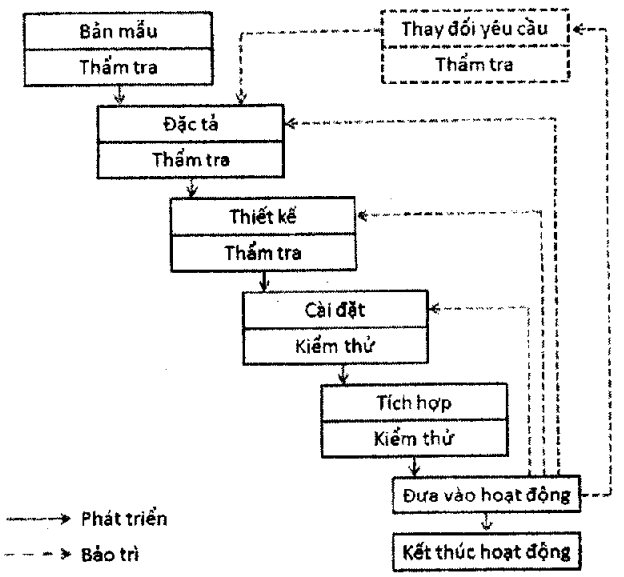
1.6 Sử dụng kịch bản

Sử dụng các kịch bản (scenarios) là cách thức mà người sử dụng có thể tiến hành trên sản phẩm nhằm hoàn thành một số mục tiêu nào đó. Chẳng hạn như một kế hoạch làm giảm cân sẽ có kịch bản là chuyên gia dinh dưỡng nhập tuổi, giới tính, khối lượng và các dữ liệu cá nhân khác của một bệnh nhân béo phì; sản phẩm in ra thực đơn cho bệnh nhân; kịch bản được đưa ra cho người sử dụng tương lai của sản phẩm; chuyên gia dinh dưỡng chỉ ra các điểm không phù hợp cho một bệnh nhân phải sử dụng các thức ăn đặc biệt được chỉ định như người bị bệnh tiểu đường, người ăn chay hay người bị bệnh đường huyết; nhà phát triển cập nhật lại kịch bản trong đó người sử dụng được hỏi về chế độ ăn uống đặc biệt cần có trước khi thực đơn được in. Trong kỹ thuật này cần cho phép người sử dụng tương tác với chính bản thân họ và nhóm RAT sẽ ghi nhận lại các thông tin này. Một số cách mô tả kịch bản như liệt kê các hành động có trong kịch bản hay tạo một bản tinh tiết lưu trữ chuỗi các sự kiện (chẳng hạn như một mẫu giấy trong đó có một chuỗi các biểu bảng, mỗi biểu bảng liên quan đến một màn hình

và trả lời của người sử dụng). Ưu điểm của kỹ thuật này là thể hiện cách đối xử của sản phẩm mà người sử dụng có thể hiểu và cảm nhận được và người sử dụng hiểu được kịch bản do đó sẽ đóng vai trò tích cực trong quá trình phân tích yêu cầu. Đồng thời nguồn thông tin về cái cần thực sự (real needs) của khách hàng sẽ do chính khách hàng và người sử dụng cung cấp. Áp dụng thành công kỹ thuật này sẽ đóng vai trò quan trọng trong việc phân tích hướng đối tượng (chính là các trường hợp sử dụng - use cases).

1.7 Dựa trên bản mẫu

Dựa trên bản mẫu (rapid prototyping) là cách làm nhằm mục tiêu xây dựng mô hình càng nhanh càng tốt. Chẳng hạn như nó được xây dựng dành cho các thay đổi phản ánh chức năng mà khách hàng thấy như các màn hình nhập, các báo cáo,...bỏ qua các khía cạnh như: cập nhật tập tin,... Khách hàng, người sử dụng tương lai và nhóm phát triển sẽ cùng nhau xem xét và ghi nhận các dữ kiện. Các nhà phát triển sẽ liên tục thay đổi mô hình cho đến khi mô hình đã chứa đựng mọi cái cần có. Quá trình định hình nhanh sẽ được sử dụng cho giai đoạn đặc tả và rất hiệu quả khi phát triển giao diện người dùng, hướng đối tượng.



***Hình 32:** Sử dụng bản mẫu cho phân tích yêu cầu.*

Kiểm thử trong giai đoạn phân tích yêu cầu do nhóm SQA tiến hành nhằm đảm bảo các cá nhân trong cơ quan khách hàng có tương tác với mô hình bản mẫu và phân tích các đề xuất. Có thể thành lập một hội đồng của khách hàng có trách nhiệm phân tích các đề xuất của khách hàng.

Đánh giá giai đoạn phân tích yêu cầu sẽ dựa trên tần suất thay đổi các yêu cầu (phỏng vấn, kịch bản), số lượng các yêu cầu thay đổi trong các giai đoạn còn lại (áp dụng cho toàn bộ các kỹ thuật) đồng thời sẽ phân tích lại khi có quá nhiều thay đổi và số lần mỗi đặc điểm được xây dựng.

2. ĐẶC TẢ/PHÂN TÍCH HỆ THỐNG

Tài liệu phân tích hệ thống/đặc tả phải đáp ứng được hai yêu cầu mâu thuẫn nhau là rõ ràng và dễ hiểu đối với khách hàng (dễ thuyết phục) đồng thời phải đầy đủ và chi tiết vì đây là nguồn thông tin duy nhất dành cho nhóm thiết kế. Các lỗi xảy ra trong giai đoạn này sẽ ảnh hưởng đến các giai đoạn còn lại của toàn bộ tiến trình, đây có thể xem như là hợp đồng (contract) giữa khách hàng và nhà phát triển. Tài liệu phải bao gồm các ràng buộc mà sản phẩm phải đáp ứng như thời hạn phân phối sản phẩm cho khách hàng, sản phẩm được cài đặt để chạy thử song song với sản phẩm hiện hành cho đến khi khách hàng chấp nhận, dễ dàng chuyển đổi trên các phần cứng hay hệ điều hành khác nhau, có độ tin cậy cao, hoạt động tốt 24/24 giờ (nếu có yêu cầu) và thời gian đáp ứng nhanh (chẳng hạn như 95% các truy vấn dạng 4 phải được trả lời trong khoảng 0.25s). Thành phần sống còn của tài liệu này là tập các tiêu chuẩn chấp thuận và cách tiếp cận chung để tạo ra sản phẩm (còn được xem như giải pháp chiến lược).

2.1 Đặc tả không hình thức

Đặc tả không hình thức được thực hiện bằng cách sử dụng ngôn ngữ tự nhiên. Cách thực hiện này thường có nhiều lỗi xảy ra và rõ ràng là ngôn ngữ tự nhiên không phải là phương cách tốt để đặc tả sản phẩm. Một phát biểu sau đây có thể xem như là một đặc tả không hình thức "Nếu doanh thu của tháng hiện tại thấp hơn với doanh thu dự kiến thì một báo cáo sẽ được in ra, trừ phi hiệu số doanh thu giữa doanh thu và doanh thu dự kiến của tháng hiện tại nhỏ hơn phân nửa hiệu số doanh thu tương tự như trên của tháng trước đó hoặc hiệu số doanh thu hiện tại này nhỏ hơn 5%".

2.2 Đặc tả bán hình thức

Đặc tả bán hình thức thường sử dụng các cấu trúc đồ họa và theo hướng tập trung về dữ liệu hoặc về xử lý. Phân tích theo cấu trúc sử dụng đồ họa và được ứng dụng rộng rãi như các kỹ thuật của Gane và Sarsen, Yourdon và Constantine, DeMarco, PSL/PSA, SADT, SREM dựa trên kỹ thuật máy hữu hạn trạng thái (bao gồm các thành phần RSL: ngôn ngữ đặc tả, REVS: tập các công cụ thực hiện các mối liên hệ trong việc đặc tả, DCDS: kỹ thuật thiết kế), mô hình thực thể - quan hệ, mô hình thực thể - kết hợp và phân tích hệ thống hướng đối tượng.

2.3 Đặc tả hình thức

Đặc tả hình thức thường gắn với các ký hiệu tương đối chặt chẽ như các kỹ thuật Ana đặc tả cho ngôn ngữ Ada, Gist dùng để mô tả các tiến trình, VDM cho ngữ nghĩa, CSP biểu diễn các sự kiện và các tiến trình với môi trường làm việc, mạng Petri và Z.

Đặc tả thường được đánh giá thông qua 5 đại lượng cơ bản như kích thước (số trang tài liệu đặc tả, kích thước sản phẩm đích,...), giá thành, thời gian, nhân lực, chất lượng (thống kê lỗi).

3. THIẾT KẾ

Hàng trăm kỹ thuật thiết kế đã ra đời trong hơn 30 năm qua, tựu trung lại có 3 nhóm chính là thiết kế hướng xử lý (action-oriented) nhằm phân rã sản phẩm thành các mô đun có tính chặt chẽ cao và ít gắn kết với nhau, thiết kế hướng dữ liệu (data-oriented) phụ thuộc vào cấu trúc dữ liệu mà các xử lý được thực hiện trên đó và thiết kế hướng đối tượng dưới dạng tổng hợp, bao gồm cả xử lý và dữ liệu. Đầu vào của thiết kế là tài liệu đặc tả, cho biết sản phẩm phải làm gì (what ?); còn đầu ra phải xác định được câu hỏi là để đạt được những công việc đã mô tả ở đầu vào, sản phẩm phải thực hiện như thế nào (how?).

Giai đoạn thiết kế phần mềm có 3 hoạt động chính: thiết kế kiến trúc, thiết kế chi tiết và kiểm thử. Thiết kế kiến trúc (architectural design, general design, logical design, high-level design) theo quan điểm trừu tượng hóa là phân chia sản phẩm thành các mô đun. Thiết kế chi tiết (detailed design, modular design, physical design, low-level design) nhằm chi tiết hóa từng mô đun, chọn giải thuật, chọn cấu trúc dữ liệu. Kiểm thử thiết kế (design testing) nhằm rà soát lại sự chính xác của tài liệu thiết kế.

Việc kiểm thử trong giai đoạn thiết kế nhằm tìm thấy lỗi trong giai đoạn này là rất quan trọng. Có thể sử dụng phương pháp walkthroughs hoặc thanh tra tương tự như trong giai đoạn đặc tả nhưng có thể không có đại diện của khách hàng. Việc kiểm thử phải phản ánh được hướng thiết kế.

Có nhiều phương pháp đánh giá trên các mặt của giai đoạn thiết kế như số lượng các mô đun (đánh giá thô về kích thước của sản phẩm), độ gắn kết của mô đun (đánh giá về chất lượng), độ nối kết giữa các mô đun (thống kê về lỗi).

4. CÀI ĐẶT

Cài đặt là quá trình chuyển đổi từ thiết kế chi tiết sang mã lệnh do nhiều người thực hiện (programming-in-the-many) hay còn gọi là lập trình. Quá trình này phụ thuộc vào việc lựa chọn ngôn ngữ lập trình (C, C++, Java, PHP,...), phụ thuộc

vào hợp ngữ của máy tính, phụ thuộc vào số lượng ngôn ngữ lập trình sẵn có và thói quen sử dụng ngôn ngữ lập trình.

Kỹ năng lập trình tốt khi hiểu rõ ngôn ngữ (language-specific), sử dụng tên biến thích hợp và có nghĩa (use of consistent and meaningful variable names), có nghĩa theo quan điểm của các nhà lập trình bảo trì, chú ý đến ngôn ngữ mẹ đẻ của các lập trình viên, thống nhất ngôn ngữ để đặt tên biến (tiếng Anh,...), tên biến phải rõ ràng và không gây lầm lẫn, dễ dàng hiểu các mã lệnh. Chú thích tự thân (the issue of self-documenting code) được hiểu là không có các dòng chú thích và các tên biến phải được diễn giải ngay từ đầu (prologue comments). Thông tin tối thiểu của một mô đun (the minimum information) cần có là tên mô đun, mô tả vắn tắt các công việc mô đun phải thực hiện, tên của lập trình viên, ngày viết mô đun, ngày mô đun được chấp thuận và được chấp thuận bởi ai, các tham số, danh sách các tên biến (nên theo thứ tự chữ cái) và cách sử dụng, tên các tập tin mà mô đun có truy xuất, tên các tập tin bị thay đổi bởi mô đun (nếu có), nhập/xuất của mô đun (nếu có), các khả năng lỗi xảy ra, tên tập tin sẽ được sử dụng để kiểm thử, danh sách các cập nhật đã được tiến hành với ngày tương ứng, người chấp thuận và các lỗi đã biết (nếu có).

Một mô đun phải chịu hai lần kiểm thử: không hình thức (informal testing), do lập trình viên tiến hành; theo phương pháp (methodical testing), do nhóm SQA thực hiện sau khi khi lập trình viên xác nhận rằng mô đun đã vận hành tốt (có thể không dựa trên việc thực thi hoặc dựa trên việc thực thi. Cách kiểm thử dở nhất là sử dụng dữ liệu kiểm thử bừa bãi, khi đó sẽ không có đủ thời gian để thực hiện và cách kiểm thử tốt nhất là xây dựng các trường hợp kiểm thử có hệ thống, các bộ dữ liệu kiểm thử được tạo ra có chọn lọc.

5. TÍCH HỢP

Sẽ không tốt nếu như việc cài đặt được thực hiện riêng rẽ và sau đó tích hợp lại toàn bộ và được kiểm thử chung. Sẽ tốt hơn nếu như việc cài đặt và tích hợp được thực hiện song song. Các dạng tích hợp thông dụng bao gồm tích hợp từ trên xuống, tích hợp từ dưới lên và tích hợp hỗn hợp. Vấn đề kiểm thử tích hợp giao diện sẽ khó khăn khi kiểm thử người dùng sử dụng chuột, kích hoạt thực đơn,... Kiểm thử sản phẩm, do nhóm SQA thực hiện theo trình tự: hộp đen, từng mô đun (hoặc từng đối tượng), dạng thô trên toàn sản phẩm, đặc biệt (khi đăng nhập,...), khối lượng (với nhiều tập tin đầu vào), xem xét lại toàn bộ các tài liệu sẽ trao cho khách hàng, đối chiếu giữa tài liệu và sản phẩm. Việc chấp nhận kiểm thử được thực hiện giữa khách hàng và nhóm SQA cần phải thực hiện trên dữ liệu thực tế chứ không đơn thuần là dữ liệu dùng để kiểm thử. Trong trường hợp nếu sản phẩm mới thay thế sản phẩm đã có sẵn, trong tài liệu đặc tả phải có một điều khoản cho phép sản phẩm mới được cài đặt song song với sản phẩm cũ cho

đến khi khách hàng chấp thuận. Kết thúc giai đoạn kiểm thử thì công việc của các nhà phát triển xem như kết thúc và sản phẩm chuyển sang giai đoạn bảo trì.

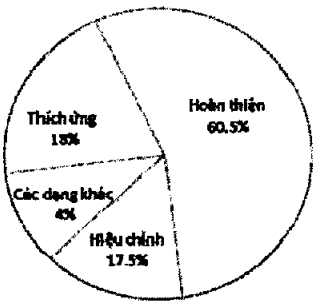
Tích hợp môi trường nhằm mục tiêu tất cả các công cụ trong cùng một môi trường có cùng giao diện người dùng. Tích hợp các tiến trình tạo môi trường thường hỗ trợ cho một tiến trình phát triển phần mềm đặc biệt nào đó, còn gọi là môi trường kỹ thuật nền. Tích hợp công cụ nhằm mục tiêu tất cả các công cụ giao tiếp với nhau thông qua các định dạng dữ liệu giống nhau (chẳng hạn theo dạng mã ASCII). Tùy tình huống mà các dạng tích hợp khác được triển khai một cách phù hợp.

6. BẢO TRÌ

Giai đoạn bảo trì bắt đầu sau khi khách hàng đã chấp thuận, đưa vào sử dụng sản phẩm và cần có các thay đổi trên sản phẩm. Các thể hiện của bảo trì là mã nguồn, tài liệu, hướng dẫn sử dụng,... Giai đoạn này còn được xem là sự tiến triển để chỉ rõ sự phát triển của sản phẩm thay vì gọi đó là bảo trì. Được xem như là dịch vụ hậu mãi (after-sales service), giữ khách hàng bằng cách cung cấp những dịch vụ bảo trì tốt nhất và là chuẩn mực cho sự thành công của công ty phần mềm. Đây là giai đoạn được xem như khó khăn nhất, nhiều thách thức của một sản phẩm phần mềm vì đụng chạm đến tất cả các giai đoạn trong tiến trình xây dựng phần mềm. Nghịch lý hiện nay tại các công ty là xem nhẹ công tác bảo trì và thường giao các công đoạn bảo trì cho các lập trình viên mới.

Lập trình viên bảo trì (MP) phải có kỹ năng lần vết (debugging skills) tốt để xác định chính xác vị trí lỗi. Cần quan tâm đến lỗi hồi qui (regression fault) do sửa chữa lỗi hiện tại phải có quan tâm đến các lỗi khác trong sản phẩm. Cần chuẩn bị tài liệu chi tiết cho toàn bộ sản phẩm cũng như cho từng mô đun riêng biệt sau khi sửa chữa xong.

Các dạng bảo trì gồm có hiệu chỉnh (khoảng 17.5% thời gian) liên quan đến các lỗi đặc tả, thiết kế, tài liệu, mã nguồn hay các dạng khác; hoàn thiện (khoảng 60.5%) liên quan các thay đổi về mã lệnh nhằm hoàn thiện hiệu năng của sản phẩm; thích ứng (khoảng 18%) liên quan đến các thay đổi nhằm tác động lại những thay đổi trong môi trường mà sản phẩm đang vận hành và khách hàng phải chịu chi phí; các dạng khác (khoảng 4%) thuộc các dạng khác ngoài ba dạng kể trên.



Hình 33: Các khoảng thời gian cho mỗi dạng bảo trì.

Việc quản lý bảo trì nhằm xây dựng cơ chế cho phép có những thay đổi trên sản phẩm khi bảo trì với lãnh đạo nhóm SQA và lãnh đạo nhóm phát triển phần mềm phải độc lập với nhau. Một số nội dung bảo trì như hình thành các báo cáo lỗi (người sử dụng điền các thông tin về lỗi trên các chức năng, đủ thông tin để MP có thể tái tạo lại lỗi), ủy quyền thay đổi trên sản phẩm (xác định lỗi, thay đổi mã nguồn, cố định mã nguồn; kiểm thử qui hồi trên toàn bộ sản phẩm; cập nhật các tài liệu để phản ánh các thay đổi; có thể cập nhật tài liệu về đặc tả cũng như thiết kế; tạo phiên bản mới; chuyển đến nhóm SQA để xác nhận lại - nhưng không được can thiệp vào công việc của các lập trình viên), bảo đảm công tác bảo trì (việc bảo trì phải được thực hiện nhiều lần, tạo nhiều phiên bản, có kế hoạch bảo trì trong suốt tiến trình phần mềm, ghi nhận cẩn thận các thông tin kỹ thuật, tài liệu phải được hoàn tất và hiệu chỉnh chu đáo, phản ánh chính xác mọi thành phần của phiên bản hiện hành), vấn đề về sự lặp lại công tác bảo trì (khách hàng thường xuyên thay đổi các yêu cầu, nên đưa ra mô hình làm việc, khi có thay đổi thì khách hàng sẽ phải trả thêm chi phí phát triển).

Việc bảo trì các phần mềm hướng đối tượng thường diễn ra thuận lợi do các khái niệm độc lập nên dễ dàng xác định vị trí nhằm hiệu chỉnh hay nâng cao và các thay đổi chỉ tác dụng bên trong đối tượng nên giảm thiểu các lỗi hồi qui. Một số khó khăn gặp phải là MP phải nghiên cứu toàn bộ các cây thừa kế, vấn đề đa hình và động khi cài đặt trên ngôn ngữ lập trình hướng đối tượng và khi lần vết các thừa kế liên tục nhau khi có một lớp nào đó có một số thay đổi.

Kiểm thử giai đoạn bảo trì khá khó khăn khi phải nắm vững toàn bộ sản phẩm, do đó cách tiến hành thường là sử dụng các tình huống kiểm thử (test cases) để đảm bảo sản phẩm vẫn còn vận hành tốt sau khi đã có cập nhật, thay đổi một số tình huống kiểm thử, lưu trữ toàn bộ các tình huống kiểm thử với kết quả cần đạt tương ứng và sử dụng kiểm thử hồi qui.

Đánh giá giai đoạn bảo trì giống như cách đánh giá cho các giai đoạn liên quan như trong quá trình phát triển. Ngoài ra còn có thể xác định thêm số lượng báo cáo lỗi, phân loại lỗi theo độ khó và kiểu lỗi và thông tin về trạng thái hiện hành

của báo cáo lỗi (chẳng hạn như 13 báo cáo lỗi nghiêm trọng đã được xử lý, 4 báo cáo lỗi nghiêm trọng chưa được xử lý).

TÀI LIỆU THAM KHẢO

[1] Len Bass *et al.*, *Software Architecture in Pratice*, Addison-Wesley, 2003.

[2] Barry W. Boehm, *Software Engineering*, IEEE Computer Society - Wiley, 2007.

[3] Alphonse Carlier, *Le développement du logiciel*, Hermes, 1995.

[4] Alistair Cockburn, *Writing Effective Use-Cases*, Addison-Wesley, 2000.

[5] Scott E. Donaldson and Stanley G. Siegel, *Successful Software Development (2nd edition)*, Prentice Hall, 2000.

[6] IEEE Std 828, *IEEE Standard for Software Configuration Management Plans*, The Institute of Electrical and Electronics Engineers, Inc., 1998.

[7] IEEE, *Guide to the Software Engineering Body of Knowledge - SWEBOK®*, IEEE Computer Society, 2004.

[8] The International Function Users Group, *Function Point Counting Practices Manual (Release 4.1)*, 1999.

[9] Stephen H. Kan, *Metrics and Models in Software Quality Engineering*, Addison-Wesley, 2002.

[10] Per Kroll and Philippe Kruchten, *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*, Addison Wesley, 2003.

[11] Philippe Kruchten, *The Rational Unified Process: An Introduction (2nd, 3rd editions)*, Addison Wesley, 2000, 2003.

[12] Craig Larman, *Agile and Iterative Development: A Manager's Guide*, Addison Wesley, 2003.

[13] Timothy C. Lethbridge and Robert Laganière, *Object-Oriented Software Engineering: Practical Software Development Using UML and Java*, McGraw-Hill, 2002.

[14] Raymond J. Madachy, *Software Process Dynamics*, IEEE Press – Wiley, 2008.

[15] Nguyễn Minh Hồng, "Hướng dẫn xác định trị giá phần mềm", *Công văn số 3364/BTTTT-UDCNTT ngày 17 tháng 10 năm 2008*, Bộ Thông tin & Truyền thông, 2008.

[16] Nguyễn Minh Hồng, "Sửa đổi giá trị trọng số BMT", *Công văn số 2496/BTTTT-UDCNTT ngày 04 tháng 08 năm 2010*, Bộ Thông tin & Truyền thông, 2010.

[17] Paulraj Ponniah, *Data Modeling Fundamentals*, Wiley, 2007.

[18] Rational Software White Paper, *Rational Unified Process® for Systems Engineering RUP® SE1.1*, Rational Software Corporation, TP 165A, 2002.

[19] Rational Software White Paper, *Rational Unified Process: Best Practices for Software Development Teams*, IBM Corporation, TP026B, Rev 11/01, 2001.

[20] Rational Software White Paper, *Reaching CMM Levels 2 and 3 with the Rational Unified Process*, Rational Software Corporation, 2000.

[21] John W. Rittinghouse, *Managing Software Deliverables: A Software Development Management Methodology*, Digital Press – Elsevier, 2004.

[22] Robert E. Park, *Software Size Measurement: A Framework for Counting Source Statements*, Technical Report CMU/SEI-92-TR-020 ESC-TR-92-020, 1996.

[23] Roger S. Pressman, *Software Engineering: A Practitioner's Approach (6^{th} edition)*, McGraw-Hill, 2005.

[24] Stephen R. Schach, *Object-Oriented and Classical Software Engineering (5^{th},6^{th},7^{th} editions)*, McGraw-Hill, 2002, 2005, 2007.

[25] Ian Sommerville, *Software Engineering (6^{th},8^{th} editions)*, Addison-Wesley, 2001, 2006.

[26] Jeff Tian, *Software Quality Engineering: Testing Quality Assurance and Quantifiable Improvement*, IEEE Computer Society - Wiley, 2005.

[27] Bhuvan Unhelkar, *Pratical Object Oriented Design*, Thompson Social Science Press, 2005.

[28] Hans van Vliet, *Software Engineering: Principals and Practice (2^{nd} edition)*, Wiley, 2000.

[29] http://www.rspa.com/

[30] http://www.sei.cmu.edu/

CÂU HỎI HƯỚNG DẪN ÔN TẬP

1. Mục tiêu chính của thiết kế kiến trúc?
2. Phân biệt phân tích yêu cầu và phân tích/đặc tả hệ thống?
3. Vai trò của từng giai đoạn trong chu trình sống của phần mềm?

ĐỊNH HƯỚNG THẢO LUẬN

1. Chất lượng thiết kế sẽ như thế nào nếu có các mẫu thiết kế?
2. Phân biệt trường hợp sử dụng và sơ đồ trường hợp sử dụng, trường hợp sử dụng và tác nhân?
3. Ngôn ngữ lập trình có đóng vai trò quan trọng trong giai đoạn thiết kế không?
4. Các dạng sơ đồ trong UML (Unified Modeling Language)?
5. Reverse engineering là gì?

6. Các thông tin tối thiểu cần có để mô tả một chương trình con/hàm?
7. Báo cáo lỗi của từng giai đoạn trong chu trình sống của phần mềm sẽ có cấu trúc như thế nào?
8. Với vai trò là người quản lý giai đoạn phân phối/bảo trì, năng lực cần có của người tham gia giai đoạn này sẽ là như thế nào?

BÀI TẬP THỰC HÀNH

1. Tìm kiếm, cài đặt và sử dụng các công cụ phần mềm hỗ trợ cho từng giai đoạn trong chu trình sống của phần mềm (IBM, Sysbase, Microsoft Visual Studio, Qt, NetBeans,...)? Đưa ra các phân tích, nhận xét?
2. Tìm kiếm và liệt kê các ngôn ngữ lập trình đang được sử dụng hiện nay?
3. Đề xuất thực hiện các nội dung chi tiết theo quy trình công nghệ phần mềm cho một dự án/phần mềm cụ thể.

CHƯƠNG 3

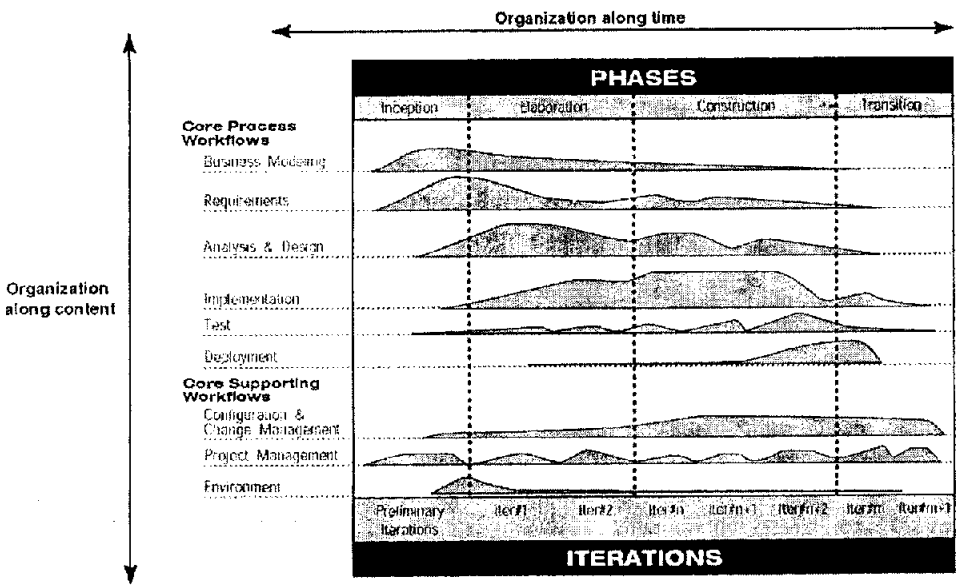
TIẾN TRÌNH RUP

1. RUP LÀ GÌ?

RUP (The Rational Unified Process®) là một quy trình/tiến trình công nghệ phần mềm và cung cấp một tiếp cận chặt chẽ, mang tính kỷ luật cao trong quá trình giao việc và trách nhiệm trong quá trình tổ chức phát triển phần mềm, đảm bảo sản xuất phần mềm với chất lượng cao cùng với sự đòi hỏi của người dùng trong khuôn khổ thời gian và ngân quỹ dự kiến. RUP hỗ trợ rất tốt cho UML (Unified Modeling Language) và là tiến trình có thể điều chỉnh được thông qua các tham số.

RUP hỗ trợ cho các nhóm thực hiện phần mềm hướng dẫn, mẫu và các công cụ tư vấn cần thiết trên cơ sở phát triển phần mềm một cách lặp, quản lý yêu cầu, dựa trên kiến trúc hướng thành phần, mô hình phần mềm trực quan, thẩm tra chất lượng phần mềm và kiểm soát các thay đổi của phần mềm.

Tiến trình RUP có thể được mô tả theo hai hướng tổ chức theo thời gian và theo nội dung. Hướng thời gian (trục hoành) thể hiện khía cạnh động của tiến trình khi vận hành và được trình bày về chu trình, giai đoạn, lặp và thời điểm quan trọng. Hướng nội dung (trục tung) thể hiện khía cạnh tĩnh của tiến trình và được trình bày về hoạt động, tác vụ, vai trò và dòng công việc.



Hình 34: Tiến trình RUP với hai trục thời gian và nội dung.

2. CẤU TRÚC ĐỘNG CỦA RUP

RUP được cấu trúc động thành các chu kỳ và được lặp lại, mỗi chu kỳ làm việc trên một thế hệ mới của sản phẩm phần mềm. RUP chia chu kỳ phát triển thành 4 giai đoạn liên tiếp: khởi đầu (inception phase), triển khai (elaboration phase), xây dựng (construction phase), chuyển giao (transition phase). Mỗi giai đoạn được xác định bởi một mốc thời gian quan trọng, tại thời điểm này một số quyết định quan trọng phải được thực hiện và như vậy các mục tiêu quan trọng phải đạt được. Mỗi giai đoạn có một mục đích cụ thể.

Trong giai đoạn khởi đầu, các tình huống nghiệp vụ của hệ thống được thiết lập và xác định giới hạn của phần mềm. Để hoàn thành giai đoạn này cần phải xác định tất cả các thực thể mà hệ thống sẽ tương tác (các tác nhân) và định nghĩa bản chất của việc tương tác này ở mức khái quát. Công việc này cũng bao gồm việc xác định tất cả các trường hợp sử dụng và mô tả một số trường hợp sử dụng quan trọng. Các trường hợp nghiệp vụ bao gồm tiêu chí thành công, đánh giá rủi ro, ước lượng nguồn lực cần có và một kế hoạch với những cột mốc về thời gian. Kết quả của giai đoạn khởi đầu có thể là một tài liệu định hướng, mô hình các trường hợp sử dụng ban đầu (hoàn thành khoảng 10%-20%), một trường hợp nghiệp vụ, một đánh giá rủi ro, một kế hoạch thực hiện hay một hay nhiều bản mẫu,....

Mục đích của giai đoạn triển khai là nhằm phân tích vấn đề đặt ra, thiết lập một nền tảng kiến trúc mạnh, phát triển kế hoạch thực hiện và loại bỏ các yếu tố mang lại nhiều rủi ro. Do đó cần phải suy xét một cách thật kỹ lưỡng ở giai đoạn này theo "cả chiều rộng và chiều sâu" liên quan đến phạm vi cần thực hiện, các chức năng chính và các chức phụ có tác động đến hiệu quả trong yêu cầu. Có thể nói đây là giai đoạn trọng yếu trong toàn bộ 4 giai đoạn cần thực hiện. Kết quả của giai đoạn này sẽ là một mô hình trường hợp sử dụng (hoàn thành khoảng 80%), các yêu cầu bổ sung nắm bắt được các đòi hỏi không chức năng, mô tả kiến trúc phần mềm, kế hoạch phát triển tổng thể của phần mềm, bản hướng dẫn sử dụng ban đầu,...

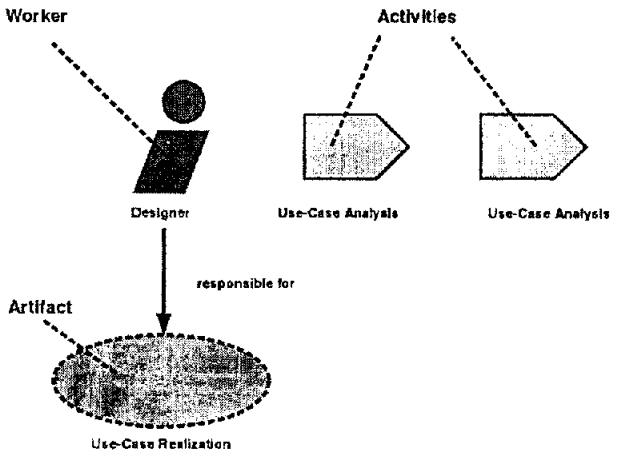
Trong giai đoạn xây dựng toàn bộ các thành phần và đặc điểm ứng dụng được phát triển và tích hợp vào sản phẩm đồng thời toàn bộ các đặc điểm được kiểm tra kỹ lưỡng. Đây là giai đoạn sản xuất vì tập trung vào quản lý nguồn lực và kiểm soát các thao tác để tối ưu hóa chi phí, lịch biểu và chất lượng. Điều đó có nghĩa là giai đoạn này phải thực hiện được tư duy quản lý việc chuyển đổi từ các nội dung suy nghĩ ban đầu ở các giai đoạn khởi đầu và triển khai sang phát triển sản phẩm khả thi trong các giai đoạn xây dựng và chuyển giao. Kết quả của giai đoạn này là sản phẩm đã sẵn sàng chuyển giao cho người sử dụng, bao gồm sản phẩm phần mềm tích hợp với nền tảng phù hợp, hướng dẫn sử dụng và mô tả cho bản phát hành hiện hành.

Mục đích của giai đoạn chuyển giao là nhằm mục tiêu đưa sản phẩm phần mềm đến tay cộng đồng người sử dụng. Khi sản phẩm đã đến tay người sử dụng thì yêu cầu đặt ra sẽ là việc phát triển các bản phát hành mới, chỉnh sửa các lỗi hoặc hoàn tất các đặc điểm đã bị trì hoãn.

Mỗi một giai đoạn trong RUP có thể được chia thêm thành các giai đoạn lặp (iteration) nhỏ hơn. Mỗi một nội dung lặp là một quá trình phát triển được lặp lại và cho kết quả là một phát hành mới của một sản phẩm thực thi được, một tập con của sản phẩm cuối cùng đang trong quá trình phát triển theo dạng tăng trưởng lặp.

3. CẤU TRÚC TĨNH CỦA RUP

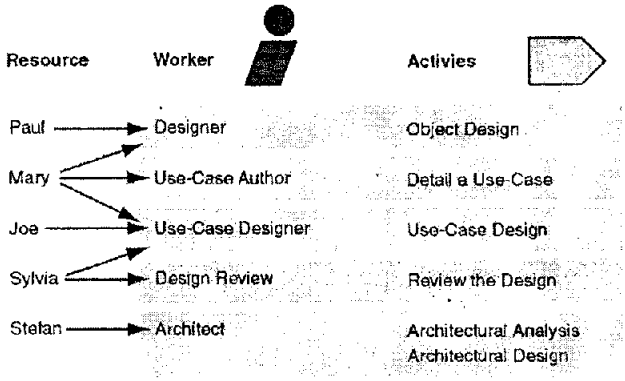
Một tiến trình thường mô tả ai làm cái gì, làm như thế nào và khi nào sẽ làm. RUP là một tiến trình xác định ai thực hiện thông qua vai trò (worker), cách làm như thế nào thông qua các hoạt động (activities), các công việc thông qua các tác vụ (artifacts) và thời gian thực hiện thông qua các dòng công việc (workflows). Ngoài ra việc tổng hợp bốn hoạt động trên được xác định thông qua khung hoạt động (disciplines).



Hình 35: Vai trò, hoạt động và tác vụ trong RUP.

Vai trò (worker/role) xác định hành vi và trách nhiệm của một cá nhân hay một nhóm người làm việc chung trong một nhóm. Các hành vi được thể hiện thông qua các hoạt động mà vai trò cần thực hiện và mỗi vai trò được gắn với một tập hợp các hoạt động. Trách nhiệm của mỗi vai trò được thể hiện thông qua các các công việc mà vai trò tạo ra, cập nhật và điều khiển. Chẳng hạn như phân tích hệ thống (system analyst) là một vai trò mà một cá nhân làm việc trên cơ sở các yêu cầu của hệ thống đã có và mô hình hóa các trường hợp sử dụng bằng các phác thảo các chức năng và giới hạn của hệ thống. Các vai trò được tổ chức thành năm nhóm phân tích, phát triển, kiểm thử, quản lý, sản xuất và hỗ trợ.

Vai trò có các hoạt động định nghĩa công việc cần thực hiện. Một hoạt động là một đơn vị công việc mà một cá nhân trong vai trò đó được yêu cầu thực hiện và tạo ra kết quả có ý nghĩa. Hoạt động có mục tiêu rõ ràng, thường thể hiện trong tạo ra hay cập nhật các nhiệm vụ cụ thể như là mô hình, lớp hoặc là kế hoạch. Mỗi một hoạt động được gắn với một vai trò cụ thể. Chẳng hạn như hoạt động xem lại thiết kế sẽ được thực hiện bởi vai trò xem xét lại thiết kế (design reviewer), hoạt động tìm các trường hợp sử dụng và tác nhân thực hiện bởi vai trò phân tích hệ thống (system analyst).



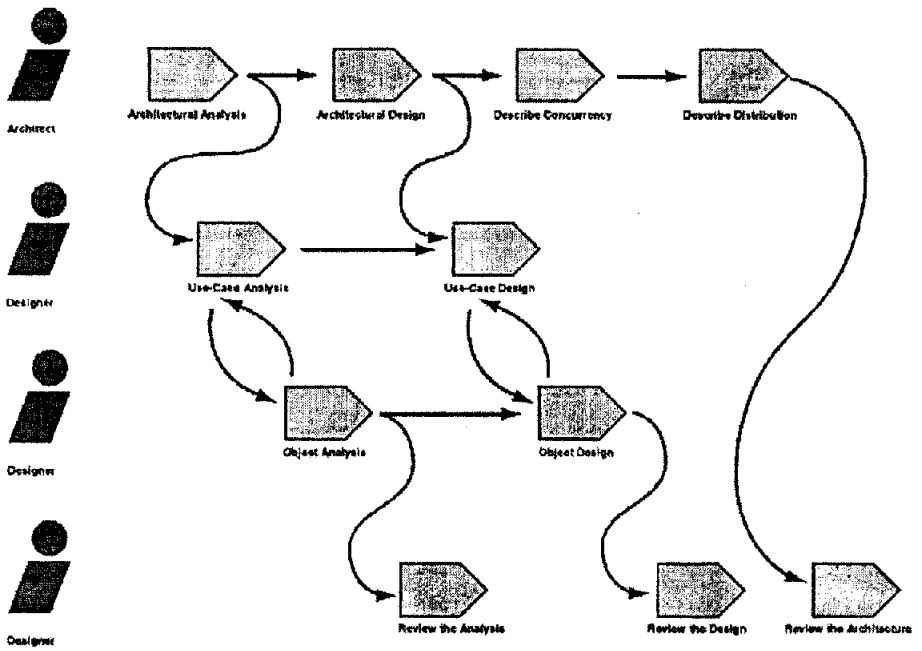
***Hình 36**: Gắn kết con người và vai trò.*

Hoạt động được chia thành các dạng chính như suy nghĩ (hiểu bản chất của nhiệm vụ, thu thập và xem xét đầu vào của nhiệm vụ và xây dựng đầu ra), thực hiện (tạo mới và cập nhật một số nhiệm vụ) và xem xét lại (thanh tra kết quả với một vài tiêu chí). Chẳng hạn như hoạt động "Tìm kiếm các trường hợp sử dụng và các tác nhân" được chia thành 7 bước như (i) tìm tác nhân, (ii) tìm trường hợp sử dụng, (iii) mô tả tương tác giữa tác nhân và trường hợp sử dụng sẽ như thế nào, (iv) đóng gói các trường hợp sử dụng và tác nhân, (v) trình bày mô hình các trường hợp sử dụng trong sơ đồ các trường hợp sử dụng, (vi) phát triển một khảo sát của mô hình các trường hợp sử dụng, (vii) đánh giá kết quả. Các bước suy nghĩ thể hiện từ (i,ii,iii), thực hiện (iv,v,vi) và đánh giá (vii).

Các hoạt động có đầu vào và đầu ra là các tác vụ. Một tác vụ là một mẩu thông tin được sản xuất, cập nhật và sử dụng bởi một tiến trình. Tác vụ là một sản phẩm hữu hình của một dự án phần mềm, là "đồ vật" mà dự án sản xuất ra hay sử dụng trong quá trình làm việc cho hoàn thành sản phẩm cuối cùng. Các tác vụ có thể là một mô hình (trường hợp sử dụng, mô hình thiết kế,...), một thành phần của mô hình (lớp, trường hợp sử dụng, hệ thống con,...), một tài liệu (trường hợp nghiệp vụ, tài liệu kiến trúc phần mềm,...), mã nguồn, tập tin thực thi,....

Khung hoạt động được sử dụng để chứa hay tổ chức các hoạt động của tiến trình. Có 9 dạng khung hoạt động trong RUP và các khung này thể hiện phân vùng của các vai trò và hoạt động vào trong các nhóm theo lĩnh vực quan tâm hay đặc thù. Các dạng khung hoạt động này được chia thành 6 khung hoạt động kỹ thuật và 3 khung hoạt động hỗ trợ. Các khung hoạt động kỹ thuật gồm có mô hình hóa nghiệp vụ, yêu cầu, phân tích và thiết kế, cài đặt, kiểm thử, triển khai. Các khung hoạt động hỗ trợ gồm có quản lý dự án, quản lý cấu hình và thay đổi, môi trường.

Các vai trò, hoạt động và tác vụ chưa đủ để tạo thành một tiến trình. Do đó cần có cách thức mô tả các chuỗi hoạt động tạo ra các kết quả có giá trị và hiển thị sự tương tác giữa các vai trò. Một dòng công việc là một chuỗi các hoạt động tạo ra một kết quả là một giá trị quan sát được. Chẳng hạn như trong UML (Unified Modeling Language) thì một dòng hoạt động có thể được biểu diễn bằng một sơ đồ tuần tự, sơ đồ phối hợp hay một sơ đồ hoạt động. RUP có ba dạng dòng công việc chính là dòng công việc cốt lõi, dòng công việc chi tiết và các kế hoạch lặp.



Hình 37: Dòng công việc.

Một số công cụ hỗ trợ cần sử dụng trong quá trình sản xuất phần mềm như (http://www.ibm.com): Rational Method Composer, Rational Requisite®Pro, Rational ClearQuest™, Rational Rose® 98, Rational SoDA®, Rational Purify®, Rational Visual Quantify™, Rational Visual PureCoverage™, Rational TeamTest — Creates, Rational PerformanceStudio™, Rational ClearCase®.

TÀI LIỆU THAM KHẢO

[1] Jim Arlow and Ila Neustadt, *UML And The Unified Process: Practical Object-Oriented Analysis & Design*, Pearson Education, 2002.

[2] Barry W. Boehm, *Software Engineering*, IEEE Computer Society - Wiley, 2007.

[3] Alphonse Carlier, *Le développement du logiciel*, Hermes, 1995.

[4] Alistair Cockburn, *Writing Effective Use-Cases*, Addison-Wesley, 2000.

[5] Scott E. Donaldson and Stanley G. Siegel, *Successful Software Development (2nd edition)*, Prentice Hall, 2000.

[6] IBM, *Essentials of Rational Unified Process (PRJ270)*, 2011.

[7] IBM, *Basic Method Authoring with IBM Rational Method Composer 7.5*, 2011.

[8] IEEE Std 828, *IEEE Standard for Software Configuration Management Plans*, The Institute of Electrical and Electronics Engineers, Inc., 1998.

[9] IEEE, *Guide to the Software Engineering Body of Knowledge - SWEBOK®*, IEEE Computer Society, 2004.

[10] The International Function Users Group, *Function Point Counting Practices Manual (Release 4.1)*, 1999.

[11] Stephen H. Kan, *Metrics and Models in Software Quality Engineering*, Addison-Wesley, 2002.

[12] Per Kroll and Philippe Kruchten, *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*, Addison Wesley, 2003.

[13] Philippe Kruchten, *The Rational Unified Process: An Introduction (2nd, 3rd editions)*, Addison Wesley, 2000, 2003.

[14] Craig Larman, *Agile and Iterative Development: A Manager's Guide*, Addison Wesley, 2003.

[15] Timothy C. Lethbridge and Robert Laganière, *Object-Oriented Software Engineering: Practical Software Development Using UML and Java*, McGraw-Hill, 2002.

[16] Raymond J. Madachy, *Software Process Dynamics*, IEEE Press – Wiley, 2008.

[17] Nguyễn Minh Hồng, "Hướng dẫn xác định trị giá phần mềm", *Công văn số 3364/BTTTT-UDCNTT ngày 17 tháng 10 năm 2008*, Bộ Thông tin & Truyền thông, 2008.

[18] Nguyễn Minh Hồng, "Sửa đổi giá trị trọng số BMT", *Công văn số 2496/BTTTT-UDCNTT ngày 04 tháng 08 năm 2010*, Bộ Thông tin & Truyền thông, 2010.

[19] Rational Software White Paper, *Rational Unified Process® for Systems Engineering RUP® SE1.1*, Rational Software Corporation, TP 165A, 2002.

[20] Rational Software White Paper, *Rational Unified Process: Best Practices for Software Development Teams*, IBM Corporation, TP026B, Rev 11/01, 2001.

[21] Rational Software White Paper, *Reaching CMM Levels 2 and 3 with the Rational Unified Process*, Rational Software Corporation, 2000.

[22] John W. Rittinghouse, *Managing Software Deliverables: A Software Development Management Methodology*, Digital Press – Elsevier, 2004.

[23] Robert E. Park, *Software Size Measurement: A Framework for Counting Source Statements*, Technical Report CMU/SEI-92-TR-020 ESC-TR-92-020, 1996.

[24] Roger S. Pressman, *Software Engineering: A Practitioner's Approach (6^{th} edition)*, McGraw-Hill, 2005.

[25] Stephen R. Schach, *Object-Oriented and Classical Software Engineering (5^{th},6^{th},7^{th} editions)*, McGraw-Hill, 2002, 2005, 2007.

[26] Stephen R. Schach, *Object-Oriented and Classical Software Engineering (5^{th} editions)*, McGraw-Hill, 2002.

[27] Ian Sommerville, *Software Engineering (6^{th},8^{th} editions)*, Addison-Wesley, 2001, 2006.

[28] Jeff Tian, *Software Quality Engineering: Testing Quality Assurance and Quantifiable Improvement*, IEEE Computer Society - Wiley, 2005.

[29] Bhuvan Unhelkar, *Pratical Object Oriented Design*, Thompson Social Science Press, 2005.

[30] Hans van Vliet, *Software Engineering: Principals and Practice (2^{nd} edition)*, Wiley, 2000.

[31] http://www-01.ibm.com/software/awdtools/rup/

CÂU HỎI HƯỚNG DẪN ÔN TẬP

1. Phân biệt vai trò và tác vụ trong RUP?
2. Phân biệt tác động của lặp và tăng trưởng trong RUP?

ĐỊNH HƯỚNG THẢO LUẬN

1. Phân biệt các quy trình công nghệ phần mềm hiện nay?
2. Ảnh hưởng của chất lượng phần mềm trong việc thực hiện quy trình công nghệ phần mềm?

BÀI TẬP THỰC HÀNH

1. Tải về, cài đặt và sử dụng công cụ RMC (Rational Method Composer) của IBM.
2. Tìm kiếm, tải về và cài đặt các công cụ thể hiện quy trình phần mềm.

CHƯƠNG 4

ƯỚC LƯỢNG CHI PHÍ

1. XÁC ĐỊNH KÍCH THƯỚC PHẦN MỀM

1.1 LOC

Kích thước phần mềm có thể được xác định dựa theo số dòng mã lệnh đếm được trong mã nguồn của phần mềm (lines of code - LOC). Khi số dòng mã lệnh của một phần mềm là khá lớn thì có thể sử đếm theo đơn vị ngàn dòng (thousand lines of code – KLOC, thousand delivered source instructions - KDSI).

Các vấn đề cần quan tâm khi tính toán kích thước cho các giai đoạn khác nhau như phân tích yêu cầu, cài đặt trên hai ngôn ngữ lập trình khác nhau (C, Java, Lisp,...), cách đếm số dòng mã lệnh (mã lệnh thực thi, định nghĩa dữ liệu,...), mã lệnh tạo công cụ dùng để hỗ trợ phát triển, sinh mã tự động, thiết kế giao diện trực tiếp (GUI),...

Một cách khác có thể được sử dụng bằng cách đếm theo số lượng toán tử (operators) và toán hạng (operands).

1.2 FFP

Kích thước phần mềm còn có thể được xác định dựa trên *số lượng chức năng* mà phần mềm sẽ thực hiện. Một chức năng "đếm được" sẽ được tính như là một điểm chức năng (function point – FP). Một điểm chức năng cũng thường được quy đổi về một số lượng dòng mã lệnh nhất định, tùy theo ngôn ngữ lập trình sử dụng. Thông thường một phần mềm sẽ được chia thành các dạng chức năng khác nhau.

Đối với phương pháp FFP (Files-Flows-Processes) áp dụng cho các phần mềm xử lý dữ liệu có kích thước trung bình sử dụng từ 2 đến 10 người/năm thì các dạng chức năng chính bao gồm tập tin (files - Fi), dòng (flows - Fl) và xử lý (processes - Pr). Fi được hiểu là tập hợp các mẩu tin (vật lý hay luận lý) có liên hệ với nhau; Fl là giao diện dữ liệu giữa sản phẩm và môi trường như màn hình, báo cáo,...; còn Pr về chức năng mà nói đó chính là một định nghĩa logic hay toán học dùng để thao tác trên dữ liệu. Khi đó kích thước phần mềm S sẽ là tổng số chức năng đếm được:

$$S = Fi + Fl + Pr$$

1.3 FP

Còn đối với phương pháp tính các điểm chức năng (funtion points - FP), các điểm chức năng tính được sẽ giúp (i) ước lượng chi phí trong các giai đoạn thiết kế, viết mã lệnh và kiểm thử phần mềm; (ii) dự đoán số lượng lỗi gặp phải trong quá trình kiểm thử và (iii) dự báo số lượng thành phần và số lượng mã nguồn cài đặt trong hệ thống.

Các điểm chức năng được xác định dựa trên 5 yếu tố sau với trọng số tương ứng:

- Số lượng các đầu vào từ bên ngoài hệ thống (external inputs - EI), do người dùng, ứng dụng khác cung cấp hoặc là thông tin điều khiển, thường được sử dụng để cập nhật các tập tin logic nội tại.
- Số lượng các đầu ra bên ngoài hệ thống (external output - EO), thường được lấy từ bên trong hệ thống và cung cấp thông tin cho người sử dụng (báo biểu, màn hình, thông báo lỗi,...).
- Số lượng các truy vấn ngoài (external inquiries - EQ), được xem như mục dữ liệu trực tuyến (online input) và trả về kết quả dạng dữ liệu trực tuyến (online output).
- Số lượng các tập tin logic nội tại (internal logical files – ILF), là nhóm dữ liệu dạng logic nằm bên trong ứng dụng và được bảo trì thông qua các các đầu vào bên ngoài hệ thống.
- Số lượng các tập tin giao diện ngoài (external interface files - EIF), là nhóm dữ liệu logic nằm bên ngoài ứng dụng, các ứng dụng khác có thể sử dụng dữ liệu này.

Bảng 2: Trọng số cho các dạng chức năng.

Function Type	Weighting factors		
	Simple	Average	Complex
External Inputs	3	4	6
External Ouputs	4	5	7
External Inquiries	3	4	6
Internal Logical Files	7	10	15
External Interface Files	5	7	10

Bảng 3: Các nhân tố ảnh hưởng đến xác định điểm chức năng.

v_i	Value adjustement factors	Range
v_1	Data communications	[0..5]
v_2	Distributed data processing	[0..5]
v_3	Performance criteria	[0..5]
v_4	Heavily utilized hardware	[0..5]
v_5	High transaction rates	[0..5]
v_6	Online data entry	[0..5]
v_7	End-user effciency	[0..5]
v_8	Online updating	[0..5]
v_9	Complex computations	[0..5]
v_{10}	Reusability	[0..5]
v_{11}	Ease of installation	[0..5]
v_{12}	Ease of operation	[0..5]
v_{13}	Portability	[0..5]
v_{14}	Maintainability	[0..5]
	$VAF = \sum_{i=1}^{14} v_i$	[0..70]

Tùy theo các ứng dụng khác nhau, với các mức độ phức tạp khác nhau, các điểm chức năng có thể được nhân với các hệ số phản ánh mức độ phức tạp của phần mềm. Cùng với sự phát triển của môi trường thực hiện phần mềm, số lượng điểm chức năng có thể được xác định chính xác hơn trên cơ sở áp dụng các nhân tố hiệu chỉnh (value adjusted factors) $VAF = \sum_{i=1}^{14} v_i$. Giá trị của các nhân tố thay đổi biến thiên từ 0 đến 5 tùy theo mức độ ảnh hưởng. Khi đó số lượng điểm chức năng sẽ là $FP = UFP \times TCF$ với UFP (unadjusted funtion points) là tổng các điểm chức năng đếm được trước khi hiệu chỉnh và TCF là độ phức tạp kỹ thuật được tính theo công thức $TCF = 0.65 + 0.01 \times VAF$.

Bảng 4: Xác định độ khó cho các dạng chức năng tập tin.

Số lượng kiểu tập tin	Số lượng phần tử dữ liệu		
	1-4	5-15	>15
0|1	Đơn giản	Đơn giản	Trung bình
2-3	Đơn giản	Trung bình	Phức tạp
>3	Trung bình	Phức tạp	Phức tạp

Bảng 5: Chuyển đổi giữa LOC và FP (tính trung bình).

Programming Language	LOCs/FP
Access	38
Ada 83	71
Ada 95	49
AI Shell	49
APL	32
Assembly - Basic	320
Assembly - Macro	213
Basic - ANSI	64
Basic - Compiled	91
Basic - Visual	32
C	128
C++	55
Cobol (ANSI 85)	91
Database – Default	40
Fifth Generation Language	4
First Generation Language	320
Forth	64
Fortran 77	107
Fortran 95	71
Fourth Generation Language	20
High Level Language	64
HTML 3.0	15
Java	53

Programming Language	LOCs /FP
Jovial	107
Lisp	64
Machine Code	640
Modula 2	80
Pascal	91
PERL	27
PowerBuilder	16
Prolog	64
Query – Default	13
Report Generator	80
Second Generation Language	107
Simulation – Default	46
Spreadsheet	6
Third Generation Language	80
Unix Shell Scripts	107
USR_1	1
USR_2	1
USR_3	1
USR_4	1
USR_5	1
Visual Basic 5.0	29
Visual C++	34

Để thuận tiện trong quá trình xác định kích thước phần mềm, một hệ thống quy đổi tương đương giữa một chức năng và số dòng mã lệnh của một ngôn ngữ lập trình được thiết lập (xem Bảng 5, Bảng 6).

Bảng 6: Chuyển đổi giữa LOC và FP.

Programming language	LOC per function point			
	Average	Median	Low	High
Access	35	38	15	47
Ada	154	-	104	205
APS	86	83	20	184
ASP 69	62	-	32	127
Assembler	337	315	91	694
C	162	109	33	704
C++	66	53	29	178
Clipper	38	39	27	70
COBOL	77	77	14	400
Cool:Gen/IEF	38	31	10	180
Culprit	51	-	-	-
DBase IV	52	-	-	-
Easytrieve+	33	34	25	41
Excel47	46	-	31	63
Focus	43	42	32	56
FORTRAN	-	-	-	-
Foxpro	32	35	25	35
Ideal	66	52	34	203
IEF/Cool:Gen	38	31	10	180
Informix	42	31	24	57
Java	63	53	77	-
JavaScript	58	63	42	75
JCL	91	123	26	150
JSP	59	-	-	-
Lotus Notes	21	22	15	25
Mantis	71	27	22	250
Mapper	118	81	16	245

Natural	60	52	22	141
Oracle	30	35	4	217
PeopleSoft	33	32	30	40
Perl	60	-	-	-
PL/1	78	67	22	263
PowerBuilder	32	31	11	105
REXX	67	-	-	-
RPG II/III	61	49	24	155
SAS	40	41	33	49
Smalltalk	26	19	10	55
SQL	40	37	7	110
VBScript36	34	27	50	-
Visual Basic	47	42	16	158

1.4 OP

Với tiếp cận hướng đối tượng (object points - OP) thì kích thước phần mềm được xác định theo sẽ dựa trên số lượng các kịch bản, các lớp chính, lớp hỗ trợ, tỷ lệ giữa số lượng lớp hỗ trợ và lớp chính và số lượng các hệ thống con.

Bảng 7: Trọng số cho dạng đối tượng giao diện.

Kiểu giao diện	Hệ số nhân
Không có đồ họa	2.00
Dựa trên văn bản	2.25
Đồ họa	2.50
Đồ họa phức tạp	3.00

1.5 EFP

Kích thước phần mềm được xác định theo dạng điểm chức năng mở rộng (extended funtion points - EFP) sẽ giúp tính toán chính xác hơn giá trị các điểm chức năng. Khi đó phần mềm được chia thành các giao dịch (transactions), mỗi giao dịch bao gồm các thành phần như đầu vào (input), tiến trình/xử lý (process), đầu ra (output). Số lượng điểm chức năng sẽ được tính dựa trên số lượng các thành phần trong từng giao dịch.

1.6 FTP

Tổng các đặc điểm (feature points - FTP) có thể được sử dụng để xác định kích thước phần mềm. Cách tính này thường được sử dụng cho các phần mềm chịu ảnh hưởng mạnh về giải thuật như các phần mềm thời gian thực (real-time software), phần mềm nhúng (embedded software), phần mềm truyền thông (communication software).

$$FTP = FP - 3 \times Maf + 3 \times Alg$$

Trong đó FP là số lượng điểm chức năng tính toán theo phương pháp thông thường, Maf là số lượng tập tin chính và Alg là số lượng giải thuật sử dụng.

1.7 UCP

Bên cạnh đó, kích thước phần mềm có thể được xác định dựa trên số lượng trường hợp sử dụng (use case points - UCP) có được của hệ thống. Số lượng dòng mã lệnh ước lượng theo các trường hợp sử dụng:

$$LOC_{estimate} = N \times LOC_{avg} + [(\frac{S_a}{S_h} - 1) + (\frac{P_a}{P_h} - 1)] \times LOC_{adjust}$$

với

N = số lượng trường hợp sử dụng hiện hành

LOC_{avg} = số lượng LOC trung bình cho hệ thống cùng kiểu

LOC_{adjust} = thể hiện sự điều chỉnh dựa trên n phần trăm của LOC_{avg} với n được định nghĩa cục bộ và thể hiện sự khác nhau giữa phần mềm này với các phần mềm khác (tính trung bình)

S_a = số lượng kịch bản hiện tại cho mỗi trường hợp sử dụng

S_h = số lượng kịch bản trung bình cho mỗi trường hợp sử dụng cho hệ thống cùng kiểu

P_a = số lượng trang hiện tại cho mỗi trường hợp sử dụng

P_h = số lượng trang trung bình cho mỗi trường hợp sử dụng cho hệ thống cùng kiểu

Xét phần mềm có 3 hệ thống con. Hệ thống con thứ nhất có 6 trường hợp sử dụng, mỗi trường hợp sử dụng được mô tả bởi tối đa là 10 kịch bản và có số trang mô tả trung bình là 6. Hệ thống con thứ hai được mô tả bởi 10 trường hợp sử dụng, mỗi trường hợp sử dụng có tối đa là 20 kịch bản với số trang trung bình là 8. Hệ thống con cuối cùng có 5 trường hợp sử dụng, mỗi trường hợp sử dụng có số kịch bản trung bình là 6 với số trang trung bình là 5. Tỷ lệ phần trăm n được xác định là 30.

Bảng 8: Ví dụ về tính kích thước phần mềm cho các trường hợp sử dụng.

	Dữ liệu hiện tại			Dữ liệu lịch sử			
	Trường hợp sử dụng	Kịch bản	Trang	Kịch bản	Trang	LOC	$LOC_{estimate}$
Hệ thống 1	6	10	6	12	5	560	3366
Hệ thống 2	10	20	8	16	8	3100	31233
Hệ thống 3	5	6	5	10	6	1650	7970
						Tổng $LOC_{estimate}$	42568

Giả sử khả năng nhân công trung bình của hệ thống là 620 LOC/người-tháng, chi phí lao động là 8000USD/tháng, chi phí trung bình cho một LOC là 13USD. Dựa trên ước lượng theo trường hợp sử dụng và dữ liệu lịch sử, chi phí ước lượng của sản phẩm là 552000USD và nhân công cần là 68 người-tháng.

Trên cơ sở tiếp cận web thì kích thước phần mềm được xác định dựa trên số lượng các trang web tĩnh, các trang web động, các liên kết nội tại, các đối tượng dữ liệu, các giao diện với hệ thống bên ngoài, các nội dung tĩnh, các nội dung động và số hàm thực thi. Chẳng hạn như ta có thể định nghĩa một chỉ số tùy biến C dựa trên số lượng các trang web tĩnh (N_{sp}) và số lượng các trang web động (N_{dp}) $C = \frac{N_{dp}}{(N_{dp} + N_{sp})}$.

2. ĐÁNH GIÁ MỨC ĐỘ TIN CẬY

Độ tin cậy (realibility) của phần mềm có thể được xác định bằng thời gian trung bình giữa các lỗi (mean-time-between-failure)

$$MTBF = MTTF + MTTR$$

với MTTF là thời gian trung bình để có lỗi xảy ra (mean-time-to-failure) và MTTR là thời gian trung bình để sửa chữa lỗi (mean-time-to-repair).

3. ĐÁNH GIÁ MỨC ĐỘ SẴN SÀNG

Độ sẵn sàng (availability) của phần mềm có thể được xác định theo công thức:

$$Availability = \frac{MTTF}{(MTTF + MTTR)} \times 100\%$$

4. ĐÁNH GIÁ THỜI GIAN THAY ĐỔI

Đánh giá thời gian thay đổi có thể dựa trên chỉ số DRE (Defect Removal Efficiency) để tính toán thời gian trôi qua và tổng nỗ lực để xác định tác động của các hoạt động đảm bảo chất lượng trên thời gian và nỗ lực cần có để tạo một sự thay đổi trên phần mềm.

$$DRE = \frac{E_{change}}{E_{change} + D_{change}}$$

Một số chỉ số cần quan tâm:

t_{queue} = Thời gian (giờ hoặc ngày) trôi qua kể từ khi một yêu cầu thay đổi được thực hiện cho đến khi được đánh giá chấp nhận.

W_{eval} = Nỗ lực/nhân công (người-giờ) để thực hiện đánh giá.

t_{val} = Thời gian (giờ hoặc ngày) trôi qua kể từ khi hoàn tất đánh giá đến khi lệnh thay đổi đến được người thực hiện.

W_{change} = Nỗ lực (người-giờ) cần có để thực hiện thay đổi.

t_{change} = Thời gian cần có (giờ hoặc ngày) để thực hiện thay đổi.

E_{change} = Lượng lỗi phát hiện trong quá trình làm việc để thực hiện thay đổi.

D_{change} = Lượng lỗi phát hiện sau khi thay đổi được phản hồi từ khách hàng.

5. ĐÁNH GIÁ PHÂN TÍCH YÊU CẦU

Dựa trên số lượng yêu cầu n_r trong một đặc tả, $n_r = n_f + n_{nf}$ với n_f là số lượng yêu cầu chức năng và n_{nf} là số lượng yêu cầu không chức năng.

- Độ cô đọng (specificity) hay súc tích của một đặc tả (ít nhầm lẫn) là $Q_1 = \frac{n_{ui}}{n_r}$ với n_{ui} là số lượng các yêu cầu được thẩm định là tốt. Mức độ nhầm lẫn giảm dần khi giá trị của Q_1 tiến về 1.
- Độ hoàn chỉnh (completness) của yêu cầu chức năng là tỷ lệ $Q_2 = \frac{n_u}{(n_i \times n_s)}$ với n_u là số lượng các yêu cầu chức năng duy nhất, n_i là số lượng đầu vào và n_s là số lượng các trạng thái được đặc tả.
- Mức độ hợp lệ của các yêu cầu được xác định bằng $Q_3 = \frac{n_c}{(n_c + n_{nv})}$ với n_c là số lượng yêu cầu được thẩm định là đúng và n_{nv} là số lượng yêu cầu chưa được thẩm định.

6. ĐÁNH GIÁ THIẾT KẾ KIẾN TRÚC

Độ phức tạp trong thiết kế kiến trúc có thể được xác định trên 3 nội dung về cấu trúc, dữ liệu và hệ thống.

- Đối với kiến trúc phân cấp (hierarchical architecture), độ phức tạp cấu trúc được xác định bằng $S(i) = f_{out}^2(i)$ với $f_{out}(i)$ =*fan-out(i)* là số lượng mô đun được gọi bởi mô đun i (còn *fan-in(i)* là số lượng mô đun gọi mô đun i).
- Độ phức tạp dữ liệu $D(i) = \frac{v(i)}{[f_{out}(i)+1]}$ xác định mức độ phức tạp bên trong (internal interface) của mô đun i, với v(i) là số lượng biến đầu vào và đầu ra.
- Độ phức tạp kiến trúc của hệ thống được xác định bằng tổng độ phức tạp cấu trúc và độ phức tạp dữ liệu $C(i) = S(i) + D(i)$.

Bên cạnh đó ta có thể đánh giá độ phức tạp kiến trúc theo hướng hình thái học (morphology) bằng cách tính tổng số nút và số cung của một thiết kế kiến trúc $size = n + a$ và tỷ lệ cung so với nút $r = \frac{a}{n}$.

Ngoài ra ta có thể tính chỉ số chất lượng cấu trúc thiết kế (Design Structure Quality Index – DSQI) với giá trị biến thiên trong khoảng [0-1].

S_1= tổng số các mô đun định nghĩa trong kiến trúc chương trình

S_2= số lượng các mô đun mà sự chính xác trong hoạt động phụ thuộc vào dữ liệu đầu vào hoặc xuất ra các dữ liệu được sử dụng ở chỗ khác (không tính các mô đun điều khiển)

S_3= số lượng các mô đun mà sự chính xác trong hoạt động phụ thuộc vào xử lý ưu tiên

S_4= số lượng mục trong cơ sở dữ liệu (bao gồm cả đối tượng dữ liệu và tất cả các thuộc tính định nghĩa đối tượng)

S_5= tổng số mục dữ liệu duy nhất trong cơ sở dữ liệu

S_6= số lượng các phân mảnh dữ liệu (các mẫu tin hoặc đối tượng riêng lẻ khác nhau)

S_7= số lượng các mô đun với đầu vào và đầu ra đơn lẻ (ngoại trừ xử lý không được xem như đa đầu ra)

Ta có thể xác định được:

- Cấu trúc chương trình (program structure) D_1: D_1=1 nếu thiết kế kiến trúc được phát triển dựa trên một một phương thức khác biệt, ngược lại thì D_1=0.

- Sự độc lập của mô đun D_2: $D_2 = 1 - \frac{S_2}{S_1}$.

- Mô đun không phụ thuộc vào xử lý ưu tiên D_3: $D_3 = 1 - \frac{S_3}{S_1}$

- Kích thước cơ sở dữ liệu D_4: $D_4 = 1 - \frac{S_5}{S_4}$

- Sự phân cách dữ liệu D_5: $D_5 = 1 - \frac{S_6}{S_4}$

- Đặc điểm mô đun vào/ra D_6: $D_6 = 1 - \frac{S_7}{S_1}$

$DSQI = \sum_{i=1}^{6} w_i D_i$ và w_i là trọng số phản ánh sự quan trọng của các giá trị D_i, $\sum w_i = 1$.

7. ĐÁNH GIÁ THIẾT KẾ KIẾN TRÚC HƯỚNG ĐỐI TƯỢNG

Mức độ thừa kế của các lớp trong thiết kế kiến trúc được xác định thông qua phương pháp tính nhân tố thừa kế (Method Inheritance Factor) $MIF = \frac{\sum_{i=1}^{T_C} M_i(C_i)}{\sum_{i=1}^{T_C} M_a(C_i)}$,

với T_C là tổng số lượng các lớp trong kiến trúc, C_i là lớp bên trong kiến trúc và

$$M_a(C_i) = M_d(C_i) + M_i(C_i)$$

trong đó

$M_a(C_i)$ = số lượng các phương thức có thể được gọi gắn với C_i.

$M_d(C_i)$ = số lượng các phương thức khai báo trong lớp C_i.

$M_i(C_i)$ = số lượng các phương thức thừa kế (không tính các phương thức xếp chồng) trong C_i.

Mức độ nối kết (Coupling Factor) CF được xác định bằng công thức dưới đây, với is_client =1 nếu có mối liên hệ tồn tại giữa lớp khách (client class) C_C và lớp chủ (server class) C_S ($C_C \neq C_S$), ngược lại thì is_client=0.

$$CF = \sum_{i=1}^{T_C} \sum_{j=1}^{T_C} \frac{is\_client(C_i, C_j)}{(T_C^2 - T_C)}$$

8. ĐÁNH GIÁ THIẾT KẾ CHI TIẾT

Mức độ nối kết (coupling) của một phân hệ với các phân hệ khác, với dữ liệu toàn cục và với môi trường bên ngoài. Độ nối kết được tính như sau dựa trên:

d_i = số lượng tham số dữ liệu đầu vào

c_i = số lượng tham số điều khiển đầu vào

d_0 = số lượng tham số dữ liệu đầu ra

c_0 = số lượng tham số điều khiển đầu ra

Mức độ nối kết chung (global coupling):

g_d = số lượng các biến toàn cục sử dụng như dữ liệu

g_c = số lượng các biến toàn cục sử dụng như điều khiển

Mức độ nối kết môi trường (environmental coupling):

w = số lượng các phân hệ đã gọi (fan-out)

r = số lượng các phân hệ gọi đến (fan-in)

Khi đó độ nối kết m_c được xác định theo công thức $m_c = \frac{k}{M}$ với k là hằng số tỷ lệ và M được xác định bằng $M = d_i + (a \times c_i) + d_0 + (b \times c_0) + g_d + (c \times g_c) + w + r$, các giá trị k, a, b và c phải được xác định bằng thực nghiệm.
Khi giá trị của m_c tăng sự nối kết tổng thể của hệ thống giảm, do đó để có thể sự tương đồng về giá trị của độ nối kết, một cách xác định khác là $C = 1 - m_c$.

9. ĐÁNH GIÁ CÀI ĐẶT

Kích thước mã nguồn của một sản phẩm phần mềm có thể được đo theo kỹ thuật Halstead nhằm xác định chiều dài của chương trình. Với các thông số:

- n_1 = số lượng các toán tử khác nhau;
- n_2 = số lượng các toán hạng khác nhau;
- N_1 = số lượng các toán tử;
- N_2 = số lượng các toán hạng.

Sẽ xác định được:

- Độ dài mã nguồn được tính theo công thức: $N = n_1 \log_2 n_1 + n_2 \log_2 n_2$
- Khối lượng chương trình theo đơn vị bits được tính bởi: $V = N \log_2(n_1 + n_2)$
- Tỷ lệ khối lượng tối thiểu: $L = \frac{2}{n_1} \times \frac{n_2}{N_2}$
- Số lượng lỗi dự đoán (/KDSI): $B = \frac{(N_1 + N_2)\log_2(n_1 + n_2)}{3000}$

10. ĐÁNH GIÁ KIỂM THỬ

Sử dụng khối lượng chương trình V và mức chương trình (program level) PL theo Halstead để tính nỗ lực kiểm thử $PL = \frac{1}{\frac{N_1}{n_1} \times \frac{N_2}{n_2}}$, $e = \frac{V}{PL}$

Tỷ lệ phần trăm của toàn bộ các nỗ lực kiểm thử (percentage of testing effort) *pte(k)* cần được xác định cho một phân hệ *k* (mô đun) được xác định theo mối quan hệ sau: $pte(k) = e(k) / \sum e(i)$ với e(k) được tính theo công thức phía trên và các ∑e(i) là tổng các nỗ lực kiểm thử tính theo công thức Halstead.

Công thức xác định thời gian cần có để kiểm thử: $\frac{\ln\left(\frac{f_{target}}{0.5+f_{target}}\right)}{\ln\left(\frac{0.5+f_{target}}{f_{total}+f_{target}}\right)} \times t_h$

trong đó f_{target} là số lượng lỗi dự đoán, f_{total} là số lượng lỗi dự đoán thực sự xảy ra sau đó, t_h là thời gian kiểm thử xảy ra lỗi và ln là logarith theo cơ số e.

Xét ví dụ một sản phẩm có 50000 LOC, hợp đồng qui định mỗi KDSI có ít hơn 0.02 lỗi. Sản phẩm được kiểm thử 400 giờ, trong thời gian này có 20 lỗi xảy ra và đã thực thi 50 giờ kể từ lỗi cuối cùng. Ta có: $f_{target} = 0.02 \times 50KDSI = 1$, $f_{total} = 20$, $t_h = 400 - 50 = 350$ giờ. Kết quả là 54 giờ (thiếu 4 giờ) nên phải kiểm thử thêm 4 giờ nữa, nếu trong 4 giờ này có lỗi xảy ra thì phải tiếp tục áp dụng công thức. Lặp lại thao tác này cho đến khi không còn thời gian thiếu nữa.

11. ĐÁNH GIÁ BẢO TRÌ

Sự bền vững của sản phẩm phần mềm được xác định bằng chỉ số SMI (Software Maturity Index) [IEEE Standard 982], giá trị chỉ số này càng gần 1.0 thì sản phẩm sẽ được xem như bền vững:

$$SMI = \frac{M_T - (F_a + F_c + F_d)}{M_T}$$

với M_T là số lượng các mô đun, F_c là số lượng các mô đun bị thay đổi, F_a là số lượng các mô đun được thêm vào và F_d là số lượng các mô đun bị xóa đi.

12. ƯỚC LƯỢNG THỰC NGHIỆM

Mô hình ước lượng trên cơ sở phân tích hồi quy là $E = A + B \times (e_v)^C$ với A, B, C là các hằng số thực nghiệm, E là nhân công tính trên đơn vị người-tháng và e_v là biến ước lượng (theo FP hay LOC). E có thể được hiệu chỉnh trong một số trường hợp cho chính xác hơn.

$E = 5.2 \times (KDSI)^{0.91}$	mô hình Walston-Felix
$E = 5.5 + 0.73 \times (KDSI)^{1.16}$	mô hình Bailey-Basili
$E = 3.2 \times (KDSI)^{1.05}$	mô hình Boehm cơ bản
$E = 5.288 \times (KDSI)^{1.047}$	mô hình Doty với KDSI > 9
$E = -91.4 + 0.355 \times (FP)$	mô hình Albrecht và Gaffney
$E = -37 + 0.96 \times (FP)$	mô hình Kemerer
$E = -12.88 + 0.405 \times (FP)$	mô hình Albrecht và Gaffney

Một số công thức tính thời gian thực hiện:

$T = 2.5 \times (E)^{0.35}$	mô hình Walston-Felix
$T = 2.4 \times (E)^{1/3}$	mô hình Putnam
$T = 2.5 \times (E)^{0.38}$	mô hình COCOMO (organic)
$T = 3.0 \times (E)^{0.33+0.2\times(b-1.01)}$	mô hình COCOMO 2.0 (nominal schedule)

13. COCOMO

13.1 COCOMO cơ bản

Mô hình COCOMO cơ bản dành cho các phần mềm có mức độ phức tạp và chi tiết ở mức trung bình. Ước lượng dựa trên hai yếu tố chiều dài sản phẩm phần mềm (KLOC/KDSI, FP, OP) và mức độ khó khi phát triển sản phẩm. Có 3 mức độ khó được cho theo bảng sau:

Bảng 9: Mức độ khó khi phát triển sản phẩm (COCOMO cơ bản).

Development mode	a_b	b_b	c_b	d_b
Organic	2.40	1.05	2.50	0.38
Semi-detached	3.00	1.12	2.50	0.35
Embedded	3.60	1.20	2.50	0.32

Các ước lượng về nhân công, thời gian thực hiện và lượng người như sau:

Nominal Effort [E] $= a_b \times (KDSI)^{b_b}$ <man-month>

Development Time [T] $= c_b \times (E)^{d_b}$ <month>

People Required [P] $= \frac{E}{T}$ <man>

13.2 COCOMO trung gian

Bảng 10: Mức độ khó khi phát triển sản phẩm (COCOMO trung gian).

Development mode	a_b	b_b	c_b	d_b
Organic	3.20	1.05	2.50	0.38
Semi-detached	3.00	1.12	2.50	0.35
Embedded	2.80	1.20	2.50	0.32

Các công thức:

Nominal Effort [E] $= a_b \times (KDSI)^{b_b} \times EAF$

Development Time [T] $= c_b \times (E)^{d_b}$

People Required [P] $= \frac{E}{T}$

Hệ số điều chỉnh nhân công [EAF]: $EAF = \prod_{i=1}^{15} f_i$

Bảng 11: Các hệ số nhân của mô hình COCOMO trung gian.

Cost drivers	f_i	Ratings					
		Very Low	Low	Nominal	High	Very High	Extra High
Product attributes							
RELY — Required software reliability	f_1	0.75	0.88	1.00	1.15	1.40	
DATA — Database size	f_2		0.94	1.00	1.08	1.16	
CPLX — Product complexity	f_3	0.70	0.85	1.00	1.15	1.30	1.65
Computer attributes							
TIME — Execution time constraint	f_4			1.00	1.11	1.30	1.66
STOR — Main storage constraint	f_5			1.00	1.06	1.21	1.56
VIRT — Virtual machine volatility*	f_6		0.87	1.00	1.15	1.30	
TURN — Computer turnaround time	f_7		0.87	1.00	1.07	1.15	
Personnel attributes							
ACAP — Analyst capabilities	f_8	1.46	1.19	1.00	0.86	0.71	
AEXP — Applications experiences	f_9	1.29	1.13	1.00	0.91	0.82	
PCAP — Programmer capabilitiy	f_{10}	1.42	1.17	1.00	0.86	0.70	
VEXP — Virtual machine experience*	f_{11}	1.21	1.10	1.00	0.90		
LEXP — Programming language experiences	f_{12}	1.14	1.07	1.00	0.95		
Project attributes							
MODP — Use of modern programming pratices	f_{13}	1.24	1.10	1.00	0.91	0.82	
TOOL — Use of software tools	f_{14}	1.24	1.10	1.00	0.91	0.83	
SCED — Required development schedule	f_{15}	1.23	1.08	1.00	1.04	1.10	
* *The underlying virtual machine:* độ phức tạp về phần cứng hoặc phần mềm để hoàn thành công việc							

13.3 COCOMO 2.0

Mô hình COCOMO 2.0 nhằm xác định kích thước phần mềm với số lượng điểm đối tượng được xác định thông qua số lượng màn hình, báo biểu và các chương trình con với trọng lượng ở ba mức đơn giản, trung bình và khó. Được sử dụng ở các thời kỳ Early Prototyping, Early Design và Post-Architecture.

Bảng 12: Các giá trị ước lượng cho COCOMO 2.0.

Model	Optimistic estimate	Pessimistic estimate
Early Prototyping	0.50 E	2.00 E
Early Design	0.67 E	1.50 E
Post-Architecture	0.80 E	1.25 E

Hệ số b_b được xác định theo công thức $b_b = 1.01 + 0.01 \times \sum_{i=1}^{5} w_i$, giá trị của b_b sẽ biến thiên từ 1.01 đến 1.26.

Bảng 13: Giá trị các hệ số điều chỉnh cho (COCOMO 2.0).

Scale factors (w_i)	Very low (5)	Low (4)	Nominal (3)	High (2)	Very high (1)	Extra high (0)
Precedentedness	Throughly unpreceden ted	Largely	Somewhat unpreceden ted	Generally familiar	Largely familiar	Thoroughly familiar
Development flexibility	Rigorous	Occasional relaxation	Some relaxation	General conformity	Some conformity	General goals
Architecture /risk resolution	Little (20%)	Some (40%)	Often (60%)	Generally (75%)	Mostly (90%)	Full (100%)
Team cohesion	Very difficult interactions	Some difficult interactions	Basically cooperative interactions	Largely cooperative	Highly cooperative	Seamless interactions
Process maturity	5 minus weighted average of "Yes" answers to CMM Maturity Questionnaire					

Các bước ước lượng điểm đối tượng:

1. Đếm số lượng đối tượng màn hình, báo biểu và số lượng thành phần 3GL (third-generation language).
2. Phân loại các đối tượng vào các mức độ khó đơn gian, trung bình và phức tạp dựa trên các đặc điểm.

Bảng 14: Đếm điểm đối tượng màn hình.

Số lượng các khung nhìn	Tổng cộng < 4 (< 2 màn hình < 3 clnt)	Tổng cộng < 8 (< 2/3 màn hình < 3-5 clnt)	Tổng cộng 8 + (> 3 màn hình > 5 clnt)
< 3	Đơn giản	Đơn giản	Trung bình
3-7	Đơn giản	Trung bình	Khó
> 8	Trung bình	Khó	Khó

Bảng 15: Đếm điểm đối tượng báo biểu.

Số lượng các phần	Tổng cộng < 4 (< 2 srvr < 3 clnt)	Tổng cộng < 8 (< 2/3 màn hình < 3-5 clnt)	Tổng cộng 8 + (> 3 màn hình > 5 clnt)
0 hoặc 1	Đơn giản	Đơn giản	Trung bình
2 hoặc 3	Đơn giản	Trung bình	Khó
4 +	Trung bình	Khó	Khó

3. Xác định trọng lượng.

Bảng 16: Trọng số độ khó các dạng đối tượng (điểm đối tượng).

Object Type	Complexity-Weight		
	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL Component			10

4. Xác định tổng số lượng điểm đối tượng.
5. Ước đoán tỷ lệ phần trăm sử dụng lại mong đợi trong dự án. Xác định lại số lượng các điểm đối tượng:

   NOP (New Object Points) = (Object Points) (100 - % reuse)/100.

6. Xác định tỷ suất năng suất PROD=NOP/person-month như sau:

Bảng 17: Xác định tỷ suất năng suất.

Developers' experience and capability	Very Low	Low	Nominal	High	Very High
ICASE maturity and capability	Very Low	Low	Nominal	High	Very High
PROD	4	7	13	25	50

7. Tính toán nhân công ước đoán: PM = NOP/PROD.

Trong đó:

- NOP: New Object Points (Object Points count adjusted for reuse).
- srvr: số lượng các bảng dữ liệu trên máy chủ sử dụng trong các màn hình hoặc báo biểu.
- clnt: số lượng các bảng dữ liệu tại máy khách sử dụng trong các màn hình hoặc báo biểu.
- % reuse: tỷ lệ phần trăm của các màn hình, báo biểu, và 3GL mô đun sử dụng lại từ các ứng dụng trước đó với tỷ lệ sử dụng lại được đánh giá mức độ quan trọng.

13.4 Sử dụng mô hình COCOMO 2.0 ước lượng trong sử dụng lại

Nếu điều chỉnh k mô đun trong tổng số m mô đun của phần mềm thì số lượng giao diện N giữa các mô đun cần kiểm tra là: $N = k \times (m-k) + k \times \frac{k-1}{2}$.

Thông thường thì tỷ lệ giữa các giai đoạn phát triển, thiết kế, cài đặt và tích hợp được xác định là 40%, 30%, 30%. Khi đó các giai đoạn thiết kế lại (redesign – DM), viết lại mã lệnh (recoding - CM) và tích hợp lại (reintegration - IM) sẽ được sử dụng trong công thức xác định độ dài mã lệnh sẽ thực hiện lại:

$$AAF = 0.4 \times DM + 0.3 \times CM + 0.3 \times IM$$

Khi đó số lượng mã lệnh mới cần phải thực hiện sẽ được ước lượng là

$$AKLOC = KLOC \times \frac{AAF}{100}$$

với KLOC là số lượng dòng mã lệnh đếm được của phần mềm trước khi có đánh giá về sử dụng lại.

Trong trường hợp mức độ hiểu biết về phần mềm SU (software understanding) biến thiên từ 10% đến 50% và mức độ đánh giá và đồng bộ hóa AA (assessment and assimilation) biến thiên từ 0% đến 8% thì số lượng dòng mã lệnh cần phải thực hiện sẽ là:

$$AKLOC = KLOC \times \frac{AAF + SU + AA}{100}$$

Bảng 18: Xác định mức độ hiểu biết SU.

	Very low	Low	Nominal	High	Very high
Structure	Very low cohesion, high coupling, spaghetti code	Moderately low cohesion, high coupling	Reasonably well-structured; some weak areas	High cohesion, low coupling	Strong modularity, information hiding in data/control structures
Application clarity	No match between program and application world views	Some correlation between program and application	Moderate correlation between program and application	Good correlation between program and application	Clear match between program and application world views
Self-descriptiveness	Obscure code: documentation missing, obscure or obsolete	Some code commentary and headers; some useful documentation	Moerate level of code commentary, headers, documentation	Good code commentary and headers; useful documentation; some weak areas	Self-descriptive code; documentation up to date, well-organized, with design rationale
SU increment to AAF	50	40	30	20	10

Bảng 19: Xác định mức độ đánh giá và đồng bộ hóa AA.

AA increment	Level of AA effort
0	None
2	Basic module search and documentation
4	Some module test and evaluation (T&E), documentation
6	Considerable module T&E, documentation
8	Extensive module T&E, documentation

13.5 Đếm số lượng điểm chức năng trong COCOMO 2.0

Nếu đếm các điểm chức năng thì có thể sử dụng các thông số điều chỉnh sau trong COCOMO 2.0:

Bảng 20: Trọng số độ khó trong COCOMO 2.0.

Function Type	Complexity-Weight		
	Low	Average	High
Internal Logical Files (ILF)	7	10	15
External Interface Files (EIF)	5	7	10
External Inputs (EI)	3	4	6
External Ouputs (EO)	4	5	7
External Inquiries (EQ)	3	4	6

Bảng 21: Xác định mức độ cho ILF và EIF.

Records elements	Data elements		
	1-19	20-50	51+
1	Low	Low	Average
2-5	Low	Average	High
6+	Average	High	High

Bảng 22: Xác định mức độ cho EO và EQ.

File types	Data elements		
	1-5	6-19	20+
0 or 1	Low	Low	Average
2-3	Low	Average	High
4+	Average	High	High

Bảng 23: Xác định mức độ cho EI.

File types	Data elements		
	1-4	5-15	16+
0 or 1	Low	Low	Average
2-3	Low	Average	High
3+	Average	High	High

Bảng 24: Các hệ số điều chỉnh trong COCOMO 2.0.

Cost drivers	f_i	Ratings					
		Very low	Low	Nominal	High	Very high	Extra high
Product factors							
RELY — Required software reliability	f_1	0.75	0.88	1.00	1.15	1.39	
DATA — Database size	f_2		0.93	1.00	1.09	1.19	
CPLX — Product complexity	f_3	0.75	0.88	1.00	1.15	1.30	1.66
RUSE — Required reusability	f_4		0.91	1.00	1.14	1.29	1.49
DOCU — Documentation match to life-cycle needs	f_5	0.89	0.95	1.00	1.06	1.13	
Platform factors							
TIME — Execution time constraint	f_6			1.00	1.11	1.31	1.67
STOR — Main storage constraint	f_7			1.00	1.06	1.21	1.57
PVOL — Platform volatility	f_8		0.87	1.00	1.15	1.30	
Personnel factors							
ACAP — Analyst capability	f_9	1.50	1.22	1.00	0.83	0.67	
PCAP — Programmer capabilitiy	f_{10}	1.37	1.16	1.00	0.87	0.74	
AEXP — Application experience	f_{11}	1.22	1.10	1.00	0.89	0.81	
PEXP — Platform experience	f_{12}	1.24	1.10	1.00	0.92	0.84	
LTEX — Language and tool experience	f_{13}	1.25	1.12	1.00	0.88	0.81	
PCON — Personnel continuity	f_{14}	1.24	1.10	1.00	0.92	0.84	
Project factors							
TOOL — Use of software tools	f_{15}	1.24	1.12	1.00	0.86	0.72	
SITE — Multisite development	f_{16}	1.25	1.10	1.00	0.92	0.84	0.78
SCED — Required development schedule	f_{17}	1.29	1.10	1.00	1.00	1.00	

13.6 Chi tiết các yếu tố về sản phẩm trong COCOMO 2.0

Bảng 25: Hệ số điều chỉnh nhân công cho thời kỳ Post-Architecture.

	Very low	Low	Nominal	High	Very high	Extra high
RELY	Slight inconvenience	Low, easily recoverable losses	Moderate, easily recoverable losses	High financial loss	Risk to human life	
DATA		DB bytes/Pgm SLOC<10	10≤D/P <100	100≤D/P < 1000	D/P ≥ 1000	
CPLX			See Table 24			
RUSE		none	Across projet	Across program	Across product line	Across multiple product lines
DOCU	Many life-cycle needs uncovered	Some life-cycle needs uncovered	Right-sized to life-cycle needs	Excessive for life-cycle needs	Very excessive for life-cycle needs	
TIME			≤ 50% of available excution time	70%	85%	95%
STOR			≤ 50% of available storage	70%	85%	95%
PVOL		Major change every 12 months; minor change every 1 month	Major: 6 months; minor: 2 weeks	Major: 2 months; minor: 1 week	Major: 2 weeks; minor: 2 days	
ACAP	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
PCAP	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
PCON	48%/year	24%/year	12%/year	6%/year	3%/year	
AEXP	≤ 2 months	6 months	1 year	3 years	6 years	
PEXP	≤ 2 months	6 months	1 year	3 years	6 years	
LTEX	≤ 2 months	6 months	1 year	3 years	6 years	
TOOL	Edit, code, debug	Simple, frontend, backend, CASE,	Basic life cycle tools, moderatel	Strong, mature life cycle tools, moderately	Strong, mature, proactive life cycle	

		little integration	y integrated	integrated	tools, well integrated with processes, methods, reuse	
SITE: Collocation	International	Multicity or multicompany	Multicity and multi company	Same city or metro. area	Same building or complex	Fully collocated
SITE: Communications	Some phone, mail	Individual phone, fax	Narrowband email	Wideband electronic communication	Wideband electronic communication occasional video conference	Interaction multimedia
SCED	75% of nominal	85%	100%	130%	160%	

13.7 Đếm số dòng mã lệnh trong COCOMO 2.0

Definition Checklist for Source Statements Counts

Definition name: Logical Source Statements Date:

(basic definition) Originator: COCOMO II

Measurement unit	Physical source lines	
	Logical source statements	■

Statement type	Definition ■	Data Array		Includes	Excludes
When a line or statement contains more than one type, classify it as the type with the highest precedence.					
1 Executable	Order of precedence →		1	■	
2 Nonexecutable					
3 Declarations			2	■	
4 Compiler directives			3	■	
5 Comments					
6 On their own lines			4		■
7 On lines with source code			5		■
8 Banners and non-blank spacers			6		■
9 Blank (empty) comments			7		■
10 Blank lines			8		■
11					
12					

How produced	Definition ■	Data array	Includes	Excludes
1 Programmed			■	
2 Generated with source code generators				■
3 Converted with automated translators			■	
4 Copied or reused without change			■	
5 Modified			■	
6 Removed				■
7				
8				

Origin	Definition ■	Data array	Includes	Excludes
1 New work: no prior existence			■	
2 Prior work: taken or adapted from				
3 A previous version, build, or release			■	
4 Commercial, off-the-shelf software (COTS), other than libraries				■
5 Government furnished software (GFS), other than reuse libraries				■
6 Another product				■
7 A vendor-supplied language support library (unmodified)				■
8 A vendor-supplied operating system or utility (unmodified)				■
9 A local or modified language support library or operating system				■
10 Other commercial library				■
11 A reuse library (software designed for reuse)			■	
12 Other software component or library			■	
13				
14				

***Hình 38:** Đếm số dòng mã lệnh.*

13.8 Tổng kết các công thức ước lượng trong COCOMO 2.0

- Early prototyping:

$$PM = \frac{(NOP \times \frac{(1-\%reuse)}{100})}{PROD}$$

- Early design:

$$E = 2.5 \times (KDSI)^{1.01+0.01\times\sum_{i=1}^{5} w_i} \times M + PM_m$$

với

$$M = PERS \times RCPX \times RUSE \times PDIF \times PREX \times FCIL \times SCED$$

và

$$PM_m = \frac{ASLOC \times \frac{AT}{100}}{ATPROD}$$

- Post-Architecture:

$$E = 2.5 \times (KDSI)^B \times M + PM_m, B = 1.01 + 0.01 \times \sum_{i=1}^{5} w_i$$

với

$$M = \prod_{i=1}^{17} f_i$$

- Thời gian thực hiện:

$$T = 3.0 \times PM^{(0.33+0.2\times(B-1.01))}$$

$$T = [3.0 \times (PM^{(0.33+0.2\times(B-1.01))})] \times \frac{SCED_{Percentage}}{100}$$

13.9 Sử dụng các hệ số nhân

Bảng 26: Hệ số nhân cho các thời kỳ Early Design và Post-Architecture.

Early design cost driver	Counterpart combined post-architecture cost driver
RCPX	RELY, DATA, CPLX, DOCU
RUSE	RUSE
PDIF	TIME, STOR, PVOL
PERS	ACAP, PCAP, PCON
PREX	AEXP, PEXP, LTEX
FCIL	TOOL, SITE
SCED	SCED

13.10 Ước lượng dựa trên phương trình

Mô hình ước lượng dựa trên cơ sở hàm đa biến là

$$E = [LOC \times \frac{B^{0.333}}{P}]^3 \times (\frac{1}{t^4})$$

với

E = nhân công theo đơn vị người-tháng hoặc người-năm

t = thời gian thực hiện phần mềm theo tháng hoặc năm

B = nhân tố về kỹ năng đặc biệt

P = tham số về năng suất (chẳng hạn như P=2000 cho phần mềm hệ thống nhúng thời gian thực, P=10000 cho phần mềm hệ thống và viễn thông, P=28000 cho ứng dụng hệ thống nghiệp vụ)

Putnam và Myers đề nghị hàm đơn giản hóa với $t_{min} = 8.14 \times (\frac{LOC}{P})0.43$ với t_{min}> 6 tháng (t_{min} được xác định theo tháng) và $E = 180 \times B \times t^3$ với E ≥ 20 người-tháng (E được xác định theo người-tháng).

Xét ví dụ trên phần mềm có 3 hệ thống con trong phần trước với P=12000 (khuyến nghị cho các phần mềm khoa học) ta được $t_{min} = 8.14 \times (\frac{33200}{12000})^{0.43} = 12.6$ tháng và $E = 180 \times 0.28 \times (1.05)^3 = 58$ người-tháng.

13.11 Ước lượng dựa trên các kỹ thuật khác

Ước lượng có thể được xác định trên cơ sở các phần mềm hướng đối tượng với các hệ số nhân (không giao diện: 2.0, giao diện văn bản: 2.25, giao diện: 2.5, giao diện phức tạp: 3.0), phát triển theo Agile hoặc theo hướng công nghệ web.

TÀI LIỆU THAM KHẢO

[1] Barry W. Boehm, *Software Engineering*, IEEE Computer Society - Wiley, 2007.

[2] Alphonse Carlier, *Le développement du logiciel*, Hermes, 1995.

[3] Alistair Cockburn, *Writing Effective Use-Cases*, Addison-Wesley, 2000.

[4] Scott E. Donaldson and Stanley G. Siegel, *Successful Software Development (2nd edition)*, Prentice Hall, 2000.

[5] Robert G. Glass, *Facts and Fallacies of Software Enginnering*, Addison Wesley, 2002.

[6] IEEE, *Guide to the Software Engineering Body of Knowledge - SWEBOK®*, IEEE Computer Society, 2004.

[7] The International Function Users Group, *Function Point Counting Practices Manual (Release 4.1)*, 1999.

[8] Stephen H. Kan, *Metrics and Models in Software Quality Engineering*, Addison-Wesley, 2002.

[9] Michael R. Lyu, *Handbook of Software Reliability Engineering*, IEEE Computer Society Press – McGraw-Hill, 1996.

[10] Timothy C. Lethbridge and Robert Laganière, *Object-Oriented Software Engineering: Practical Software Development Using UML and Java*, McGraw-Hill, 2002.

[11] Raymond J. Madachy, *Software Process Dynamics*, IEEE Press – Wiley, 2008.

[12] Robert E. Park, *Software Size Measurement: A Framework for Counting Source Statements*, Technical Report CMU/SEI-92-TR-020 ESC-TR-92-020, 1996.

[13] Roger S. Pressman, *Software Engineering: A Practitioner's Approach (6th edition)*, McGraw-Hill, 2005.

[14] Stephen R. Schach, *Object-Oriented and Classical Software Engineering (5th, 6th, 7th editions)*, McGraw-Hill, 2002, 2005, 2007.

[15] Ian Sommerville, *Software Engineering (6th, 8th editions)*, Addison-Wesley, 2001, 2006.

[16] Hans van Vliet, *Software Engineering: Principals and Practice (2nd edition)*, Wiley, 2000.

[17] http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html

[18] http://www.rspa.com/

CÂU HỎI HƯỚNG DẪN ÔN TẬP

1. Phân biệt kỹ thuật xác định kích thước phần mềm theo số dòng mã lệnh và theo số chức năng?
2. Phân biệt các cách thức đánh giá ở mỗi giai đoạn trong chu trình sống của phần mềm?
3. Ưu nhược điểm của phương pháp COCOMO?
4. Sự chuyển đổi/tương ứng giữa số dòng mã lệnh và chức năng nên dựa trên cơ sở nào?

ĐỊNH HƯỚNG THẢO LUẬN

1. Những khó khăn khi so sánh các mô hình ước lượng chi phí khác nhau?
2. Tác động của chính sách lên các mô hình ước lượng?
3. Tổng kết về các mô hình ước lượng theo chức năng và theo đối tượng?
4. Giả sử bạn được giao trách nhiệm phát triển một phần mềm được ước lượng khoảng 100 người-tháng. Bạn ước lượng thời gian thực hiện cho phần mềm này như thế nào? Nếu phải rút ngắn thời gian xuống còn 6 tháng thì có khả thi hay không?

BÀI TẬP THỰC HÀNH

1. Tìm, tải về, cài đặt và sử dụng các công cụ đếm số dòng mã lệnh thông dụng hiện nay (LocMetrics, ...).
2. Tìm, tải về, cài đặt và sử dụng các công cụ sử dụng kỹ thuật ước lượng COCOMO (COCOMO 81, Intermediate COCOMO, COCOMO 2.0, Agile COCOMO,...).
3. Cài đặt chương trình xác định kích thước theo kỹ thuật Halstead.
4. Cài đặt chương trình thể hiện nội dung đánh giá các giai đoạn trong chu trình sống của phần mềm.
5. Bạn là kỹ sư phần mềm, một năm trước đây nhà quản lý trong công ty thông báo với bạn rằng sản phẩm tiếp theo sẽ có 23 tập tin, 57 luồng và 66 tiến trình. Sử dụng độ đo FFP, hãy xác định kích thước của sản phẩm.
6. Một sản phẩm có 17 đầu vào đơn giản, 25 đầu vào trung bình và 9 đầu vào phức tạp. Ngoài ra sản phẩm còn có 27 đầu ra trung bình, 15 truy vấn đơn giản, 6 truy vấn phức tạp, 17 tập tin chính trung bình và 15 giao diện phức tạp.
   (i) Hãy xác định số lượng các điểm chức năng chưa hiệu chỉnh.

(ii) Giả sử tổng mức độ ảnh hưởng là 47, hãy xác định số lượng điểm chức năng.

(iii) Tính số lượng KDSI theo ngôn ngữ lập trình Java ?

7. Bạn được giao trách nhiệm phát triển một sản phẩm phần mềm dạng lớn có 76 KDSI trong đó các yếu tố ảnh hưởng đều bình thường ngoại trừ kích thước cơ sở dữ liệu được xếp loại rất lớn và việc sử dụng công cụ phần mềm là thấp. Sử dụng phương pháp COCOMO trung gian, hãy ước lượng các thông số cần thiết.

8. Bạn được giao trách nhiệm phát triển hai sản phẩm phần mềm dạng cơ bản có 58 KDSI. Cả hai sản phẩm đều có các thông số ảnh hưởng được xếp loại bình thường ngoại trừ sản phẩm P_1 có độ phức tạp cực kỳ cao và sản phẩm P_2 có độ phức tạp cực kỳ thấp. Để phát triển các sản phẩm bạn cần hai nhóm phát triển phần mềm. Nhóm A có khả năng phân tích, kinh nghiệm ứng dụng và khả năng lập trình được xếp loại rất cao. Nhóm A cũng có kinh nghiệm cao về máy ảo và ngôn ngữ lập trình. Nhóm B được xếp loại rất thấp trên cả năm thuộc tính.

(i) Hãy ước lượng các thông số cần thiết nếu nhóm A phát triển sản phẩm P_1 và nhóm B phát triển sản phẩm P_2.

(ii) Hãy ước lượng các thông số cần thiết nếu nhóm B phát triển sản phẩm P_1 và nhóm A phát triển sản phẩm P_2.

(iii) Hãy đưa ra các nhận xét của bạn về kết quả của (i) và (ii).

9. Bạn được giao trách nhiệm phát triển một sản phẩm phần mềm dạng trung bình có 134 KDSI trong đó các yếu tố ảnh hưởng đều được xếp loại bình thường ngoại trừ độ phức tạp được xếp loại cực kỳ cao, yêu cầu về sử dụng lại là rất cao, vấn đề liên tục của nhân sự là cao và việc sử dụng công cụ phần mềm là rất cao. Sử dụng phương pháp COCOMO 2.0:

(i) Giả sử rằng chi phí phải trả cho một nhân công (người-tháng) là 20 triệu đồng, hãy ước lượng các thông số cần thiết và chi phí của sản phẩm phần mềm.

(ii) Giả sử bạn quyết định điều chỉnh nhóm phát triển phần mềm khi bắt đầu dự án. Khi đó yêu cầu về lịch biểu phát triển là cực kỳ cao, ràng buộc về thời gian thực thi là rất cao và bạn thay thế nhóm bình thường bằng một nhóm có kinh nghiệm, khả năng và tính liên tục là rất cao nhưng chi phí phải trả cho một nhân công (người-tháng) lên đến 42 triệu đồng. Hãy ước lượng các thông số cần thiết, chi phí của sản phẩm phần mềm.

(iii) Hãy đưa ra các nhận xét đánh giá của bạn về (i) và (ii).

CHƯƠNG 5

XÁC ĐỊNH TRỊ GIÁ PHẦN MỀM THEO CÔNG VĂN 3364/BTTTT

1. XÁC ĐỊNH TRỊ GIÁ PHẦN MỀM

Trị giá phần mềm (G) được tính trên cơ sở điểm trường hợp sử dụng (use-case point) theo công thức

$$G = 1.4 \times E \times P \times H \times 1.1$$

trong đó E là giá trị nỗ lực; P là thời gian lao động để thực hiện được một điểm trường hợp sử dụng sau hiệu chỉnh; H là mức lương lao động bình quân; 1.4 là hệ số nỗ lực cho điều chỉnh, sửa lỗi; 1.1 là thuế VAT (10%).

Bảng 27: Xác định trị giá phần mềm.

TT	Hạng mục	Diễn giải	Giá trị	Ghi chú
I	***Tính điểm trường hợp sử dụng (use-case)***			
1	Điểm tác nhân (TAW)			
2	Điểm trường hợp sử dụng (TBF)			
3	Tính điểm UUCP	UUCP=TAW+TBF		
4	Hệ số phức tạp về kỹ thuật công nghệ (TCF)	TCF=0.6+(0.01×TFW)		
5	Hệ số phức tạp về môi trường (EF)	EF=1.4+(-0.03×EFW)		
6	Tính điểm AUCP	AUCP=UUCP×TCF×EF		
II	***Nội suy thời gian lao động (P)***	P=người/giờ/AUCP		
III	***Giá trị nỗ lực thực tế (E)***	E=10/6×AUCP		
IV	***Mức lương lao động bình quân (H)***	H=người/giờ		
V	***Định giá phần mềm nội bộ (G)***	G=1.4×E×P×H×1.1		

2. GIÁ TRỊ NỖ LỰC THỰC TẾ

Giá trị nỗ lực thực tế $E = \frac{10}{6} \times AUCP$ trong đó $\frac{10}{6}$ là hệ số điều chỉnh nỗ lực; AUCP là giá trị điểm trường hợp sử dụng sau hiệu chỉnh với $AUCP = UUCP \times TCF \times EF$ trong đó

- UUCP là giá trị điểm trường hợp sử dụng trước hiệu chỉnh.
- TCF là hệ số phức tạp kỹ thuật công nghệ.
- EF là hệ số phức tạp môi trường.

Giá trị điểm trường hợp sử dụng trước hiệu chỉnh UUCP được xác định bằng $UUCP = TAW + TBF$ với TAW là giá trị các điểm tác nhân và TBF là điểm các trường hợp sử dụng.

3. TÍNH ĐIỂM TÁC NHÂN

Điểm của từng loại tác nhân:

$$TAW = \text{Số tác nhân} \times \text{Trọng số}.$$

Bảng 28: Xác định điểm tác nhân.

TT	Loại tác nhân	Mô tả	Số tác nhân	Điểm của từng loại tác nhân	Ghi chú
1	Đơn giản	Thuộc loại giao diện của chương trình			
2	Trung bình	Giao diện tương tác hoặc phục vụ một giao thức hoạt động			
3	Phức tạp	Giao diện đồ họa			
	Cộng (1+2+3)	**TAW**			

Bảng 29: Xác định trọng số tác nhân.

TT	Loại tác nhân	Trọng số	Diễn giải
1	Đơn giản	1	Một máy tính với giao diện lập trình ứng dụng API
2	Trung bình	2	Hoặc là giao diện người máy qua "dòng lệnh" hoặc thông qua một giao thức nào đó nhưng không có lập trình qua API
3	Phức tạp	3	Giao diện người máy qua giao diện đồ họa

4. TÍNH ĐIỂM TRƯỜNG HỢP SỬ DỤNG

Điểm của từng loại trường hợp sử dụng:

$$TBF = \text{Số trường hợp sử dụng} \times \text{Trọng số} \times \text{Hệ số BMT}$$

Bảng 30: Xác định điểm trường hợp sử dụng.

TT	Loại trường hợp sử dụng	Số trường hợp sử dụng	Điểm của từng loại trường hợp sử dụng	Mô tả
1	***B***			Các yêu cầu phải thỏa mãn thì phần mềm mới được chấp nhận. Số lượng giao dịch ≤ 3 hoặc đường chỉ thị.
	Đơn giản			
	Trung bình			
	Phức tạp			
2	***M***			Các chức năng không phải là cốt lõi hay các chức năng phụ trợ hoặc theo yêu cầu của bên đặt hàng. Số lượng giao dịch trung bình từ 4 đến 7.
	Đơn giản			
	Trung bình			
	Phức tạp			
3	***T***			Các yêu cầu được tư vấn thêm hoặc đưa ra để bên đặt hàng lựa chọn thêm nếu muốn. Số lượng giao dịch lớn hơn 7.
	Đơn giản			
	Trung bình			
	Phức tạp			
	Cộng (1+2+3)			**TBF**

Bảng 31: Xác định trọng số BMT.

TT	Loại trường hợp sử dụng	Trọng số	Hệ số BMT
1	B		
	Đơn giản	5	1
	Trung bình	10	1
	Phức tạp	15	1
2	M		
	Đơn giản	5	1.2
	Trung bình	10	1.2
	Phức tạp	15	1.2
3	T		
	Đơn giản	5	1.5
	Trung bình	10	1.5
	Phức tạp	15	1.5

5. TÍNH HỆ SỐ PHỨC TẠP KỸ THUẬT – CÔNG NGHỆ

Hệ số phức tạp kỹ thuật công nghệ TCF được xác định bằng $TCF = 0.6 + (0.01 \times TFW)$ với 0.6, 0.01 là các trọng số đo chuẩn và TFW là hệ số kỹ thuật công nghệ.

Hệ số kỹ thuật công nghệ (TFW) được xác định như sau:

$$TFW = \sum_{i=1}^{13} Q_i^{xephang} \times \text{Trọng số}$$

với $Q_i^{xephang}$ là giá trị xếp hạng của 13 hệ số thành phần, biến thiên trong khoảng giá trị từ 0 đến 5 phản ánh tác động từ không quan trọng đến có vai trò tác động căn bản.

Bảng 32: Xác định hệ số kỹ thuật công nghệ TFW.

TT	Các hệ số	Trọng số	Giá trị xếp hạng	Kết quả	Ghi chú
I	**Hệ số kỹ thuật công nghệ (TFW)**				
1	Hệ thống phân tán	2			
2	Tính chất đáp ứng tức thời hoặc yêu cầu đảm bảo thông lượng	1			
3	Hiệu quả sử dụng trực tuyến	1			
4	Độ phức tạp của xử lý bên trong	1			
5	Mã nguồn phải tái sử dụng được	1			
6	Dễ cài đặt	0.5			
7	Dễ sử dụng	0.5			
8	Khả năng chuyển đổi	2			
9	Khả năng dễ thay đổi	1			
10	Sử dụng đồng thời	1			
11	Có các tính năng bảo mật đặc biệt	1			
12	Cung cấp truy nhập trực tiếp tới các phần mềm bên thứ ba	1			
13	Yêu cầu phương tiện đào tạo đặc biệt cho người sử dụng	1			
II	**Hệ số phức tạp về kỹ thuật công nghệ (TCF)**				

6. TÍNH HỆ SỐ PHỨC TẠP MÔI TRƯỜNG

Hệ số phức tạp môi trường EF được xác định bằng $EF = 1.4 + (-0.03 \times EFW)$ trong đó 1.4, 0.03 là trọng số đo chuẩn; EFW là hệ số tác động môi trường và nhóm làm việc.

7. DỰ KIẾN TRÌNH ĐỘ VÀ KINH NGHIỆM CẦN CÓ CỦA NHÂN CÔNG LAO ĐỘNG

Bảng 33: Dự kiến trình độ và kinh nghiệm.

TT	KỸ NĂNG	ĐIỂM ĐÁNH GIÁ
1	**Kỹ năng lập trình**	
	HTML	
	PHP/MySQL	
	Java	
	JavaScript	
	VB	
	VC++	
	C/C++	
	Microsoft.NET	
	Kylix	
	Perl	
	C#	
	Delphi	
	...	
2	**Kiến thức về phần mềm**	
	Flash	
	Illustrator	
	Photoshop	
	Firework	
	SQL Server	
	Oracle	
	IIS	
	Frontpage	
	MS Word	
	MS Excel	

	Open Office	
	MS Access	
	Visio	
	MS Project	
	Linux	
	Unix	
	Windows NT/2003/2008	
	Windows 2000/XP/Vista/7	
	LAN	
	WAN	
	Internet	
	Intranet	
	...	
3	**Hiểu biết về quy trình và kinh nghiệm thực tế**	
	Có áp dụng quy trình phát triển phần mềm theo mẫu RUP và có hiểu biết về RUP	
	Có kinh nghiệm về ứng dụng tương tự (application experiences)	
	Có kinh nghiệm về hướng đối tượng (object-oriented)	
	Có khả năng lãnh đạo nhóm	
	Có tính cách năng động	
4	**Loại khác** (ghi rõ loại)	
	...	

8. TÍNH HỆ SỐ TÁC ĐỘNG MÔI TRƯỜNG VÀ NHÓM LÀM VIỆC, HỆ SỐ PHỨC TẠP VỀ MÔI TRƯỜNG, XÁC ĐỊNH ĐỘ ỔN ĐỊNH KINH NGHIỆM VÀ NỘI SUY THỜI GIAN LAO ĐỘNG

Bảng 34: Xác định hệ số tác động.

TT	Các hệ số tác động môi trường	Giá trị xếp hạng	Kết quả	Độ ổn định kinh nghiệm
I	**Hệ số tác động môi trường và nhóm làm việc (EFW)**			
	Đánh giá cho từng thành viên			

1	Có áp dụng quy trình phát triển phần mềm theo mẫu RUP và có hiểu biết về RUP			
2	Có kinh nghiệm về ứng dụng tương tự			
3	Có kinh nghiệm về hướng đối tượng			
4	Có khả năng lãnh đạo nhóm			
5	Tính chất năng động			
	Đánh giá chung cho dự án			
6	Độ ổn định của các yêu cầu			
7	Có sử dụng các nhân viên làm việc bán thời gian (part-time)			
8	Dùng ngôn ngữ lập trình loại khó			
II	**Hệ số phức tạp về môi trường (EF)**			
III	**Độ ổn định kinh nghiệm (ES)**			
IV	**Nội suy thời gian lao động (P)**			

Điểm đánh giá tại mục I biến thiên trong khoảng từ 1 đến 5 thể hiện trình độ yếu (1), trung bình (3) và giỏi (5); chấp nhận giá trị lẻ đến hàng chục (một chữ số thập phân sau dấu phẩy). Giá trị ở mục II được xác định trên cơ sở giá trị xếp hạng tại mục I.

9. TÍNH HỆ SỐ TÁC ĐỘNG MÔI TRƯỜNG VÀ NHÓM LÀM VIỆC

$$EFW = \sum_{i=1}^{8} M_i^{xephang} \times \text{Trọng số}$$

với $M_i^{xephang}$ là giá trị xếp hạng của 8 hệ số thành phần, biến thiên trong khoảng giá trị từ 0 đến 5 phản ánh tác động từ không có kinh nghiệm đến trình độ chuyên gia.

Bảng 35: Xác định hệ số tác động môi trường và nhóm làm việc.

Thứ tự các hệ số tác động môi trường	Giá trị xếp hạng (từ 0 đến 5)	Trọng số
	Đánh giá cho từng thành viên	
1	0 = Không có kinh nghiệm 3 = Trung bình 5 = Trình độ chuyên gia	1.5
2	0 = Không có kinh nghiệm 3 = Trung bình 5 = Trình độ chuyên gia	0.5

3	0 = Không có kinh nghiệm 3 = Trung bình 5 = Trình độ chuyên gia	1
4	0 = Không có kinh nghiệm 3 = Trung bình 5 = Trình độ chuyên gia	0.5
5	0 = Không có kinh nghiệm 3 = Trung bình 5 = Trình độ chuyên gia	1
	Đánh giá chung cho nhóm làm việc	
6	0 = Rất bất định 5 = Không hay thay đổi	2
7	0 = Không có nhân viên làm việc bán thời gian 3 = Có nhân viên làm việc bán thời gian 5 = Tất cả đều làm bán thời gian	-1
8	0 = Ngôn ngữ lập trình dễ 3 = Trung bình 5 = Khó	-1

10. TÍNH ĐỘ ỔN ĐỊNH KINH NGHIỆM

Độ ổn định kinh nghiệm được xác định bằng tổ các giá trị nội suy từ kết quả tính được các hệ số tác động môi trường và nhóm làm việc

$$ES = \sum_{i=1}^{8} S_i^{noisuy}$$

trong đó S_i^{noisuy} là giá trị nội suy tương ứng của 8 hệ số thành phần.

Bảng 36: Xác định giá trị ổn định kinh nghiệm.

Kết quả	Giá trị nội suy
≤ 0	0.00
> 0	0.05
> 1	0.10
> 2	0.60
> 3	1.00

11. TÍNH THỜI GIAN LAO ĐỘNG

Thời gian lao động (P) được xác định trên cơ sở nội suy độ ổn định kinh nghiệm.

Bảng 37: Xác định giá trị thời gian lao động.

ES	Giá trị nội suy (P)
< 1	48
≥ 1	32
≥ 3	20

12. MỨC LƯƠNG LAO ĐỘNG BÌNH QUÂN

Giá trị mức lao động bình quân H được xác định căn cứ theo mặt bằng giá của thị trường lao động phổ biến (theo khu vực, mức tiền lương, phụ cấp,...)

$$H = g^{nc} \times (1 + f)$$

trong đó g^{nc} là mức đơn giá tiền lương giờ công trực tiếp bình quân tương ứng với cấp bậc lương hoặc năng lực; f là tổng các khoản phụ cấp lương, lương phụ, có tính chất ổn định với

$$f = f_1 + f_2 + f_3$$

- f_1 là tổng các khoản phụ cấp lương có tính chất ổn định (kể cả các khoản hỗ trợ lương);
- f_2 là lương phụ và một số chi phí có thể trả trực tiếp cho người lao động;
- f_3 là hệ số điều chỉnh cho phù hợp với thị trường nhân công khu vực và đặc thù của môi trường lao động.

TÀI LIỆU THAM KHẢO

[1] Alistair Cockburn, *Writing Effective Use-Cases*, Addison-Wesley, 2000.

[2] Scott E. Donaldson and Stanley G. Siegel, *Successful Software Development (2nd edition)*, Prentice Hall, 2000.

[3] IEEE, *Guide to the Software Engineering Body of Knowledge - SWEBOK®*, IEEE Computer Society, 2004.

[4] Stephen H. Kan, *Metrics and Models in Software Quality Engineering*, Addison-Wesley, 2002.

[5] Per Kroll and Philippe Kruchten, *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*, Addison Wesley, 2003.

[6] Philippe Kruchten, *The Rational Unified Process: An Introduction (2^{nd}, 3^{rd} editions)*, Addison Wesley, 2000, 2003.

[7] Timothy C. Lethbridge and Robert Laganière, *Object-Oriented Software Engineering: Practical Software Development Using UML and Java*, McGraw-Hill, 2002.

[8] Nguyễn Minh Hồng, "Hướng dẫn xác định trị giá phần mềm", *Công văn số 3364/BTTTT-UDCNTT ngày 17 tháng 10 năm 2008*, Bộ Thông tin & Truyền thông, 2008.

[9] Nguyễn Minh Hồng, "Sửa đổi giá trị trọng số BMT", *Công văn số 2496/BTTTT-UDCNTT ngày 04 tháng 08 năm 2010*, Bộ Thông tin & Truyền thông, 2010.

[10] Rational Software White Paper, *Rational Unified Process® for Systems Engineering RUP® SE1.1*, Rational Software Corporation, TP 165A, 2002.

[11] Rational Software White Paper, *Rational Unified Process: Best Practices for Software Development Teams*, IBM Corporation, TP026B, Rev 11/01, 2001.

[12] Robert E. Park, *Software Size Measurement: A Framework for Counting Source Statements*, Technical Report CMU/SEI-92-TR-020 ESC-TR-92-020, 1996.

[13] Roger S. Pressman, *Software Engineering: A Practitioner's Approach (6^{th} edition)*, McGraw-Hill, 2005.

[14] Stephen R. Schach, *Object-Oriented and Classical Software Engineering (5^{th},6^{th},7^{th} editions)*, McGraw-Hill, 2002, 2005, 2007.

[15] John Smith, *The Estimation of Effort Based on Use Cases*, IBM, 2003.

[16] Ian Sommerville, *Software Engineering (6^{th},8^{th} editions)*, Addison-Wesley, 2001, 2006.

[17] Hans van Vliet, *Software Engineering: Principals and Practice (2^{nd} edition)*, Wiley, 2000.

[18] http://mic.gov.vn/

[19] http://www.ibm.com/developerworks/rational/library/edge/09/mar09/collaris_dekker/

CÂU HỎI HƯỚNG DẪN ÔN TẬP

1. Phân biệt các cách tính hệ số tác động?
2. Cách tính mức lương lao động bình quân?
3. Phân tích cách xác định trị giá phần mềm dựa trên cơ sở các trường hợp sử dụng?

ĐỊNH HƯỚNG THẢO LUẬN

1. Phân biệt tiếp cận dựa trên các trường hợp sử dụng và các tiếp cận khác (dòng mã lệnh, chức năng,...) trong xác định trị giá phần mềm?
2. Ảnh hưởng của ngôn ngữ lập trình và môi trường làm việc lên trị giá phần mềm?

BÀI TẬP THỰC HÀNH

1. Thực hiện xác định trị gía phần mềm cho một phần mềm cụ thể.
2. Cài đặt việc tính toán trị giá phần mềm theo các trường hợp sử dụng.

PHỤ LỤC A

CÁC PHẦN TỬ TRONG RUP

1. CÁC VAI TRÒ

1.1 Các vai trò phân tích (analyst roles/workers)

Business-Process Analyst. Vai trò này có trách nhiệm định nghĩa kiến trúc nghiệp vụ, các trường hợp sử dụng và tác nhân và cách thức tương tác giữa trường hợp sử dụng và tác nhân. Vai trò này cũng có trách nhiệm trong việc phác thảo và phân định ranh giới việc mô hình hóa.

Business Designer. Vai trò này chi tiết hóa đặc tả của một phần tổ chức cũng như xác định dòng công việc các trường hợp sử dụng nghiệp vụ với các vai trò và thực thể nghiệp vụ. Việc phân phối các hành vi cho những vai trò và thực thể nghiệp vụ - định nghĩa các trách nhiệm, xử lý, thuộc tính và mối quan hệ tương ứng.

System Analyst. Vai trò này dẫn dắt và phối hợp các yêu cầu tìm thấy và mô hình hóa trường hợp sử dụng bằng cách phác thảo các chức năng của hệ thống và giới hạn của hệ thống.

Requirements Specifier. Vai trò này xác định chi tiết của một hay nhiều khía cạnh của các yêu cầu, đảm bảo tính toàn vẹn của các tác vụ.

1.2 Các vai trò phát triển (developer roles)

Software Architect. Vai trò kiến trúc sư phần mềm có trách nhiệm tổng thể để điều khiển các quyết định kỹ thuật chủ yếu, được thể hiện như các kiến trúc phần mềm. Thường bao gồm việc xác định và lập tài liệu về các khía cạnh kiến trúc quan trọng của hệ thống, bao gồm cả các yêu cầu, thiết kế, thực hiện và triển khai "điểm" của hệ thống. Đây cũng là vai trò có trách nhiệm cung cấp lý do cho những quyết định, cân bằng các mối quan tâm của các bên liên quan khác nhau, điều khiển các rủi ro kỹ thuật, và đảm bảo rằng các quyết định được truyền đạt, xác nhận, và tôn trọng một cách hiệu quả.

Designer. Vai trò thiết kế chịu trách nhiệm thiết kế một phần của hệ thống, trong những hạn chế của các yêu cầu, kiến trúc, và quá trình phát triển cho dự án. Vai trò này xác định và định nghĩa trách nhiệm, hoạt động, thuộc tính, và mối quan hệ của các yếu tố thiết kế. Đồng thời vai trò này cũng đảm bảo rằng thiết kế phù hợp với kiến trúc phần mềm và chi tiết đến mức mà có thể tiến hành cài đặt.

User-Interface Designer. Vai trò thiết kế giao diện người sử dụng thực hiện việc thiết kế giao diện người sử dụng. Vai trò này cũng tham gia vào việc thu thập yêu cầu sử dụng và thiết kế các ứng viên mẫu giao diện phù hợp với yêu cầu đặt ra.

Capsule Designer. Vai trò thiết kế đóng gói chịu trách nhiệm cho việc thiết kế đảm bảo hệ thống có thể đáp ứng với sự kiện một cách kịp thời và phù hợp với các yêu cầu tương tranh.

Database Designer. Vai trò thiết kế cơ sở dữ liệu định nghĩa các bảng, chỉ mục, khung nhìn, ràng buộc, kích hoạt (triggers), thủ tục lưu trữ sẵn, không gian bảng hoặc các thông số lưu trữ và các cơ sở dữ liệu cụ thể cần thiết để lưu trữ, truy lục, và xóa các đối tượng quá hạn.

Implementer. Vai trò cài đặt chịu trách nhiệm phát triển và thử nghiệm các thành phần, theo quy định về các tiêu chuẩn của dự án đã được thông qua, nhằm tích hợp vào các hệ thống con lớn hơn. Vai trò này cũng trách nhiệm phát triển và thử nghiệm các thành phần thử nghiệm và các hệ thống con tương ứng.

Integrator. Vai trò tích hợp có trách nhiệm lập kế hoạch tích hợp và thực hiện việc tích hợp các yếu tố cai đặt để sản xuất phần mềm.

1.3 Các vai trò quản lý (manager roles)

Project Manager. Vai trò quản lý dự án phân bổ nguồn lực, dạng ưu tiên, phối hợp tương tác với khách hàng và người sử dụng và thường cố gắng để giữ cho nhóm thực hiện dự án tập trung vào các mục tiêu đúng. Vai trò này cũng thiết lập một tập các thông lệ để đảm bảo tính toàn vẹn và chất lượng của các tác vụ của dự án.

Change Control Manager. Vai trò quản lý thay đổi giám sát quy trình kiểm soát thay đổi. Vai trò này thường được thực hiện bởi một ban kiểm soát thay đổi (CCB). Ban này bao gồm các đại diện từ tất cả các bên quan tâm, bao gồm khách hàng, nhà phát triển, và người sử dụng. Trong một dự án nhỏ, một người duy nhất, chẳng hạn như quản lý dự án hoặc kiến trúc sư phần mềm, có thể đóng vai trò này.

Configuration Manager. Vai trò quản lý cấu hình chịu trách nhiệm cung cấp cơ sở hạ tầng và môi trường cho nhóm phát triển sản phẩm. Chức năng của vai trò này là hỗ trợ các hoạt động phát triển sản phẩm để người phát triển và người tích hợp có không gian làm việc thích hợp để thực hiện và kiểm tra công việc của họ và tất cả các tác vụ có sẵn theo yêu cầu. Vai trò này đảm bảo môi trường thuận lợi, tạo điều kiện xem xét lại, thay đổi và hoạt động theo dõi lỗi của sản phẩm. Quản lý cấu hình cũng chịu trách nhiệm viết kế hoạch quản lý cấu hình và báo cáo thống kê tiến độ thay đổi dựa trên yêu cầu.

Test Manager. Quản lý kiểm thử có trách nhiệm tổng thể cho sự thành công của nỗ lực thử nghiệm. Vai trò này liên quan đến chất lượng và thử nghiệm đúng nguyên tắc, lập kế hoạch quản lý tài nguyên và giải quyết các vấn đề cản trở các nỗ lực thử nghiệm.

Deployment Manager. Vai trò quản lý triển khai có trách nhiệm lập kế hoạch chuyển giao của sản phẩm đến cộng đồng người sử dụng, đảm bảo các kế hoạch này được ban hành phù hợp, quản lý các vấn đề và theo dõi tiến độ.

Process Engineer. Vai trò kỹ sư tiến trình chịu trách nhiệm cho bản thân quá trình phát triển phần mềm. Điều này bao gồm việc cấu hình quá trình trước khi dự án khởi động và liên tục cải tiến quy trình trong nỗ lực phát triển.

Management Reviewer. Vai trò quản lý xem xét lại có trách nhiệm thẩm định kế hoạch phần mềm và đánh giá các tác vụ của phần mềm tại các điểm xem xét lại quan trọng trong chu trình sống của phần mềm.

1.4 Các vai trò kiểm thử (tester roles)

Test Analyst. Vai trò phân tích kiểm thử có trách nhiệm xác định và định nghĩa các kiểm thử cần thiết, theo dõi chi tiết tiến độ kiểm thử và kết quả kiểm thử trong mỗi chu kỳ kiểm thử, và đánh giá chất lượng kinh nghiệm tổng thể như là kết quả của hoạt động kiểm thử. Vai trò này thường chịu trách nhiệm đại diện phù hợp cho nhu cầu của các bên liên quan không có đại diện trực tiếp hoặc thường xuyên trong dự án.

Test Designer. Vai trò thiết kế kiểm thử có trách nhiệm lập kế hoạch, thiết kế, cài đặt và đánh giá các kiểm thử, bao gồm việc hình thành ra kế hoạch kiểm thử và mô hình kiểm thử, cài đặt các thủ tục kiểm thử và đánh giá tầm bao phủ của kiểm thử, kết quả kiểm thử và hiệu quả.

Tester. Vai trò kiểm thử viên có trách nhiệm cho các hoạt động cốt lõi của những nỗ lực kiểm thử, trong đó có việc tiến hành các kiểm thử cần thiết và khai thác các kết quả kiểm thử này.

1.5 Các vai trò sản xuất và hỗ trợ (production and support roles)

System Administrator. Vai trò quản trị hệ thống duy trì môi trường phát triển cả phần cứng và phần mềm, và chịu trách nhiệm quản trị hệ thống, sao lưu, ...

Technical Writer. Vai trò người viết kỹ thuật tạo ra các nội dung hỗ trợ người dùng cuối, chẳng hạn như hướng dẫn sử dụng, tài liệu trợ giúp, ghi chú phát hành, ...

Graphic Artist. Vai trò đồ họa mỹ thuật sáng tạo ra tác phẩm nghệ thuật đồ họa sử dụng trong một phần của bao bì sản phẩm và tài liệu.

Tool Specialist. Vai trò chuyên gia công cụ chịu trách nhiệm về các công cụ hỗ trợ của dự án. Điều này bao gồm sự đánh giá sự cần thiết nhằm hỗ trợ công cụ và lựa chọn mua lại các công cụ.

Course Developer. Vai trò phát triển khóa học phát triển các nội dung đào tạo để hướng dẫn người sử dụng cách thức sử dụng các sản phẩm. Vai trò này tạo ra các

slides trình bày, ghi chú của người học, ví dụ, hướng dẫn, và tăng cường sự hiểu biết về sản phẩm.

1.6 Các vai trò khác (additional roles)

Reviewer. Vai trò người xem xét lại là một vai trò chung chung, có trách nhiệm cung cấp thông tin phản hồi kịp thời để thành viên của nhóm dự án trên các tác vụ do mình tạo ra (xem thêm vai trò quản lý xem xét lại và vai trò kỹ thuật xem xét lại).

Review Coordinator. Vai trò điều phối viên xem xét lại có trách nhiệm tạo điều kiện xem xét lại chính thức và thanh tra, đảm bảo rằng quá trình này luôn sẵn sàng khi có yêu cầu và được thực hiện với một tiêu chuẩn thỏa đáng.

Any Role. Đây là một vai trò chung chung được sử dụng để thực hiện các hoạt động mà bất kỳ thành viên nào dự án cũng có thể thực hiện.

Stakeholder. Vai trò bên liên quan là bất kỳ người nào chịu tác động bởi kết quả của dự án. Đây là một vai trò chung chung, bao phủ và tùy thuộc vào bối cảnh dự án, khách hàng, người sử dụng, người mua, người quản lý sản phẩm, ...

2. CÁC TÁC VỤ (artifacts)

Target-Organization Assessment	Business-Process Analyst
Business Vision	Business-Process Analyst
Business Glossary	Business-Process Analyst
Business Rules	Business-Process Analyst
Supplementary Business Specification	Business-Process Analyst
Business Use-Case Model	Business-Process Analyst
- Business Use Case	Business Designer
- Business Actor	Business Designer
- Business Goal	Business Designer
Business Analysis Model	Business-Process Analyst
- Business System	Business Designer
- Business Rule	Business Designer
- Business Event	Business Designer
- Business Worker	Business Designer
- Business Use-Case realization	Business Designer
- Business Entity	Business Designer
Business Architecture Document	Business-Process Analyst

Hình 39: *Các tác vụ mô hình hóa nghiệp vụ.*

Requirements Management Plan	System Analyst
Software Requirement	Requirements Specifier
Stakeholder Requests	System Analyst
Glossary	System Analyst
Vision	System Analyst
Requirements Attributes	System Analyst
Supplementary Specification	System Analyst
Software Requirements Specification	Requirements Specifier
Use-Case Model	System Analyst
- Use-Case Package	Requirements Specifier
- Use Case	Requirements Specifier
- Actor	Requirements Specifier
Storyboard	User-Interface Designer

Hình 40: *Các tác vụ yêu cầu.*

Reference Architecture	Software Architect
Architectural Proof-of-Concept	Software Architect
Software Architecture Document	Software Architect
Use-Case Realization	Designer
Analysis Model	Software Architect
- Analysis Class	Designer
Design Model	Software Architect
- Design Subsystem	Designer
- Design Package	Designer
- Design Class	Designer
- Interface	Designer
- Use-Case Realization	Designer
- Capsule	Designer
- Protocol	Software Architect
- Signal	Software Architect
- Event	Software Architect
- Testability Class	Designer
- Test Design	Test Designer
Deployment Model	Software Architect
Data Model	Database Designer
User-Interface Prototype	User-Interface Designer
Navigation Map	User-Interface Designer

Hình 41: *Các tác vụ phân tích và thiết kế.*

Build	Integrator
Implementation Model	Software Architect
- Implementation Subsystem	Implementer
- Implementation Element	Implementer
- Test Stub	Implementer
- Testability Element	Implementer
Integration Build Plan	Integrator
Developer Test	Implementer

Hình 42: Các tác vụ cài đặt .

Test Plan	Test Manager
Test Evaluation Summary	Test Manager
Test Strategy	Test Designer
Test Idea List	Test Analyst
Test Case	Test Analyst
Test Data	Test Analyst
Test Results	Test Analyst
Workload Analysis Model	Test Analyst
Test Interface Specification	Test Designer
Test Script	Test Designer
Test Suite	Test Designer
Test Environment Configuration	Test Designer
Test Automation Architecture	Test Designer
Test Log	Tester

Hình 43: Các tác vụ kiểm thử.

Deployment Plan	Deployment Manager
Product	Deployment Manager
- Bill of Materials	Deployment Manager
- Deployment Unit	Configuration Manager
- Product Artwork	Graphic Artist
- Installation artifacts	Implementer
End-User Support Material	Technical Writer
- Release Notes	Development Manager
- Training material	Course Developer

Hình 44: Các tác vụ triển khai.

Configuration Management Plan	Configuration Manager
Project Repository	Configuration Manager
Workspace	Any Role
Configuration Audit Findings	Configuration Manager
Change Request	Change Control Manager

Hình 45: *Các tác vụ quản lý cấu hình và thay đổi cấu hình.*

Deployment Plan	Deployment Manager
Business Case	Project Manager
Software Development	Plan Project Manager
- Quality Assurance Plan	Project Manager
- Problem Resolution Plan	Project Manager
- Risk Management Plan	Project Manager
- Product Acceptance Plan	Project Manager
- Measurement Plan	Project Manager
Risk List	Project Manager
Issues List	Project Manager
Iteration Plan	Project Manager
Iteration Assessment	Project Manager
Status Assessment	Project Manager
Work Order	Project Manager
Project Measurements	Project Manager
Review Record	Reviewer

Hình 46: *Các tác vụ quản lý dự án.*

Development Organization Assessment	Process Engineer
Development Process	Process Engineer
- Development Case	Process Engineer
- Project-Specific Templates	Process Engineer
- Project Specific Guidelines	Process Engineer
Development Infrastructure	System Administrator
Tools	Tool Specialist

Hình 47: *Các tác vụ môi trường.*

PHỤ LỤC B

KẾ HOẠCH VỀ PHẦN MỀM

1. Quản lý trách nhiệm
2. Hệ thống chất lượng
3. Xem xét hợp đồng
4. Kiểm soát thiết kế
5. Kiểm soát tài liệu và dữ liệu
6. Mua sắm
7. Kiểm soát khách hàng cung cấp sản phẩm
8. Xác định và truy xuất nguồn gốc sản phẩm
9. Kiểm soát quá trình
10. Thanh tra và kiểm thử
11. Kiểm soát thanh tra, đo và kiểm thử trang bị
12. Thanh tra và kiểm thử trạng thái
13. Kiểm soát sản phẩm không phù hợp
14. Hành động khắc phục và phòng ngừa
15. Xử lý, lưu trữ, bảo quản, phân phối
16. Kiểm soát hồ sơ chất lượng
17. Kiểm toán chất lượng nội bộ
18. Huấn luyện
19. Phục vụ
20. Kỹ thuật thống kê

Hình 48: *ISO 9001: Các hệ thống chất lượng.*

1. Mục tiêu
2. Tài liệu tham khảo
3. Quản lý
4. Tài liệu
5. Tiêu chuẩn, thông lệ, thỏa thuận và đánh giá
6. Xem xét và kiểm toán
7. Kiểm thử
8. Báo cáo vấn đề và biện pháp khắc phục
9. Công cụ, kỹ thuật và phương pháp
10. Kiểm soát mã lệnh
11. Kiểm soát truyền thông

12. Kiểm soát cung cấp
13. Tập hợp mẫu tin, bảo trì và duy trì
14. Huấn luyện
15. Quản lý rủi ro

Hình 49: *Kế hoạch đảm bảo chất lượng phần mềm [IEEE Standard 730].*

1. Giới thiệu
    - 1.1 Mục tiêu
    - 1.2 Phạm vi
    - 1.3 Định nghĩa, tên tắt và viết tắt
    - 1.4 Tham khảo
    - 1.5 Tổng quan
2. Mô tả tổng quan
    - 2.1 Quan điểm của sản phẩm
    - 2.2 Chức năng của sản phẩm
    - 2.3 Đặc điểm của người sử dụng
    - 2.4 Ràng buộc
    - 2.5 Giả định và phụ thuộc
    - 2.6 Tập yêu cầu nhỏ hơn
3. Yêu cầu cụ thể
    - 3.1 Yêu cầu giao diện bên ngoài
        - 3.1.1 Giao diện người dùng
        - 3.1.2 Giao diện phần cứng
        - 3.1.3 Giao diện phần mềm
        - 3.1.4 Giao diện liên lạc
    - 3.2 Yêu cầu chức năng
        - 3.2.1 Yêu cầu chức năng 1
            - 3.2.1.1 Giới thiệu
            - 3.2.1.2 Đầu vào
            - 3.2.1.3 Xử lý
            - 3.2.1.4 Đầu ra
    - 3.3 Yêu cầu hiệu quả
    - 3.4 Ràng buộc thiết kế
        - 3.4.1 Tuân thủ tiêu chuẩn
        - 3.4.2 Giới hạn phần cứng
    - 3.5 Thuộc tính hệ thống phần mềm
        - 3.5.1 Sẵn sàng

3.5.2 An ninh
3.5.3 Bảo trì
3.6 Các yêu cầu khác

Hình 50: *Đặc tả yêu cầu phần mềm [IEEE Standard 830].*

1. Mục tiêu
2. Các tài liệu tham khảo
3. Các định nghĩa
4. Tổng quan về thẩm tra và xác nhận
    4.1 sTổ chức
    4.2 Kế hoạch tổng thể
    4.3 Tổng kết tài nguyên
    4.4 Các trách nhiệm
    4.5 Công cụ, kỹ thuật và phương pháp
5. Thẩm tra và xác nhận chu trình sống
    5.1Quản lý V&V
    5.2 Giai đoạn yêu cầu V&V
    5.3 Giai đoạn thiết kế V&V
    5.4 Giai đoạn cài đặt V&V
    5.5 Giai đoạn kiểm thử V&V
    5.6 Giai đoạn cài đặt và kiểm tra V&V
    5.7 Vận hành và bảo trì V&V
6. Báo cáo thẩm tra và xác nhận phần mềm
7. Các thủ tục quản lý thẩm tra và xác nhận
    7.1 Báo cáo và giải quyết bất thường
    7.2 Chính sách về lặp lại công việc
    7.3 Độ lệch chính sách
    7.4 Thủ tục kiểm soát
    7.5 Tiêu chuẩn, thông lệ và quy ước

Hình 51: *Kế hoạch thẩm tra và hợp lệ hóa phần mềm [IEEE Standard 1012].*

1. Giới thiệu
    - 1.1 Mục tiêu
    - 1.2 Phạm vi
    - 1.3 Định nghĩa và viết tắt
    - 1.4 Tham khảo
2. Quản lý
    - 2.1 Tổ chức
    - 2.2 Các trách nhiệm
    - 2.3 Chính sách áp dụng, chỉ thị và thủ tục
3. Hoạt động
    - 3.1 Xác định cấu hình
    - 3.2 Kiểm soát cấu hình
    - 3.3 Kết toán tình trạng cấu hình
    - 3.4 Kiểm toán và xem xét lại cấu hình
    - 3.5 Kiểm soát giao diện
    - 3.6 Kiểm soát nhà thầu phụ/nhà cung cấp
4. Lịch biểu
5. Tài nguyên
6. Kế hoạch bảo trì

Hình 52: *Kế hoạch quản lý cấu hình [IEEE Standard 90b].*

1. Giới thiệu
    - 1.1 Khái quát về dự án
    - 1.2 Phân phối sản phẩm
    - 1.3 Phát triển của SPMP
    - 1.4 Tài liệu tham khảo
    - 1.5 Các định nghĩa và từ viết tắt
2. Tổ chức dự án
    - 2.1 Mô hình xử lý
    - 2.2 Cấu trúc tổ chức
    - 2.3 Tổ chức các ranh giới và giao diện
    - 2.4 Trách nhiệm trong dự án
3. Quy trình quản lý
    - 3.1 Quản lý các mục đích và các quyền ưu tiên
    - 3.2 Sự đảm đương, phụ thuộc và ràng buộc
    - 3.3 Quản lý rủi ro
    - 3.4 Các cơ chế giám sát và điều khiển
    - 3.5 Kế hoạch nhân sự
4. Quy trình kỹ thuật
    - 4.1 Các phương thức, công cụ và kỹ thuật
    - 4.2 Tài liệu phần mềm

4.3 Các chức năng hỗ trợ dự án

5. Các gói công việc, kế hoạch và ngân sách
   - 5.1 Các gói công việc
   - 5.2 Các phụ thuộc
   - 5.3 Yêu cầu tài nguyên
   - 5.4 Ngân sách và việc phân bổ tài nguyên
   - 5.5 Kế hoạch làm việc
6. Các thành phần bổ sung
   - 6.1 Cần thiết trong một số dự án
   - 6.2 Kế hoạch quản lý các thầu phụ, kế hoạch an ninh, kế hoạch đào tạo, kế hoạch mua phần cứng, kế hoạch cài đặt và kế hoạch bảo trì

Hình 53: *Kế hoạch quản lý dự án phần mềm [IEEE Standard 1058].*

PHỤ LỤC C

TÍNH TRỊ GIÁ PHẦN MỀM "Website trên Internet"

Các yêu cầu chức năng:

TT	Mô tả yêu cầu	Phân loại	Mức độ
1	Người quản trị hệ thống có thể thêm bài viết	Dữ liệu đầu vào	Đơn giản
2	Người quản trị hệ thống có thể sửa chi tiết bài viết	Dữ liệu đầu vào	Đơn giản
3	Người quản trị hệ thống có thể xóa bài viết	Dữ liệu đầu vào	Đơn giản
4	Người quản trị hệ thống có thể thêm tin mới	Dữ liệu đầu vào	Đơn giản
5	Người quản trị hệ thống có thể sửa tin tức	Dữ liệu đầu vào	Đơn giản
6	Người quản trị hệ thống có thể xóa tin tức	Dữ liệu đầu vào	Đơn giản
7	Khi người sử dụng thực hiện thao tác xóa hay chỉnh sửa, hệ thống hiện thông báo yêu cầu xác nhận cho tác vụ này	Dữ liệu đầu ra	Đơn giản
8	Người quản trị hệ thống có thể cấu hình giao diện giao tiếp với người sử dụng của hệ thống: thay đổi bố trí giao diện	Dữ liệu đầu vào	Trung bình
9	Người quản trị hệ thống có thể cấu hình giao diện giao tiếp với người sử dụng của hệ thống: thay đổi cách hiển thị các chuyên mục	Dữ liệu đầu vào	Trung bình
10	Người quản trị hệ thống có thể cấu hình giao diện giao tiếp với người sử dụng của hệ thống: cách hiển thị các quảng cáo	Dữ liệu đầu vào	Trung bình
11	Người quản trị hệ thống có thể cấu hình giao diện giao tiếp với người sử dụng của hệ thống: thay đổi cách thức hiển thị tin tức	Dữ liệu đầu vào	Trung bình
12	Người quản trị hệ thống có thể cấu hình giao diện giao tiếp với người sử dụng của hệ thống: thay đổi cách thức hiển thị bài viết	Dữ liệu đầu vào	Trung bình
13	Người quản trị hệ thống có thể thay đổi chuyên mục	Dữ liệu đầu vào	Trung bình
14	Người quản trị hệ thống có thể xóa chuyên mục	Dữ liệu đầu vào	Đơn giản

15	Người quản trị hệ thống có thể tạo mới chuyên mục	Dữ liệu đầu vào	Đơn giản
16	Người quản trị hệ thống có thể liệt kê các chuyên mục	Yêu cầu truy vấn	Trung bình
17	Người quản trị hệ thống có thể thêm mới chủ đề thông tin	Dữ liệu đầu vào	Đơn giản
18	Người quản trị hệ thống có thể thay đổi chủ đề thông tin	Dữ liệu đầu vào	Trung bình
19	Người quản trị hệ thống có thể xóa chủ đề thông tin	Dữ liệu đầu vào	Đơn giản
20	Người quản trị hệ thống có thể liệt kê các chủ đề thông tin	Yêu cầu truy vấn	Trung bình
21	Người quản trị hệ thống có thể thêm mới cuộc thăm dò ý kiến	Dữ liệu đầu vào	Đơn giản
22	Người quản trị hệ thống có thể thay đổi nội dung cuộc thăm dò ý kiến	Dữ liệu đầu vào	Phức tạp
23	Người quản trị hệ thống có thể tính toán kết quả thăm dò ý kiến	Yêu cầu truy vấn	Phức tạp
24	Người quản trị hệ thống có thể cấu hình thư viện hình ảnh, dữ liệu của hệ thống (thêm, bớt, chỉnh sửa)	Dữ liệu đầu vào	Trung bình
25	Người quản trị hệ thống cấu hình website thông qua một bảng điều khiển tích hợp	Dữ liệu đầu vào	Phức tạp
26	Người quản trị hệ thống có thể tạo lập các quyền sử dụng của người sử dụng	Dữ liệu đầu vào	Phức tạp
27	Người quản trị hệ thống có thể sửa đổi các quyền sử dụng của người sử dụng	Dữ liệu đầu vào	Phức tạp
28	Người quản trị hệ thống có thể hủy bỏ các quyền sử dụng của người sử dụng	Dữ liệu đầu vào	Đơn giản
29	Người quản trị hệ thống có thể nhóm các quyền riêng lẻ lại thành nhóm quyền sử dụng cho các nhóm người sử dụng	Dữ liệu đầu vào	Trung bình
30	Người quản trị hệ thống có thể xóa người sử dụng	Dữ liệu đầu vào	Đơn giản
31	Người quản trị hệ thống có thể quản lý các chức năng trên forum	Dữ liệu đầu vào	Phức tạp
32	Người quản trị hệ thống có thể quản lý các tiện ích trên trang web (tải về, đếm, khảo sát,...)	Dữ liệu đầu vào	Phức tạp
33	Người quản trị hệ thống có thể nhận thông tin phản hồi từ người truy cập	Dữ liệu đầu ra	Đơn giản
34	Người quản trị hệ thống có thể đăng thông tin phản hồi từ người truy cập trên website	Dữ liệu đầu vào	Đơn giản
35	Người quản trị hệ thống có thể trả lời cho người truy cập	Dữ liệu đầu vào	Đơn giản
36	Người quản trị hệ thống có thể sao lưu dữ liệu của website phục vụ khôi phục hoạt động khi xảy ra sự cố	Cơ sở dữ liệu	Trung bình
37	Người sử dụng nhập tên người dùng duy nhất và mật khẩu để đăng nhập hệ thống	Dữ liệu đầu vào	Đơn giản
38	Khách có thể đăng ký với hệ thống để trở thành thành viên	Dữ liệu đầu vào	Đơn giản
39	Sau khi đăng ký thành công khách phải kích hoạt thư điện tử thì tài khoản người dùng mới có hiệu lực	Dữ liệu đầu vào	Đơn giản

40	Khách có thể xem tin tức	Yêu cầu truy vấn	Trung bình
41	Khách có thể gửi thông tin phản hồi về tin tức qua thư điện tử	Dữ liệu đầu vào	Đơn giản
42	Khách có thể xem bài viết trên diễn đàn	Yêu cầu truy vấn	Trung bình
43	Khách có thể tải về dữ liệu, công cụ cần thiết	Dữ liệu tra cứu	Đơn giản
44	Khách có thể tải về thư viện ảnh	Dữ liệu tra cứu	Đơn giản
45	Khách có thể đăng ký các chuyên mục tin mà quan tâm để nhận được bản tin do hệ thống thông báo vào thời điểm định trước	Dữ liệu đầu vào	Trung bình
46	Khách có thể đăng ký các chuyên mục tin mà mình được phép của mục tin tức thông qua các thông tin lựa chọn như: tiêu đề, từ khóa, loại tin, tác giả, nội dung	Yêu cầu truy vấn	Phức tạp
47	Khách có thể xem những bài viết trên diễn đàn mà mình được quyền truy cập thông qua các thông tin lựa chọn như: tiêu đề, từ khóa, loại tin, tác giả, nội dung	Yêu cầu truy vấn	Phức tạp
48	Các tùy chọn dùng để tìm kiếm có thể kết hợp với nhau theo nhiều cách để tìm dữ liệu theo nhiều khả năng khác nhau	Dữ liệu đầu vào	Phức tạp
49	Thành viên có thể xem những thông tin cần thiết mà mình được phép của mục tin tức thông qua các thông tin lựa chọn như: tiêu đề, từ khóa, loại tin, tác giả, nội dung	Yêu cầu truy vấn	Phức tạp
50	Thành viên có thể xem những bài viết trên diễn đàn mà mình được quyền truy cập thông qua các thông tin lựa chọn như: tiêu đề, từ khóa, loại tin, tác giả, nội dung	Yêu cầu truy vấn	Phức tạp
51	Thành viên có thể xem danh sách thống kê các tin, bài theo các thông tin trích yếu như: tiêu đề, từ khóa, ngày ban hành, tác giả, thời lượng, ban biên tập	Yêu cầu truy vấn	Phức tạp
52	Thành viên có thể gửi tin nhắn cho quản trị hệ thống để thay đổi thông tin thành viên của mình	Dữ liệu đầu vào	Đơn giản
53	Thành viên có thể gửi thư điện tử cho quản trị hệ thống để thay đổi thông tin thành viên của mình	Dữ liệu đầu vào	Đơn giản
54	Thành viên có thể xem tin tức	Yêu cầu truy vấn	Trung bình
55	Thành viên có thể viết bài	Dữ liệu đầu vào	Đơn giản
56	Thành viên có thể xóa bài viết của mình trước khi nó được đăng trên website	Dữ liệu đầu vào	Đơn giản
57	Thành viên có thể sửa bài viết của mình trước khi nó được đăng trên website	Dữ liệu đầu vào	Trung bình
58	Thành viên có thể gửi các thông tin phản hồi về tin tức	Dữ liệu đầu vào	Đơn giản

59	Thành viên có thể bình luận tin tức	Dữ liệu đầu vào	Đơn giản
60	Thành viên có thể đánh giá tin tức	Dữ liệu đầu vào	Đơn giản
61	Thành viên có thể tải về dữ liệu, công cụ cần thiết	Dữ liệu tra cứu	Đơn giản
62	Thành viên có thể tạo chủ đề	Dữ liệu đầu vào	Đơn giản
63	Thành viên có thể sửa chủ đề	Dữ liệu đầu vào	Phức tạp
64	Thành viên có thể xóa chủ đề	Dữ liệu đầu vào	Đơn giản
65	Biên tập viên có thể soạn tin tức	Dữ liệu đầu vào	Đơn giản
66	Biên tập viên có quyền hạn ở chuyên mục mà mình phụ trách căn cứ vào tên người dùng và mật khẩu	Cơ sở dữ liệu	Đơn giản
67	Biên tập viên có thể đăng tin tức lên website	Dữ liệu đầu vào	Đơn giản
68	Biên tập viên có thể đăng bài viết lên website	Dữ liệu đầu vào	Đơn giản
69	Biên tập viên có thể xét duyệt bài viết của thành viên gửi lên	Dữ liệu đầu ra	Trung bình
70	Biên tập viên có thể xét duyệt tin tức của thành viên gửi lên	Dữ liệu đầu ra	Trung bình
71	Biên tập viên có thể xét duyệt bài viết của khách gửi lên	Dữ liệu đầu ra	Trung bình
72	Biên tập viên có thể xét duyệt tin tức của khách gửi lên	Dữ liệu đầu ra	Trung bình
73	Biên tập viên có thể sửa tin tức	Dữ liệu đầu vào	Đơn giản
74	Biên tập viên có thể xóa tin tức	Dữ liệu đầu vào	Đơn giản
75	Biên tập viên có thể gửi thông báo đến thành viên qua thư điện tử	Dữ liệu đầu vào	Đơn giản
76	Biên tập viên có thể gửi thông báo đến thành viên qua tin nhắn	Dữ liệu đầu vào	Đơn giản
77	Biên tập viên có thể xem những thông tin liên quan đến bài viết của mình để tạo thành chuỗi thông tin liên quan	Yêu cầu truy vấn	Phức tạp
78	Biên tập viên có thể nhúng thêm hình ảnh, âm thanh, tập tin tài liệu vào để minh họa cho bài viết	Dữ liệu đầu vào	Đơn giản
79	Biên tập viên có thể xem lại bài viết trước khi đăng tin tức và cập nhật vào cơ sở dữ liệu	Dữ liệu đầu ra	Đơn giản
80	Biên tập viên có thể xóa các tin tức không muốn đăng nữa	Dữ liệu đầu vào	Đơn giản
81	Biên tập viên có thể xóa các tin tức đã quá hạn	Dữ liệu đầu vào	Đơn giản
82	Hệ thống có thể tự động không hiển thị các tin tức đã quá ngày cho phép đăng	Yêu cầu truy vấn	Phức tạp
83	Hệ thống có thể tự động tải lên ảnh và cố định kích cỡ ảnh khi hiển thị	Dữ liệu tra cứu	Trung bình
84	Hệ thống có thể hiển thị các tin bài liên quan thông qua từ khóa	Yêu cầu truy vấn	Phức tạp

85	Hệ thống có thể cho phép hiển thị một số lượng nhất định các tin bài cùng một lúc	Yêu cầu truy vấn	Phức tạp
86	Hệ thống có thể lưu vết thông tin truy cập của người duyệt website	Dữ liệu đầu ra	Phức tạp
87	Hệ thống có thể đưa ra thông báo thống kê về các bài viết được đọc nhiều nhất nhằm mục đích tối ưu hóa hoạt động của website	Dữ liệu đầu ra	Phức tạp
88	Hệ thống có thể đưa ra thông báo thống kê thông tin người sử dụng: hệ điều hành, trình duyệt, màu màn hình, độ phân giải	Dữ liệu đầu ra	Phức tạp
89	Người quản trị hệ thống có thể thực hiện đồng bộ dữ liệu với hệ thống thông tin điều hành, tác nghiệp nội bộ	Yêu cầu truy vấn	Phức tạp
90	Hệ thống có thể tự động phân phát các bản tin trên website đến địa chỉ thư điện tử của các độ giả có yêu cầu	Dữ liệu đầu ra	Trung bình

Chuyển đổi yêu cầu chức năng sang trường hợp sử dụng:

TT	Tên trường hợp sử dụng	Tên tác nhân chính	Tên tác nhân phụ	Mô tả trường hợp sử dụng	Mức độ cần thiết
1	Quản trị bài viết	Người quản trị hệ thống	Khách, thành viên, biên tập viên	Trường hợp sử dụng khái quát hóa	B
				Người quản trị hệ thống có thể thêm bài viết	
				Người quản trị hệ thống có thể sửa chi tiết bài viết	
				Người quản trị hệ thống có thể sửa bài viết	
				Khách có thể xem bài viết trên diễn đàn	
				Khách có thể xem những bài viết trên diễn đàn mà mình được quyền truy cập thông qua các thông tin lựa chọn (tiêu đề, từ khóa, loại tin, tác giả, nội dung)	
				Thành viên có thể những bài viết trên diễn đàn mà mình được quyền truy cập thông qua các thông tin lựa chọn (tiêu đề, từ khóa, loại tin, tác giả, nội dung)	
				Thành viên có thể xem danh sách thống kê các bài viết theo các thông tin trích yếu (tiêu đề, từ khóa, ngày ban hành, tác giả, thời lượng, ban biên tập)	
				Thành viên có thể viết bài	

				Thành viên có thể xóa bài viết của mình trước khi nó được đăng lên website	
				Thành viên có thể sửa bài viết của mình trước khi nó được đăng lên website	
				Biên tập viên có thể đăng bài viết lên website	
				Biên tập viên có thể xét duyệt bài viết của thành viên gửi lên	
				Biên tập viên có thể xem những thông tin liên quan đến bài viết của mình để tạo thành chuỗi thông tin liên quan	
				Biên tập viên có thể nhúng thêm hình ảnh, âm thanh, tập tin tài liệu vao để minh họa cho bài viết	
				Biên tập viên có thể xem lại bài viết trước khi đăng tin tức và và cập nhật vào cơ sở dữ liệu	
				Hệ thống có thể hiển thị các tin bài liên quan thông qua từ khóa	
				Hệ thống có thể cho phép hiển thị một số lượng nhất định các tin bài cùng một lúc	
				Các tùy chọn dùng để tìm kiếm có thể kết hợp với nhau theo nhiều cách để tìm dữ liệu theo nhiều khả năng khác nhau	
				Thành viên có thể tải về dữ liệu, công cụ cần thiết	
				Khách có thể tải về dữ liệu, công cụ cần thiết	
				Khách có thể tải về thư viện ảnh	
2	Quản trị tin	Người quản trị hệ thống	Khách, thành viên, biên tập viên	Trường hợp sử dụng khái quát hóa	B
				Người quản trị hệ thống có thể thêm tin mới	
				Người quản trị hệ thống có thể sửa tin tức	
				Người quản trị hệ thống có thể xóa tin tức	
				Khách có thể xem tin tức	
				Khách có thể đăng ký các chuyên mục tin mà mình quan tâm để nhận được bản tin do hệ thống thông báo vào thời điểm định trước	

				Khách có thể xem những thông tin cần thiết mà mình được phép của mục tin tức thông qua các thông tin lựa chọn (tiêu đề, từ khóa, loại tin, tác giả, nội dung)	
				Thành viên có thể xem những thông tin cần thiết mà mình được phép của mục tin tức thông qua các thông tin lựa chọn (tiêu đề, từ khóa, loại tin, tác giả, nội dung)	
				Thành viên có thể xem danh sách thống kê các tin theo các thông tin trích yếu (tiêu đề, từ khóa, ngày ban hành, tác giả, thời lượng, ban biên tập)	
				Thành viên có thể xem tin tức	
				Thành viên có thể bình luận tin tức	
				Thành viên có thể đánh giá tin tức	
				Biên tập viên có thể soạn tin tức	
				Biên tập viên có thể đăng tin tức lên website	
				Biên tập viên có thể xét duyệt tin tức của thành viên gửi lên	
				Biên tập viên có thể xét duyệt tin tức của khách gửi lên	
				Biên tập viên có thể xét duyệt bài viết của khách gửi lên	
				Biên tập viên có thể sửa tin tức	
				Biên tập viên có thể xóa tin tức	
				Biên tập viên có thể xóa các tin tức không muốn đăng nữa	
				Biên tập viên có thể xóa các tin tức đã quá hạn	
				Hệ thống có thể tự động không hiển thị các tin tức đã quá ngày cho phép đăng	
				Hệ thống có thể tự động tải về ảnh và cố định kích cỡ ảnh khi hiển thị	
				Các tùy chọn dùng để tìm kiếm có thể kết hợp với nhau theo nhiều cách để tìm dữ liệu theo nhiều khả năng khác nhau	
				Thành viên có thể tải về dữ liệu, công cụ cần thiết	
				Khách có thể tải về thư viện ảnh	
3	Quản trị cấu hình hệ thống	Người quản trị hệ thống			B

				Người quản trị hệ thống có thể cấu hình giao diện giao tiếp với người sử dụng của hệ thống: thay đổi bố trí	
				Người quản trị hệ thống có thể cấu hình giao diện giao tiếp với người sử dụng của hệ thống: thay đổi cách hiển thị các chuyên mục	
				Người quản trị hệ thống có thể cấu hình giao diện giao tiếp với người sử dụng của hệ thống: cách hiển thị các quảng cáo	
				Người quản trị hệ thống có thể cấu hình giao diện giao tiếp với người sử dụng của hệ thống: thay đổi cách thức hiển thị tin tức	
				Người quản trị hệ thống có thể cấu hình giao diện giao tiếp với người sử dụng của hệ thống: thay đổi cách thức hiển thị bài viết	
				Người quản trị hệ thống cấu hình website thông qua một bảng điều khiển tích hợp	
				Người quản trị hệ thống có thể quản lý các chức năng trên diễn đàn	
				Người quản trị hệ thống có thể quản lý các tiện ích trên trang web (tải về, đếm, khảo sát,...)	
4	Quản lý sao lưu	Người quản trị hệ thống			B
				Người quản trị hệ thống có thể sao lưu dữ liệu của website phục vụ khôi phục hoạt động khi xảy ra sự cố	
5	Đồng bộ dữ liệu	Người quản trị hệ thống		Người quản trị hệ thống có thể thực hiện đồng bộ dữ liệu với hệ thống thông tin điều hành, tác nghiệp nội bộ	B
6	Quản trị chuyên mục	Người quản trị hệ thống			B
				Người quản trị hệ thống có thể thay đổi chuyên mục	
				Người quản trị hệ thống có thể xóa chuyên mục	
				Người quản trị hệ thống có thể tạo mới chuyên mục	
				Người quản trị hệ thống có thể liệt kê các chuyên mục	

7	Quản trị chủ đề	Người quản trị hệ thống			B
				Người quản trị hệ thống có thể thêm mới chủ đề thông tin	
				Người quản trị hệ thống có thể thay đổi chủ đề thông tin	
				Người quản trị hệ thống có thể xóa chủ đề thông tin	
				Người quản trị hệ thống có thể liệt kê các chủ đề thông tin	
8	Quản trị thư viện dữ liệu	Người quản trị hệ thống			B
				Người quản trị hệ thống có thể thêm hình ảnh, dữ liệu vào thư viện dữ liệu	
				Người quản trị hệ thống có thể xóa hình ảnh, dữ liệu ra khỏi thư viện dữ liệu	
				Người quản trị hệ thống có thể thay đổi hình ảnh, dữ liệu trong thư viện dữ liệu	
9	Đăng ký thành viên	Người quản trị hệ thống	Khách		B
				Khách có thể đăng ký với hệ thống để thành thành viên	
				Sau khi đăng ký thành công khách phải kích hoạt thư điện tử thì tài khoản người dùng mới có hiệu lực	
10	Phân quyền sử dụn	Người quản trị hệ thống			B
				Người quản trị hệ thống có thể tạo lập các quyền sử dụng của người sử dụng	
				Người quản trị hệ thống có thể sửa đổi các quyền sử dụng của người sử dụng	
				Người quản trị hệ thống có thể hủy bỏ các quyền sử dụng của người sử dụng	
				Người quản trị hệ thống có thể nhóm các quyền riêng lẻ lại thành nhóm quyền sử dụng cho các nhóm người sử dụng	
				Người quản trị hệ thống có thể xóa người sử dụng	
11	Quản lý thông tin phản hồi	Người quản trị hệ thống	Biên tập viên, thành viên,	Trường hợp sử dụng khái quát hóa	B

			khách		
				Người quản trị hệ thống có thể nhận thông tin phản hồi từ người truy cập	
				Người quản trị hệ thống có thể đăng thông tin phản hồi từ người truy cập lên website	
				Người quản trị hệ thống có thể trả lời cho người truy cập	
				Khách có thể gửi thông tin phản hồi về tin tức qua thư điện tử	
				Thành viên có thể gửi tin nhắn cho quản trị hệ thống để thay đổi thông tin thành viên của mình	
				Thành viên có thể gửi thư điện tử cho quản trị hệ thống để thay đổi thông tin thành viên của mình	
				Biên tập viên có thể gửi thông báo đến thành viên qua thư điện tử	
				Biên tập viên có thể gửi thông báo đến thành viên qua thư tin nhắn	
				Khi người sử dụng thực hiện thao tác xóa hay chỉnh sửa, hệ thống hiện thông báo xác nhận cho tác vụ này	
				Thành viên có thể gửi các thông tin phản hồi về tin tức	
12	Quản trị thăm dò ý kiến	Người quản trị hệ thống			B
				Người quản trị hệ thống có thể thêm mới cuộc thăm dò ý kiến	
				Người quản trị hệ thống có thể thay đổi nội dung cuộc thăm dò ý kiến	
				Người quản trị hệ thống có thể tính toán kết quả thăm dò ý kiến	
13	Quản lý chủ đề	Thành viên			
				Thành viên có thể tạo chủ đề	
				Thành viên có thể sửa chủ đề	
				Thành viên có thể xóa chủ đề	
14	Quản lý thông				B

	tin phiên làm việc				
				Hệ thống có thể lưu vết thông tin truy cập của người duyệt website	
				Hệ thống có thể đưa ra thông báo thống kê về các bài viết được đọc nhiều nhất nhằm mục đích tối ưu hóa hoạt động của website	
				Hệ thống có thể đưa ra thông báo thống kê thông tin người sử dụng như hệ điều hành, trình duyệt, màu màn hình, độ phân giải	
				Hệ thống có thể tự động phân phát các bản tin trên website đến địa chỉ thư điện tử của các độc giả có yêu cầu	
15	Đăng nhập hệ thống	Người quản trị hệ thống	Khách, biên tập viên, thành viên		8
				Người sử dụng nhập tên người dùng duy nhất và mật khẩu để đăng nhập hệ thống	

Tính toán điểm các tác nhân tương tác, trao đổi thông tin với phần mềm:

TT	Loại tác nhân	Mô tả	Số tác nhân	Điểm của từng loại tác nhân	Ghi chú
1	Đơn giản	Thuộc loại giao diện của chương trình		0	
2	Trung bình	Giao diện tương tác hoặc phục vụ một giao thức hoạt động (hệ thống)	1	2	
3	Phức tạp	Giao diện đồ họa (quản trị hệ thống, khách, biên tập viên, thành viên)	4	12	
	Cộng (1+2+3)			14	

Tính toán điểm các trường hợp sử dụng:

TT	Loại	Số trường hợp sử dụng	Điểm của từng loại trường hợp sử dụng	Mô tả
1	B			Các yêu cầu phải thỏa mãn thì phần mềm mới được chấp nhận
	Đơn giản	9	45	Trường hợp sử dụng đơn giản ≤ 3 giao dịch hoặc đường chỉ thị
	Trung bình	1	10	
	Phức tạp	5	75	
2	M			Các chức năng không phải là cốt lõi hay các chức năng phụ trợ hoặc theo yêu cầu của bên đặt hàng
	Đơn giản		0	Trường hợp sử dụng trung bình từ 4 đến 7 giao dịch
	Trung bình		0	
	Phức tạp		0	
3	T			Các yêu cầu được bên phát triển phần mềm tư vấn thêm hoặc đưa ra để bên đặt hàng lựa chọn thêm nếu muốn
	Đơn giản		0	Trường hợp sử dụng phức tạp > 7 giao dịch
	Trung bình		0	
	Phức tạp		0	
	Cộng (1+2+3)		130	

Tính toán hệ số phức tạp kỹ thuật – công nghệ:

TT	Các hệ số	Giá trị xếp hạng	Kết quả	Ghi chú
I	Hệ số kỹ thuật – công nghệ (TFW)		41	
1	Hệ thống phân tán	3	6	
2	Tính chất đáp ứng tức thời hoặc yêu cầu đảm bảo thông lượng	4	4	
3	Hiệu quả sử dụng trực tuyến	3	3	
4	Độ phức tạp của xử lý bên trong	3	3	
5	Mã nguồn phải tái sử dụng được	3	3	
6	Dễ cài đặt	4	2	
7	Dễ sử dụng	4	2	
8	Khả năng chuyển đổi	3	6	
9	Khả năng dễ thay đổi	3	3	
10	Sử dụng đồng thời	3	3	
11	Có các tính năng bảo mật đặc biệt	3	3	
12	Cung cấp truy nhập trực tiếp tới các phần mềm bên thứ ba	3	3	
13	Yêu cầu phương tiện đào tạo đặc biệt cho người sử dụng	0	0	
II	Hệ số phức tạp về kỹ thuật – công nghệ (TCF)		1.01	

Tính hệ số tác động môi trường và nhóm làm việc, hệ số phức tạp về môi trường:

TT	Các hệ số tác động môi trường	Giá trị xếp hạng	Kết quả	Đánh giá độ ổn định KN
I	Hệ số tác động môi trường và nhóm làm việc (EFW)	16	17	
	Đánh giá cho từng thành viên			
1	Có áp dụng quy trình phát triển phần mềm theo mẫu RUP và có hiểu biết về RUP			
2	Có kinh nghiệm về ứng dụng tương tự			
3	Có kinh nghiệm về hướng đối tượng			
4	Có khả năng lãnh đạo nhóm			
5	Tính chất năng động			
	Đánh giá chung cho dự án			
6	Độ ổn định của các yêu cầu			
7	Có sử dụng các nhân viên làm bán thời gian			
8	Dùng ngôn ngữ lập trình loại khó			
II	Hệ số phức tạp về môi trường (EF)			
III	Độ ổn định kinh nghiệm (ES)			
IV	Nội suy thời gian lao động (P)			

Tính trị giá phần mềm:

TT	Hạng mục	Diễn giải	Giá trị	Ghi chú
I	Tính điểm trường hợp sử dụng			
1	Điểm tác nhân (TAW)			
2	Điểm trường hợp sử dụng (TBF)			
3	Điểm UUCP			
4	Hệ số phức tạp về kỹ thuật công nghệ (TCF)			
5	Hệ số phức tạp về môi trường (EF)			
6	Điểm AUCP			
II	Nội suy thời gian lao động (P)			
III	Giá trị nỗ lực thực tế (E)			
IV	Mức lương lao động bình quân (H)			đồng
V	Định giá phần mềm nội bộ (G)			đồng