

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**  
**TRƯỜNG ĐẠI HỌC DUY TÂN**

---

**BÀI GIẢNG MÔN HỌC**  
**CÔNG NGHỆ PHẦN MỀM**  
**Mã môn học: CS 403**

Bộ môn: Công nghệ phần mềm  
Khoa: Công nghệ thông tin

Đà Nẵng, tháng 8 năm 2021

## MỤC LỤC

Chương 1. TỔNG QUAN VỀ CÔNG NGHỆ PHẦN MỀM.....	5
1.1. Định nghĩa phần mềm .....	5
1.2. Kiến trúc phần mềm.....	7
1.3. Các khái niệm.....	8
1.4. Đặc tính chung của phần mềm.....	9
1.5. Độ đo phần mềm.....	9
1.5.1 Độ đo theo kích cỡ .....	11
1.5.2. Độ đo theo điểm chức năng .....	11
1.5.3 Độ đo điểm chức năng mở rộng.....	13
1.6. Các ứng dụng phần mềm.....	16
Chương 2. KHỦNG HOẢNG PHẦN MỀM.....	18
2.1. Khủng hoảng phần mềm.....	18
2.2. Những vấn đề trong sản xuất phần mềm.....	20
Chương 3. CÔNG NGHỆ PHẦN MỀM .....	23
3.1. Lịch sử phát triển ngành công nghiệp phần mềm.....	23
3.2. Sự phát triển của các phương pháp thiết kế phần mềm.....	25
3.3. Định nghĩa công nghệ phần mềm.....	27
3.4. Vòng đời của sản phẩm phần mềm.....	28
3.5 Một số mô hình xây dựng phần mềm.....	29
3.5.1 Mô hình tuyến tính (The linear sequential model).....	29
3.5.2 Mô hình mẫu (Prototyping model).....	30
3.5.3 Mô hình xoắn ốc (The spiral model).....	31
3.5.4 Mô hình đài phun nước .....	32
3.5.5 Mô hình phát triển dựa trên thành phần .....	33
3.6 Phương pháp phát triển phần mềm .....	34

3.7 Vai trò của người dùng trong giai đoạn phát triển phần mềm.....	34
3.8. Quy trình phát triển phần mềm.....	35
Chương 4. DỰ ÁN PHẦN MỀM.....	36
4.1. Tổng quan về dự án phần mềm.....	36
4.2. Người quản lý dự án và các tiêu chuẩn chọn lựa người QLDA .....	44
4.3. Bốn “chiều” của dự án .....	49
4.4. Lập kế hoạch.....	50
4.5. Sự giám sát dự án .....	51
4.6. Các định chuẩn và tiêu chuẩn đo lường .....	51
4.7. Các nguyên tắc cơ bản về kỹ thuật .....	52
4.8. Các giai đoạn phát triển dự án .....	52
4.9. Lập kế hoạch dự án.....	54
4.10. Tổ chức dự án.....	55
Chương 5. YÊU CẦU PHẦN MỀM.....	70
5.1. Tổng quát về yêu cầu phần mềm.....	70
5.2. Kỹ thuật xác định yêu cầu.....	70
5.3. Nội dung xác định yêu cầu.....	72
5.4. Các nguyên lý phân tích yêu cầu.....	73
5.5 Tài liệu đặc tả yêu cầu.....	75
5.6.Các bước phân tích và đặc tả yêu cầu .....	77
Chương 6. PHƯƠNG PHÁP THIẾT KẾ HỆ THỐNG.....	84
6.1. Khái niệm thiết kế hệ thống.....	84
6.2. Phương pháp thiết kế hệ thống .....	84
Chương 7. KỸ THUẬT THIẾT KẾ PHẦN MỀM.....	88
7.1. Khái niệm thiết kế phần mềm .....	88
7.2. Phương pháp thiết kế phần mềm.....	88

Chương 8. KỸ THUẬT LẬP TRÌNH.....	89
8.1. Khái quát về ngôn ngữ lập trình .....	90
8.2. Ngôn ngữ lập trình và sự ảnh hưởng đến công nghệ phần mềm .....	93
8.3. Cấu trúc chương trình.....	94
8.4. Lập trình hướng hiệu quả thực hiện .....	98
Chương 9. KIỂM THỬ PHẦN MỀM.....	101
9.1. Khái niệm kiểm thử .....	101
9.2. Phương pháp kiểm thử .....	102
9.3. Thiết kế các trường hợp kiểm thử .....	104
Chương 10. BẢO TRÌ PHẦN MỀM.....	106
10.1. Khái niệm bảo trì phần mềm.....	106
10.2. Quy trình nghiệp vụ bảo trì phần mềm.....	115
10.3. Các vấn đề về bảo trì phần mềm hiện nay .....	116
Chương 11. CÁC CHỦ ĐỀ KHÁC TRONG CNPM .....	117
11.1. Ước lượng chi phí phần mềm .....	117
11.2. Năng suất phần mềm.....	119
11.3. Kỹ thuật ước lượng .....	124
TÀI LIỆU THAM KHẢO .....	129

## Chương 1. TỔNG QUAN VỀ CÔNG NGHỆ PHẦN MỀM

Mục đích của chương này là đưa ra những nhận định cơ bản và tạo nên một bức tranh cơ sở về những phương pháp tiếp cận khác nhau của công việc tạo nên công nghệ phần mềm. Các vấn đề cần làm rõ, chi tiết thêm sẽ được trình bày ở các chương tiếp sau của giáo trình

### 1.1. Định nghĩa phần mềm

Mục tiêu của công nghệ phần mềm là tạo ra những phần mềm tốt, giảm đến tối thiểu những rủi ro có thể gây cho các người liên quan. Trong quá trình đề cập, chúng ta sử dụng các thuật ngữ:

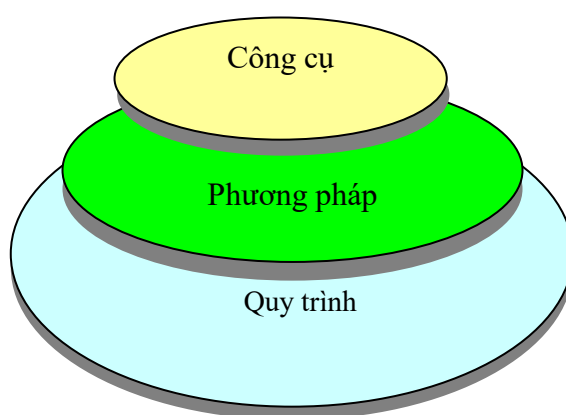
Phần mềm (software): là một tập hợp các câu lệnh được viết bằng một hoặc nhiều ngôn ngữ lập trình, nhằm tự động thực hiện một số các chức năng giải quyết một bài toán nào đó.

Công nghệ (engineering): là cách sử dụng các công cụ, các kỹ thuật trong cách giải quyết một vấn đề nào đó.

Công nghệ phần mềm (software engineering): là việc áp dụng các công cụ, các kỹ thuật một cách hệ thống trong việc phát triển các ứng dụng dựa trên máy tính. Đó chính là việc áp dụng các quan điểm, các tiến trình có kỷ luật và lượng hoá được, có bài bản và hệ thống để phát triển, vận hành và bảo trì phần mềm.

Theo quan điểm của nhiều nhà nghiên cứu, có thể nhìn công nghệ phần mềm là một mô hình được phân theo ba tầng mà tất cả các tầng này đều nhằm tới mục tiêu chất lượng, chi phí, thời hạn phát triển phần mềm.

Mô hình được phân theo ba tầng của công nghệ phần mềm được mô tả như sau:



**Hình 1.1 Mô hình phân theo ba tầng của công nghệ phần mềm**

Ở đây tầng quy trình (process) liên quan tới vấn đề quản trị phát triển phần mềm như lập kế hoạch, quản trị chất lượng, tiến độ, chi phí, mua bán sản phẩm phụ, cấu hình phần mềm, quản trị sự thay đổi, quản trị nhân sự (trong môi trường làm việc nhóm), việc chuyển giao, đào tạo, tài liệu;

Tầng phương pháp (methods) hay cách thức, công nghệ, kỹ thuật để làm phần mềm: liên quan đến tất cả các công đoạn phát triển hệ thống như nghiên cứu yêu cầu, thiết kế, lập trình, kiểm thử và bảo trì. Phương pháp dựa trên những nguyên lý cơ bản nhất cho tất cả các lĩnh vực công nghệ kể cả các hoạt động mô hình hoá và kỹ thuật mô tả.

Tầng công cụ (tools) liên quan đến việc cung cấp các phương tiện hỗ trợ tự động hay bán tự động cho các tầng quá trình và phương pháp (công nghệ).

Qua sơ đồ trên, ta thấy rõ công nghệ phần mềm là một khái niệm đề cập không chỉ tới các công nghệ và công cụ phần mềm mà còn tới cả cách thức phối hợp công nghệ, phương pháp và công cụ theo các quy trình nghiêm ngặt để làm ra sản phẩm có chất lượng.

Kỹ sư phần mềm (software engineer): là một người biết cách áp dụng rộng rãi những kiến thức về cách phát triển ứng dụng vào việc tổ chức phát triển một cách có hệ thống các ứng dụng. Công việc của người kỹ sư phần mềm là: đánh giá, lựa chọn, sử dụng những cách tiếp cận có tính hệ thống, chuyên biệt, rõ ràng trong việc phát triển, đưa vào ứng dụng, bảo trì, và thay thế phần mềm.

Do đặc điểm nghề nghiệp, người kỹ sư phần mềm phải có những kỹ năng cơ bản như:

Định danh, đánh giá, cài đặt, lựa chọn một phương pháp luận thích hợp và các công cụ CASE.

Biết cách sử dụng các mẫu phần mềm (prototyping).

Biết cách lựa chọn ngôn ngữ, phần cứng, phần mềm.

Quản lý cấu hình, lập sơ đồ và kiểm soát việc phát triển của các tiến trình.

Lựa chọn ngôn ngữ máy tính và phát triển chương trình máy tính.

Đánh giá và quyết định khi nào loại bỏ và nâng cấp các ứng dụng.

Mục tiêu của kỹ sư phần mềm là sản xuất ra các sản phẩm có chất lượng cao và phù hợp với các quy trình phát triển chuẩn mực.

Việc phát triển (development): được bắt đầu từ khi quyết định phát triển sản phẩm phần mềm và kết thúc khi sản phẩm phần mềm được chuyển giao cho người sử dụng.

Việc sử dụng (operations): là việc xử lý, vận hành hằng ngày sản phẩm phần mềm

Việc bảo trì (maintenance): thực hiện những thay đổi mang tính logic đối với hệ thống và chương trình để chữa những lỗi cố định, cung cấp những thay đổi về công việc, hoặc làm cho phần mềm được hiệu quả hơn.

Việc loại bỏ (retirement): thường là việc thay thế các ứng dụng hiện thời bởi các ứng dụng mới.

Nhiều người đồng nghĩa thuật ngữ *phần mềm* với những chương trình máy tính. Tuy nhiên, chúng ta muốn đưa ra một định nghĩa khái quát hơn ở đó phần mềm không chỉ là những chương trình mà còn là toàn bộ những tài liệu liên quan và cấu hình dữ liệu cần thiết để tạo nên những chương trình đó thực thi một cách chính xác.

Một hệ thống phần mềm thường bao gồm một số lượng các chương trình riêng biệt, các file cấu hình được sử dụng để cài đặt những chương trình này, các tài liệu hệ thống mô tả cấu trúc của hệ thống, và tài liệu người dùng giải thích cách sử dụng hệ thống và những Website cho người sử dụng có thể tải những thông tin về sản phẩm mới nhất.

## **1.2. Kiến trúc phần mềm**

Kiến trúc phần mềm là khung cơ bản cho việc cấu trúc nên hệ thống. Các thuộc tính của một hệ thống như việc thực thi, bảo mật và lợi ích bị ảnh hưởng bởi kiến trúc được sử dụng.

Những quyết định thiết kế kiến trúc bao gồm các quyết định trên các loại ứng dụng, sự phân tán hệ thống, những kiểu kiến trúc được sử dụng và những hướng mà kiến trúc nên được lập tài liệu và phát triển theo.

Những mô hình kiến trúc khác chẳng hạn như một mô hình cấu trúc, một mô hình điều khiển và một mô hình phân tích có thể được phát triển trong quá trình thiết kế kiến trúc.

Những mô hình có tổ chức của một hệ thống bao gồm các mô hình kho chứa, các mô hình client-server và các mô hình máy trừu tượng. Các mô hình kho chứa chia sẻ dữ liệu ở một phần lưu trữ chung. Những mô hình client-server thường phân tán dữ liệu. Những mô hình máy ảo được phân lớp, với mỗi lớp được thực hiện sử dụng các điều kiện được cung cấp bởi lớp thiết lập của nó.

Những kiểu phân tích gồm phân tích hướng đối tượng và hướng chức năng. Những mô hình đường ống là những mô hình chức năng, và các mô hình đối tượng dựa trên việc gắn lỏng các thực thể mà tự mang trạng thái và các thao tác.

Các kiểu điều khiển gồm việc điều khiển trung tâm và điều khiển dựa trên sự kiện. Trong các mô hình điều khiển trung tâm, các quyết định điều khiển được thực hiện phụ thuộc vào trạng thái của hệ thống; trong các mô hình điều khiển dựa trên sự kiện, các sự kiện bên ngoài điều khiển hệ thống.

Các kiến trúc tham khảo có thể được sử dụng như phương tiện để thảo luận các kiến trúc miền xác định và để quyết định, so sánh các thiết kế kiến trúc.

Chú ý rằng giá trị chủ yếu của kiến trúc tham khảo là ở việc phân loại và so sánh các công cụ và môi trường CASE được tích hợp. Hơn nữa, nó cũng có thể được sử dụng trong đào tạo để nêu bật các đặc điểm chính của những môi trường và để thảo luận chúng theo một cách chung nhất.

### 1.3. Các khái niệm

**Bảng 1. 1 Những khái niệm trong phần mềm**

Hỏi	Giải đáp
Phần mềm là gì?	Những chương trình máy tính và những tài liệu liên quan. Những sản phẩm phần mềm có thể được phát triển cho một khách hàng cụ thể hoặc có thể được phát triển cho thị trường chung.
Công nghệ phần mềm là gì?	Công nghệ phần mềm là một ngành công nghệ tập trung hướng vào các sản phẩm phần mềm.
Sự khác nhau giữa công nghệ phần mềm và khoa học máy tính?	Khoa học máy tính thì quan tâm chú ý đến những lý thuyết và cơ sở; còn công nghệ phần mềm thì quan tâm đến tính thực tiễn của ứng dụng và việc cho ra những phần mềm có ích.
Sự khác nhau giữa công nghệ phần mềm công nghệ hệ thống?	Công nghệ hệ thống tập trung toàn bộ vào phát triển hệ thống lấy cơ sở là máy tính, nó bao gồm có phần cứng, phần mềm và những công nghệ xử lý. Công nghệ phần mềm chỉ là một phần của quá trình này.
Một tiến trình phần mềm là gì?	Đó là một tập những công việc mà mục đích của nó là sự phát triển hay nâng cấp của phần mềm.
Một mẫu tiến trình phần mềm là gì?	Đó là một thể hiện đơn giản của một tiến trình phần mềm, được đưa ra từ một viễn cảnh đặc trưng.
Những chi phí của công nghệ phần mềm là gì?	Xấp xỉ 60% chi phí là chi phí phát triển, 40% là chi phí kiểm tra. Đối với phần mềm khách hàng, chi phí phát sinh thường vượt quá chi phí phát triển. [ <i>theo tài liệu thống kê Software Engineering-A Practitioner's Approach, Roger S.Pressman, 5<sup>th</sup> Edition, McGraw-Hill.Inc,2010</i> ]
Những phương thức công nghệ phần mềm là gì?	Đó là những cách tiếp cận mang tính cấu trúc tới sự phát triển phần mềm bao gồm những mẫu hệ thống, những chú giải, những luật, những lời khuyên cho việc thiết kế, và những sự chỉ đạo tiến trình.
Công nghệ phần mềm có sự trợ giúp của máy tính là gì?	Đó là những hệ thống phần mềm có ý định cung cấp những hỗ trợ một cách tự động cho những hoạt động trong tiến trình phần mềm. Những hệ thống này thường được sử dụng để hỗ trợ về mặt phương thức.
Một phần mềm tốt có những tính chất nào?	Là một phần mềm cung cấp được những tính năng và thực hiện được các thực thi được yêu cầu bởi người dùng, dễ bảo trì, đáng tin cậy, và sử dụng được.
Công nghệ phần mềm phải đối mặt với những thử thách chính nào?	Tính đa dạng ngày càng tăng, đòi hỏi của việc giảm thời gian xuất xưởng, tăng tính tin cậy của phần mềm.



#### 1.4. Đặc tính chung của phần mềm

Cùng với những dịch vụ mà nó cung cấp, những sản phẩm phần mềm có một số thuộc tính được gắn kết với nó và phản ánh chất lượng của phần mềm đó. Những thuộc tính này không trực tiếp tập trung vào mục đích của phần mềm. Thay vào đó, chúng phản ánh mức độ hoạt động cùng với cấu trúc và tổ chức của chương trình nguồn và những tài liệu liên quan đến phần mềm. Những ví dụ về những thuộc tính này (đôi khi còn gọi là những thuộc tính không mang tính chức năng) là thời gian phản hồi của phần mềm đối những truy vấn của người dùng và khả năng dễ hiểu của mã chương trình.

Tập các thuộc tính mà ta có thể kỳ vọng vào một sản phẩm phần mềm phụ thuộc vào ứng dụng của nó. Bởi vậy, một hệ thống nhà băng cần phải được bảo vệ, một trò chơi tương tác cần phải được phản hồi, một hệ thống chuyển mạch điện thoại cần phải có tính thực tế vv... Những điều này được tổng quát hóa trong bảng dưới là những đặc điểm cần thiết cho một hệ thống phần mềm được thiết kế tốt.

**Bảng 1. 2 Những thuộc tính cần thiết của một phần mềm tốt**

Đặc điểm phần mềm	Mô tả
Khả năng bảo trì	Phần mềm nên được viết theo cách mà nó có thể đáp ứng được sự thay đổi nhu cầu của khách hàng. Đây là một thuộc tính quan trọng vì sự thay đổi của phần mềm là một trình tự không thể tránh được của môi trường kinh doanh thay đổi.
Tính đáng tin cậy	Tính đáng tin cậy của phần mềm bao gồm một loạt các thuộc tính khác như là tính thực tế, bảo mật và an toàn. Một phần mềm gọi đáng tin cậy thì nó không được gây ra những tổn thất về kinh tế cũng như vật chất thậm chí ngay cả trong trường hợp hỏng hóc hệ thống.
Tính hiệu quả	Phần mềm không được gây ra lãng phí tài nguyên của hệ thống như bộ nhớ hay các vòng xử lý. Tính hiệu quả do vậy gồm có khả năng đáp trả, thời gian xử lý và sự tận dụng bộ nhớ.
Khả năng sử dụng	Phần mềm phải dễ dùng, không đòi hỏi quá nhiều công sức của người sử dụng. Điều này có nghĩa là nó cần phải có một giao diện người dùng thích hợp và tài liệu đầy đủ.

#### 1.5. Độ đo phần mềm

Người ta cần phải đo phần mềm bởi nhiều lý do:

1. Để chỉ ra chất lượng phần mềm.

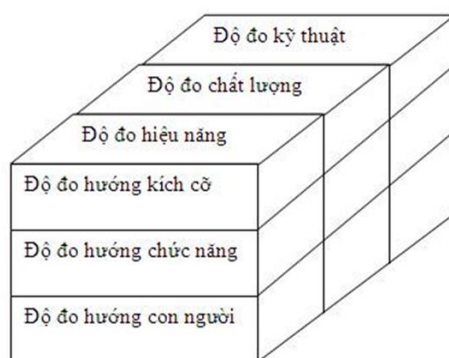
2. Để khẳng định hiệu suất của những người tạo ra sản phẩm.
3. Để khẳng định lợi ích (dưới dạng hiệu suất và chất lượng) mà phương pháp và công cụ kỹ nghệ phần mềm đem lại.
4. Để tạo ra một vạch ranh giới cho ước lượng.
5. Để giúp biện minh cho các yêu cầu về công cụ mới và việc huấn luyện bổ sung.

Ta có thể chia các cách đo trong thế giới vật lý theo hai cách: Đo trực tiếp (chẳng hạn như chiều dài chiếc then), đo gián tiếp (như chất lượng chiếc then được tạo ra, được đo bằng cách đếm số các then bị loại bỏ). Độ đo phần mềm cũng có thể phân loại tương tự.

Cách đo trực tiếp cho tiến trình kỹ nghệ phần mềm bao gồm việc đo về chi phí và công sức bỏ ra. Cách đo trực tiếp cho một sản phẩm bao gồm số dòng lệnh (LOC) được tạo ra, tốc độ thực hiện, kích cỡ bộ nhớ và những khiếm khuyết được báo cáo lại qua một thời kỳ nào đó. Việc đo gián tiếp sản phẩm bao gồm chức năng, chất lượng, độ phức tạp, tính hiệu quả, độ tin cậy, tính bảo trì và nhiều khả năng khác.

Chi phí và công sức cần cho việc xây dựng phần mềm, số dòng lệnh và những cách đo trực tiếp khác thì tương đối dễ thu thập mỗi khi đã thiết lập trước được các quy ước đặc biệt cho việc đo. Tuy nhiên, chất lượng và chức năng phần mềm hay tính hiệu quả và tính dễ bảo trì của nó thì khó xác định hơn và chỉ có thể đo một cách gián tiếp.

Chúng ta có thể phân loại được thêm nữa về lĩnh vực các độ đo phần mềm như trong hình sau:



**Hình 1.2 Phân loại độ đo**

Độ đo hiệu năng tập trung vào đầu ra của tiến trình kỹ nghệ phần mềm. Độ đo chất lượng đưa ra một chỉ dẫn về việc phần mềm tuân thủ chặt chẽ đến đâu đối với các yêu cầu tường minh và không tường minh của người dùng (tính thích hợp của phần mềm với việc sử dụng). Độ đo kỹ thuật tập trung vào đặc trưng của phần mềm như độ phức tạp logic, mức độ module) hơn là chú ý vào tiến trình qua đó phần mềm được phát triển.

Lưu ý rằng cũng có thể phát triển cách phân loại thứ hai. Độ đo kích cỡ được dùng để thu thập đầu ra và chất lượng kỹ nghệ phần mềm. Độ đo theo chức năng đưa ra cách đo gián tiếp và độ đo theo con người thì thu thập thông tin là cách thức để phát triển phần mềm máy

tính và cảm nhận của con người về tính hiệu quả của công cụ và phương pháp.

### 1.5.1 Độ đo theo kích cỡ

Độ đo phần mềm theo kích cỡ là cách đo trực tiếp cho phần mềm và tiến trình phân tích của nó. Nếu một tổ chức làm phần mềm duy trì các bản ghi đơn giản, thì có thể tạo ra một bảng các dữ liệu theo kích cỡ, như bảng sau:

**Bảng 1.3 Độ đo theo kích cỡ**

Dự án	Công	1000\$	KLOC	Trang tư liệu	Lỗi	Người
Aaa-01	24	168	12.1	365	29	3
Bbb-04	62	440	27.2	1224	86	5
Fff-05	43	314	20.2	1050	64	6

Bảng này liệt kê theo từng dự án phát triển phần mềm đã được hoàn thành trong vài năm qua và dữ liệu theo kích cỡ tương ứng cho dự án đó.

Từ dữ liệu thô trong bảng này, ta có thể phát triển một tập các độ đo chất lượng và hiệu năng theo kích cỡ cho từng dự án. Có thể tính ra giá trị trung bình cho mọi dự án:

Hiệu năng = KLOC/người-tháng

Chất lượng = khiếm khuyết/ KLOC

Bên cạnh đó có thể tính các độ đo đáng quan tâm khác:

Chi phí = \$/KLOC

Tư liệu = số trang tư liệu/KLOC

Các độ đo theo kích cỡ vẫn còn được bàn cãi rất nhiều và vẫn chưa được chấp nhận phổ biến như cách tốt nhất đo chất lượng phần mềm. Phần lớn các tranh luận xoay quanh việc dùng các dòng chương trình làm thước đo chính. Những người ủng hộ LOC cho rằng LOC là điều giả tạo của các dự án phát triển phần mềm, dễ dàng được đếm, rằng các mô hình ước lượng phần mềm hiện có đều dùng LOC hay KLOC làm cái vào chính, và rằng phần lớn các tài liệu và dữ liệu đều dựa trên LOC đã có sẵn. Mặt khác, những người phản đối thì nói rằng LOC phụ thuộc vào ngôn ngữ lập trình, rằng chúng ảnh hưởng tới chương trình thiết kế tốt nhưng ngắn hơn và rằng chúng không thể dễ dàng điều tiết cho các ngôn ngữ phi thủ tục và rằng việc dùng chúng trong ước lượng đòi hỏi một mức độ chi tiết thường khó đạt tới.

### 1.5.2. Độ đo theo điểm chức năng

Độ đo phần mềm theo điểm chức năng là cách đo gián tiếp cho phần mềm và tiến triển phát triển nó. Thay vì đếm LOC, độ đo theo chức năng tập trung vào chức năng hay tiện ích của chương trình. Độ đo điểm chức năng ban đầu được Albrecht đề nghị, người gợi ý ra một cách tiếp cận đo hiệu năng gọi là phương pháp điểm chức năng. Các điểm chức năng (Function Point- FP) được suy ra bằng cách dùng một quan hệ kinh nghiệm dựa trên các cách đo đếm được về lĩnh vực thông tin của phần mềm và các khẳng định về độ phức tạp phần mềm.

Điểm chức năng được tính bằng cách hoàn thiện bảng sau :

**Bảng 1.4 Tính độ đo điểm chức năng**

Tham biến độ đo	Số đếm	Nhân tố trọng số			
		Đơn giản	Trung bình	Phức tạp	
Số cái vào theo người dùng	x	3	4	6	=
Số cái ra theo người dùng	x	4	5	7	=
Số yêu cầu người dùng	x	3	4	6	=
Số các tệp	x	7	10	15	=
Số các giao diện ngoài	x	5	7	10	=
Số đếm tổng cộng					

Năm đặc trưng thông tin sẽ được xác định và số đếm được nêu ra tại các vị trí thích hợp. Các giá trị miền thông tin được xác định theo cách sau:

**Số thông tin đầu vào theo người dùng:** Phải tính các thông tin đầu vào mà từng người dùng cung cấp dữ liệu theo ứng dụng phân biệt cho phần mềm. Cần phải phân biệt các thông tin đầu vào với những câu hỏi thường được mang tính riêng biệt.

**Số thông tin đầu ra theo người dùng:** Cũng phải tính tới thông tin đầu ra mà từng người dùng cung cấp thông tin theo ứng dụng cho phần mềm. Trong hoàn cảnh này, thông tin đầu ra nói tới các báo cáo màn hình, thông báo lỗi. Từng khoản mục dữ liệu riêng lẻ bên trong một báo cáo không được tính riêng biệt.

**Số các câu hỏi của người dùng:** Câu hỏi được định nghĩa như một thông tin đầu vào trực tuyến trong việc sinh ra một đáp ứng ngay lập tức của phần mềm dưới dạng cái ra trực tuyến. Mỗi câu hỏi phân biệt phải được tính tới.

**Số các giao diện ngoài:** Mọi giao diện máy móc đọc được như các tệp dữ liệu trên băng hay đĩa) được dùng để truyền thông tin sang hệ thống khác đều cần phải tính tới.

Một khi dữ liệu trên đã được thu thập thì có thể gán cho từng số đếm một giá trị phức tạp. Các tổ chức sử dụng phương pháp điểm chức năng đang phát triển các tiêu chí để xác định xem liệu một mục tiêu đặc biệt là đơn giản, trung bình hay phức tạp. Tuy nhiên việc xác định độ phức tạp theo cách nào đó vẫn còn mang tính chủ quan nhiều.

Để tính các điểm chức năng người ta dùng mối quan hệ sau đây:

$$FP = \text{tổng số đếm} \times [0.65 + 0.01 \times \text{SUM}(Fi)]$$

với tổng số đếm là tổng của tất cả các mục FP có được trong bảng.  $Fi(i=1 \text{ tới } 14)$  là giá trị điều chỉnh độ phức tạp dựa trên việc đáp ứng các câu hỏi trong bảng 3.3. Các giá trị hằng trong phương trình trên và nhân tố trọng số được áp dụng cho các số đếm lĩnh vực thông tin được xác định theo kinh nghiệm.

Một khi đã tính được các điểm chức năng thì ta có thể dùng chúng theo cách tương tự như LOC để đo hiệu năng, phẩm chất và các thuộc tính khác của phần mềm trong số các yếu tố khác.

Hiệu năng = FP/ người-tháng

Phẩm chất = khiếm khuyết/ FP

Chi phí = \$/ FP

Tư liệu = Số trang tư liệu/ FP

[Tham khảo]

**Bảng 1.5 Tính điểm chức năng**

Tỷ lệ nhân tố theo thang từ 0 đến 5					
0	1	2	3	4	5
Không ảnh hưởng	Ngẫu nhiên	Đơn giản	Trung bình	Có ý nghĩa	Bản chất

Danh sách những câu hỏi thường dùng khi đánh giá phần mềm :

1. Hệ thống có đòi hỏi sao lưu và khôi phục tin cậy hay không?
2. Có đòi hỏi trao đổi dữ liệu không?
3. Có chức năng xử lý phân bố không?
4. Có đòi hỏi cao về làm việc tốt không?
5. Hệ thống có chạy trong một môi trường hiện có nặng về vận hành tiện tích không?
6. Hệ thống có đòi hỏi việc vào dữ liệu trực tuyến không?
7. Việc vào dữ liệu trực tuyến có đòi hỏi phải xây dựng giao tác đưa vào thông qua nhiều màn hình hay thao tác không?
8. Tập chính có phải cập nhật trực tuyến hay không?
9. Cái vào, cái ra, tệp, câu hỏi có phức tạp không?
10. Xử lý bên trong có phức tạp không?
11. Mã chương trình có được thiết kế để dùng lại không?
12. Việc chuyển đổi và cài đặt có được đưa vào trong thiết kế hay không?
13. Hệ thống có được thiết kế cho nhiều cài đặt trong các tổ chức khác nhau không?
14. Liệu ứng dụng có được thiết kế để làm thuận tiện cho việc thay đổi và dễ dàng cho người dùng?

### 1.5.3 Độ đo điểm chức năng mở rộng

Độ đo điểm chức năng ban đầu được thiết kế để ứng dụng cho các ứng dụng hệ thống thông tin nghiệp vụ. Để thích hợp với các ứng dụng này, chiều dữ liệu (giá trị miền thông tin đã đề cập ở trên) được nhấn mạnh, bỏ qua chiều chức năng và hành vi(điều khiển). Vì lý do này, mà độ đo điểm chức năng không thích hợp với các hệ thống kỹ nghệ và nhúng (các hệ

thống nhấn mạnh vào chức năng và điều khiển). Một mở rộng của độ đo điểm chức năng được đưa ra để giải quyết tình trạng trên.

Một mở rộng điểm chức năng gọi là điểm tính năng (feature points) là một tập con của độ đo điểm chức năng được áp dụng cho các ứng dụng hệ thống và công nghệ phần mềm. Độ đo điểm tính năng thích hợp với các ứng dụng có các giải thuật phức tạp. Các ứng dụng thời gian thực, điều khiển tiến trình và các ứng dụng nhúng có xu hướng có các thuật toán phức tạp do đó phải sử dụng điểm đặc tính năng.

Để tính điểm tính năng, lại phải tính đến các giá trị miền thông tin và đánh trọng số như đã mô tả ở trên. Bên cạnh đó, độ đo điểm tính năng còn tính đến đặc trưng phần mềm mới: thuật toán. Thuật toán được định nghĩa là “một vấn đề tính toán có giới hạn được bao hàm trong một chương trình máy tính đặc biệt”. Việc lấy nghịch đảo ma trận, giải mã một xâu bit, hay xử lý một ngắt tất cả đều là các thí dụ về thuật toán.

Một mở rộng điểm chức năng cho hệ thống thời gian thực và sản phẩm kỹ nghệ đã được Boeing phát triển. Cách tiếp cận của Boeing tích hợp hướng dữ liệu của phần mềm với hướng chức năng và điều khiển để cung cấp một độ đo hướng chức năng gọi là điểm chức năng 3D.

Hướng dữ liệu được tính giống như phần trên. Số các dữ liệu được giữ lại (cấu trúc dữ liệu bên trong chương trình, file) và các dữ liệu bên ngoài (đầu vào, đầu ra, các tham chiếu bên ngoài) được sử dụng cùng với các độ đo tính phức tạp nhận được từ đếm các hướng dữ liệu.

Hướng chức năng được tính bằng cách xem xét “số các phép toán bên trong được yêu cầu để biến đổi dữ liệu từ đầu vào thành đầu ra”. Để tính điểm chức năng 3D, ta coi sự biến đổi là một dãy các bước xử lý bị ràng buộc bởi các câu ngữ nghĩa (semantic statement). Nói chung, một sự biến đổi được hoàn thành với một thuật toán mà kết quả là chuyển dữ liệu đầu vào thành dữ liệu đầu ra. Các bước xử lý lấy dữ liệu từ file và đơn giản là đặt dữ liệu vào trong bộ nhớ chương trình không được coi là một sự biến đổi. Dữ liệu bản thân nó không được thay đổi theo bất kỳ cách cơ bản nào.

Mức độ phức tạp xác định cho mỗi sự biến đổi là một hàm của số bước xử lý và số các câu ngữ nghĩa điều khiển các bước xử lý. Bảng sau cung cấp một hướng dẫn cho sự thiết lập tính phức tạp cho hướng chức năng:

**Bảng 1.6: Tính độ đo điểm tính năng**

Tham biến độ đo	Số đếm	Trọng số	
Số cái vào theo người dùng	x	4	=
Số cái ra theo người dùng	x	5	=
Số yêu cầu người dùng	x	4	=
Số các tệp	x	7	=
Số các giao diện ngoài	x	7	=
Thuật toán	x	3	=
Số đếm tổng cộng			

Hướng điều khiển được tính bằng cách đếm số lượng biến đổi giữa các trạng thái.

Một trạng thái mô tả một cách ứng xử quan sát được từ bên ngoài và sự biến đổi xảy ra như khi có một số sự kiện gây ra cho hệ thống hay phần mềm thay đổi các ứng xử của nó.

Để tính độ đo điểm chức năng 3D, ta sử dụng biểu thức sau:

$$\text{Index} = I + O + Q + F + E + T + R$$

Trong đó I, O, Q, F, E, T, R là các giá trị trọng số phức tạp cho các thành phần: đầu vào, đầu ra, câu hỏi, cấu trúc dữ liệu bên trong, file ngoài, sự biến đổi và chuyển. Mỗi giá trị trọng số phức tạp này lại được tính bằng cách sử dụng biểu thức sau:

$$\text{Giá trị trọng số phức tạp} = Nil + NiaWia + NihWih$$

Với Nil, Nia, Nih là số đếm cho mỗi thành phần với mỗi mức độ của độ phức tạp (cao, thấp, trung bình) và Wil, Wia và Wih là các trọng số tương ứng. Toàn bộ cách tính điểm chức năng 3D được cho trong bảng sau:

**Bảng 1.7 Bảng quyết định độ phức tạp của các biến đổi cho điểm chức năng 3D**

Số câu ngữ nghĩa	1-5	6-10	11+
Số bước xử lý			
1-10	Thấp	Thấp	Trung bình
11-20	Thấp	Trung bình	Cao
21+	Trung bình	Cao	Cao

Cần chú ý rằng các điểm tính năng và các điểm chức năng và các điểm chức năng 3D biểu thị cho cùng một điều – “chức năng” hay “tiện ích” do phần mềm cung cấp. Trong thực tế, các cách đo này đều cho giá trị FP đối với các ứng dụng tính toán công nghệ quy ước hay hệ thống tin. Đối với các hệ thống thời gian thực phức tạp hơn, số đếm điểm tính năng thường giữa 20 và 35% cao hơn số đếm được xác định bằng cách dùng một mình điểm chức năng.

Độ đo điểm chức năng (hay độ đo điểm tính năng) như LOC vẫn còn đang được bàn cãi nhiều. Những người ủng hộ nói rằng FP là phụ thuộc vào ngôn ngữ lập trình, làm cho nó trở thành lý tưởng cho các ứng dụng có dùng các ngôn ngữ quy ước hay phi thủ tục; họ cũng nói rằng nó dựa trên dữ liệu và rất có thể dùng được từ rất sớm trong tiến hoá của một dự án, làm cho FP trở nên hấp dẫn hơn như một cách tiếp cận ước lượng. Những người phản đối lại nói rằng phương pháp này đòi hỏi một số khả năng thủ công trong đó tính toán nhiều dựa vào chủ quan chứ không phải khách quan, không dựa vào dữ liệu, rằng thông tin về lĩnh vực thông tin có thể khó thâm nhập, và rằng FP không có ý nghĩa vật lý trực tiếp, nó chỉ là con số.

## 1.6. Các ứng dụng phần mềm

Phần mềm có thể được áp dụng trong bất kỳ tình huống nào có một tập các bước thủ tục (như một thuật toán) đã được xác định trước (các ngoại lệ đáng chú ý với quy tắc này là các phần mềm hệ chuyên gia và các phần mềm mạng nơ ron). Nội dung thông tin và tính tất định là các nhân tố quan trọng trong việc xác định bản chất ứng dụng của phần mềm. Nội dung nói tới ý nghĩa và hình dạng của thông tin vào và ra. Tính tất định thông tin nói tới việc tiên đoán trước trật tự và thời gian của thông tin.

Theo một cách nào đó thì có khó khăn trong việc phát triển các phạm trù tổng quát có ý nghĩa đối với các ứng dụng phần mềm. Khi độ phức tạp phần mềm tăng lên, việc phân chia rõ rệt sẽ biến mất. Các lĩnh vực phần mềm sau chỉ ra sự trải rộng của các ứng dụng tiềm năng:

**Phần mềm hệ thống:** Phần mềm hệ thống là tập hợp các chương trình được viết để phục vụ cho chương trình khác. Phần mềm hệ thống (trình biên dịch, trình soạn thảo, các tiện ích quản lý tệp...) xử lý các cấu trúc thông tin phức tạp nhưng xác định. Các ứng dụng hệ thống khác (như thành phần hệ điều hành, bộ xử lý viễn thông) có dữ liệu chủ yếu không xác định. Trong cả hai trường hợp, lĩnh vực phần mềm hệ thống được đặc trưng chủ yếu bởi tương tác với hệ thống máy tính; sử dụng rất nhiều cho các hệ thống nhiều người dùng, thao tác tương tranh đòi hỏi lập lịch, dùng chung tài nguyên và các quản lý tiến trình phức tạp; cấu trúc dữ liệu phức tạp và nhiều giao diện ngoài.

**Phần mềm thời gian thực:** Phần mềm điều phối, phân tích, kiểm soát các sự kiện của thế giới thực khi chúng xuất hiện gọi là phần mềm thời gian thực. Các yếu tố của phần mềm thời gian thực bao gồm một thành phần thu thập dữ liệu để thu thập và định dạng thông tin từ môi trường bên ngoài, một thành phần phân tích biến đổi thông tin theo yêu cầu của ứng dụng, một thành phần kiểm soát/đưa ra để đáp ứng với môi trường bên ngoài, một thành phần điều phối để điều hoà các thành phần khác sao cho có thể duy trì đáp ứng thời gian thực điển hình. Hệ thống thời gian thực phải đáp ứng trong những ràng buộc thời gian chặt chẽ.

**Phần mềm nghiệp vụ:** Xử lý thông tin nghiệp vụ là lĩnh vực ứng dụng phần mềm lớn nhất. Các hệ thống rời rạc (như tính lương, kế toán thu/chi, quản lý kho) đã tiến hoá thành



các phần mềm hệ thông tin quản lý thâm nhập vào một hay nhiều cơ sở dữ liệu lớn chứa thông tin nghiệp vụ. Những ứng dụng trong lĩnh vực này cấu trúc lại dữ liệu hiện có theo cách thuận tiện cho các thao tác nghiệp vụ hay ra quyết định quản lý. Bên cạnh các ứng dụng xử lý dữ liệu quy ước, các ứng dụng phần mềm nghiệp vụ còn bao gồm cả các tính toán tương tác (như xử lý các giao dịch cho các điểm bán hàng).

**Phần mềm khoa học và công nghệ:** Phần mềm khoa học và công nghệ được đặc trưng bởi các thuật toán ‘máy nghiền số’. Các ứng dụng dao động từ thiên văn cho đến núi lửa, từ phân tích căng thẳng về ô tô cho đến sự biến động quỹ đạo tàu con thoi, từ sinh học phân tử đến chế tạo tự động. Tuy nhiên, những ứng dụng mới bên trong lĩnh vực khoa học và công nghệ đang di chuyển nhanh ra khỏi các thuật ngữ quy ước. Thiết kế có sự trợ giúp của máy tính (CAD), mô phỏng hệ thống và những ứng dụng tương tác khác đã bắt đầu kế tục các đặc trưng thời gian thực và thậm chí cả phần mềm hệ thống.

**Phần mềm nhúng:** Các sản phẩm thông minh đã trở nên thông dụng. Phần mềm nhúng nằm trong bộ nhớ chỉ đọc và được dùng để điều khiển các sản phẩm và các hệ thống cho người tiêu dùng và thị trường công nghiệp. Phần mềm nhúng có thể thực hiện các chức năng rất giới hạn như điều khiển bàn phím cho lò vi sóng hay đưa ra các khả năng điều khiển và vận hành có nghĩa như chức năng số hoá trong ô tô, kiểm tra xăng, hiển thị bảng đồng hồ, hệ thống phanh.

**Phần mềm máy tính cá nhân:** Xử lý văn bản, đồ họa máy tính, quản trị cơ sở dữ liệu, các ứng dụng tài chính cá nhân và nghiệp vụ, mạng bên ngoài hay thâm nhập cơ sở dữ liệu chỉ là một số lĩnh vực kể ra được trong hàng trăm ứng dụng. Trong thực tế, phần mềm máy tính cá nhân tiếp tục biểu thị cho một số thiết kế giao diện người - máy được cải tiến nhiều nhất trong mọi phần mềm.

**Phần mềm trí tuệ nhân tạo:** Phần mềm trí tuệ nhân tạo dùng các thuật toán phi số để giải quyết các vấn đề phức tạp mà tính toán hay phân tích trực tiếp không quản lý nổi. Hiện nay lĩnh vực trí tuệ nhân tạo hoạt động mạnh nhất là các hệ chuyên gia, cũng còn gọi là các hệ cơ sở tri thức. Tuy nhiên, các lĩnh vực áp dụng khác của phần mềm AI còn là nhận dạng (hình ảnh và tiếng nói), chứng minh định lý và trò chơi. Trong những năm gần đây, một nhánh mới của phần mềm AI, gọi là mạng nơ ron nhân tạo, đã phát triển. Mạng nơ ron mô tả cấu trúc của việc xử lý trong bộ óc và cuối cùng dẫn đến lớp phần mềm mới có thể nhận dạng các mẫu phức tạp và học từ “kinh nghiệm quá khứ”.

### **Câu hỏi:**

1. Phần mềm là gì, các khái niệm trong phần mềm ?
2. Các đặc tính của phần mềm. Việc phân loại phần mềm có đem lại lợi ích gì trong việc phát triển phần mềm.
3. Phân tích các đặc điểm của các mô hình phát triển phần mềm. Khi xây dựng một phần mềm, người ta dựa vào các yếu tố nào để chọn mô hình phát triển. (Vòng đời?)

4. Hãy chỉ ra các lợi ích khi có được người sử dụng tham gia vào quá trình phát triển phần mềm.

5. Liệt kê các ví dụ phần mềm cho từng loại mà bạn biết.

## **Chương 2. KHỦNG HOẢNG PHẦN MỀM**

Mục tiêu của công nghệ phần mềm là sản xuất ra những phần mềm tốt, có chất lượng cao. Nhưng cũng như những ngành công nghiệp khác phần mềm cũng gặp những khủng hoảng nhất định. Chương này nhằm tìm hiểu sơ lược những khủng hoảng xảy ra trong phần mềm, và các vấn đề trong sản xuất phần mềm.

### **2.1. Khủng hoảng phần mềm**

Cơn khủng hoảng phần mềm là kết quả trực tiếp từ sự ra đời của phần cứng máy tính mới dựa trên công nghệ mạch tích hợp. Sức mạnh của nó đã biến những ứng dụng máy tính tưởng chừng như không thể cho tới nay lại trở thành những công việc có khả năng thực thi. Kết quả là nhưng hệ thống phần mềm ngày càng có tầm quan trọng lớn hơn và phức tạp hơn những hệ thống phần mềm trước đó.

Vậy khủng hoảng phần mềm là gì ?

Vào tháng 10/1968 tại hội nghị của NATO các chuyên gia phần mềm đã đưa ra thuật ngữ “Khủng hoảng phần mềm” (software crisis). Qua hàng chục năm thuật ngữ này vẫn được sử dụng và ngày càng mang tính cấp bách.

Khủng hoảng là gì ? theo tra cứu của từ điển Webster [Webster’s Dict.]

Khủng hoảng là :

– Điểm ngoặt trong tiến trình của bất kỳ cái gì; thời điểm, giai đoạn hoặc biến cố quyết định hay chủ chốt

– Điểm ngoặt trong quá trình diễn biến bệnh khi trở nên rõ ràng bệnh nhân sẽ sống hay chết.

Khủng hoảng phần mềm là: Sự day dứt kinh niên( kéo dài theo thời gian hoặc thường tái diễn, liên tục không kết thúc) gặp phải trong phát triển phần mềm máy tính, như :

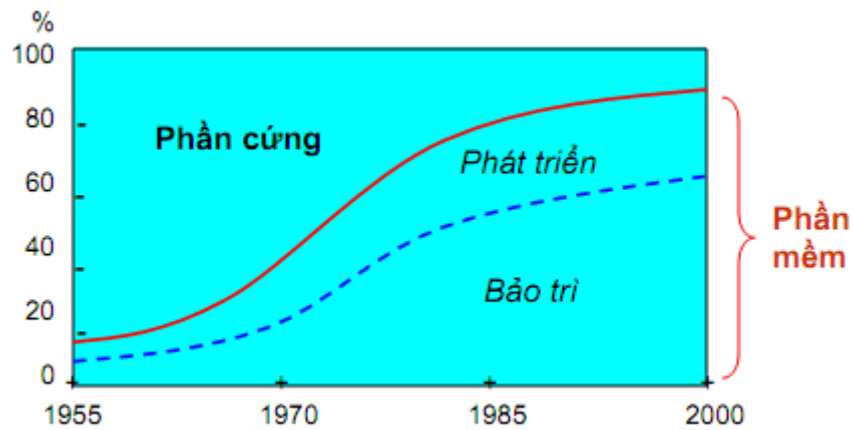
- Phải làm thế nào với việc giảm chất lượng vì những lỗi tiềm tàng có trong phần mềm?
- Phải xử lý ra sao khi bảo dưỡng phần mềm đã có?
- Phải giải quyết như thế nào khi thiếu kỹ thuật viên phần mềm
- Phải làm thế nào khi có yêu cầu phát triển theo quy cách mới xuất hiện
- Phải xử lý ra sao khi sự cố phần mềm gây ra những vấn đề xã hội.

#### **2.1.1 Một số yếu tố dẫn đến sự khủng hoảng trong phần mềm:**

Phần mềm càng lớn sẽ kéo theo sự phức tạp hóa và tăng chi phí phát triển

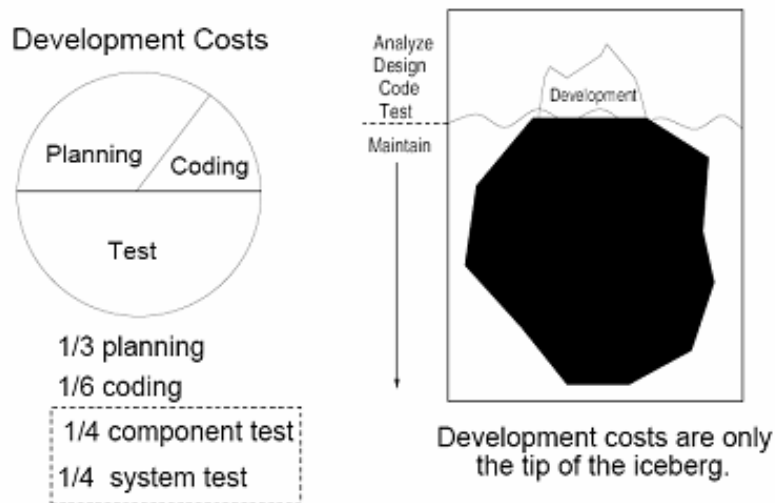
Đôi vai trò giá thành giữa phần mềm với phần cứng

Công sức cho bảo trì càng tăng thì chi phí cho tồn đọng (backlog) càng lớn  
 Nhân lực chưa đáp ứng được cho nhu cầu phần mềm  
 Những vấn đề của phần mềm gây ra những vấn đề xã hội



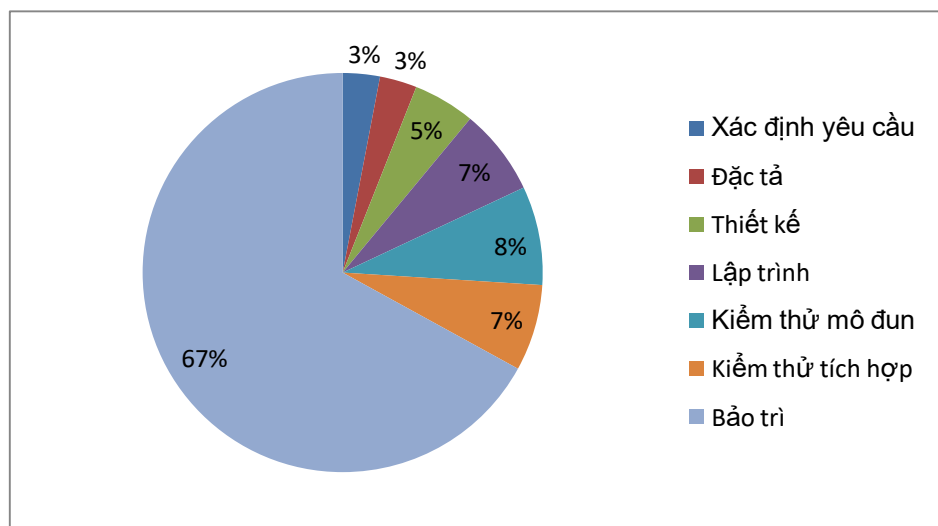
**Hình 2.1: So sánh chi phí giữa phần cứng và phần mềm**

Chi phí cho các pha và nguyên lý tảng băng trôi của phát triển phần mềm

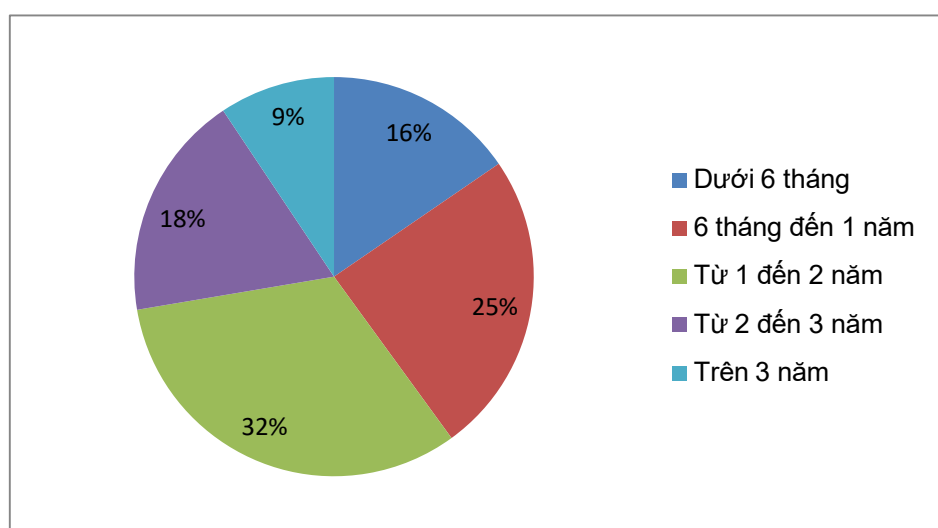


(Chi phí phát triển chỉ là phần đỉnh của tảng băng trôi)

**Hình 2.2: Nguyên lý tảng băng trôi của phát triển phần mềm.**



**Hình 2.3: So sánh chi phí cho các pha.**



**Hình 2.4: Backlog Nhật Bản năm 1985.**

## 2.2. Những vấn đề trong sản xuất phần mềm

Ngày nay, hầu như mọi quốc gia trên thế giới đều phụ thuộc vào các hệ thống phức tạp hoạt động dựa trên máy tính. Các cơ sở hạ tầng và lợi ích quốc gia đều dựa trên những hệ thống trên và hầu hết các sản phẩm điện đều bao gồm một máy tính và một phần mềm điều khiển. Sản xuất công nghiệp và sự phân phối hàng hóa đã được tin học hóa hoàn toàn, ví dụ như là hệ thống tài chính. Bởi vậy, việc sản xuất và bảo trì phần mềm với chi phí hiệu quả là rất cần thiết cho chiến lược phát triển của mỗi quốc gia và của nền kinh tế toàn cầu.

Công nghệ phần mềm là một ngành công nghệ có kỷ luật mà tiêu điểm của nó là chi phí phát triển hiệu quả của những hệ thống phần mềm chất lượng cao. Phần mềm là một cái gì đó trừu tượng và không cầm nắm được. Nó không bị gò ép bởi các tài liệu, hay bị quản lý bởi các đạo luật cũng như các quá trình sản xuất. Nói theo cách nào đó, chính điều này làm đơn giản đi công nghệ phần mềm bởi vì không hề có bất kì một giới hạn nào cho tiềm năng

của phần mềm. Tuy nhiên, sự thiếu đi những gò ép tự nhiên cũng có nghĩa là phần mềm có thể dễ dàng trở nên cực kì phức tạp và do đó rất khó để có thể hiểu được.

Khái niệm về công nghệ phần mềm lần đầu tiên được đưa ra vào năm 1968 tại một cuộc hội thảo mà sau này được gọi là “con khủng hoảng phần mềm”. Con khủng hoảng phần mềm này cũng là kết quả trực tiếp từ sự ra đời của phần cứng máy tính mới dựa trên công nghệ mạch tích hợp. Sức mạnh của nó đã biến những ứng dụng máy tính tưởng chừng như không thể cho tới nay lại trở thành những công việc có khả năng thực thi. Kết quả là những hệ thống phần mềm ngày càng có tầm quan trọng lớn hơn và phức tạp hơn những hệ thống phần mềm trước đó.

Những kinh nghiệm đầu tiên trong việc xây dựng hệ thống này đã chỉ ra rằng việc phát triển phần mềm không theo quy định là không tốt. Những dự án chính có khi muộn hàng năm trời. Các phần mềm có chi phí lớn hơn nhiều so với dự tính, không đáng tin cậy, khó để bảo trì và thực thi rất kém. Phát triển phần mềm đã rơi vào tình trạng khủng hoảng. Giá của phần cứng ngày càng giảm trong khi giá phần mềm thì tăng lên nhanh chóng. Những công nghệ mới và những phương thức mới đã trở nên hết sức cần thiết để điều chỉnh tính phức tạp vốn có trong những hệ thống phần mềm lớn.

Những công nghệ này đã trở thành một phần của công nghệ phần mềm và được sử dụng rộng rãi ngày nay. Tuy nhiên, do khả năng sản xuất phần mềm của con người ngày càng tăng lên, do đó cũng nảy sinh những sự phức tạp của hệ thống phần mềm mà chúng ta cần đến. Những công nghệ mới kết quả từ sự hội tụ của hệ thống truyền thông và máy tính cùng với những giao diện đồ họa người dùng phức tạp tạo ra nhiều yêu cầu mới đối với những kỹ sư phần mềm. Một số công ty do vẫn không chịu tuân theo những kỹ thuật của công nghệ phần mềm một cách hiệu quả nên nhiều dự án vẫn sản xuất ra những phần mềm không thực tế, bị giảm doanh thu và phá sản.

Chúng ta nghĩ rằng chúng ta đã tạo ra một quá trình mạnh mẽ kể từ năm 1968 và sự phát triển của công nghệ phần mềm đã cải thiện một cách rõ rệt các phần mềm của chúng ta. Chúng ta đã có được nhiều hiểu biết hơn về những hoạt động liên quan đến phát triển phần mềm. Chúng ta cũng đã phát triển những phương thức hiệu quả trong cách thiết kế và thực thi phần mềm. Những chỉ dẫn và công cụ mới góp phần giảm bớt công sức để sản xuất ra những hệ thống lớn và phức tạp.

Chúng ta có thể thấy được những vấn đề khó khăn trong sản xuất phần mềm tự chung như sau :

- Không có phương pháp mô tả rõ ràng định nghĩa yêu cầu của người dùng (khách hàng), sau khi bàn giao sản phẩm dễ phát sinh những trục trặc (troubles).
- Với những phần mềm quy mô lớn, tư liệu đặc tả đã cố định thời gian dài, do vậy khó đáp ứng nhu cầu thay đổi của người dùng một cách kịp thời trong thời gian đó.

- Nếu không có phương pháp luận thiết kế nhất quán mà thiết kế theo cách riêng (của công ty, nhóm), thì sẽ dẫn đến suy giảm chất lượng phần mềm (do phụ thuộc quá nhiều vào con người).
- Nếu không có chuẩn về làm tư liệu quy trình sản xuất phần mềm, thì những đặc tả không rõ ràng sẽ làm giảm chất lượng phần mềm.
- Nếu không kiểm thử tính đúng đắn của phần mềm ở từng giai đoạn mà chỉ kiểm ở giai đoạn cuối và phát hiện ra lỗi, thì thường bàn giao sản phẩm không đúng hạn.
- Nếu coi trọng việc lập trình hơn khâu thiết kế thì thường dẫn đến làm giảm chất lượng phần mềm.
- Nếu coi thường việc tái sử dụng phần mềm (software reuse), thì năng suất lao động sẽ giảm.
- Phần lớn trong quy trình phát triển phần mềm có nhiều thao tác do con người thực hiện, do vậy năng suất lao động thường bị giảm.
- Không chứng minh được tính đúng đắn của phần mềm, do vậy độ tin cậy của phần mềm sẽ giảm.
- Chuẩn về một phần mềm tốt không thể đo được một cách định lượng, do vậy không thể đánh giá được một hệ thống đúng đắn hay không.
- Khi đầu tư nhân lực lớn vào bảo trì sẽ làm giảm hiệu suất lao động của nhân viên.
- Công việc bảo trì kéo dài làm giảm chất lượng của tư liệu và ảnh hưởng xấu đến những việc khác.

Chúng ta biết rằng ngày nay không chỉ có một cách tiếp cận lý tưởng đối với công nghệ phần mềm. Sự đa dạng và nhiều kiểu khác nhau của các hệ thống và các tổ chức sử dụng những hệ thống đồng nghĩa với việc cần phải có một sự đa dạng những cách tiếp cận đến công nghệ phần mềm. Tuy nhiên, những chỉ dẫn cốt yếu và tổ chức hệ thống chính là nền tảng cho tất cả những công nghệ này, và đây mới là cốt lõi của kỹ nghệ phần mềm.

Những kỹ sư phần mềm có thể tự hào về những thành tựu mà họ đã đạt được. Không có những phần mềm phức tạp chúng ta không thể nào thám hiểm được không gian, không thể có Internet và hệ thống viễn thông hiện đại, và tất cả các chuyến đi sẽ trở nên nhiều nguy hiểm và đắt đỏ hơn. Công nghệ phần mềm đã có một sự đóng góp rất to lớn, và chúng ta nhận thấy rằng, bởi tính kỷ lưỡng có kỷ luật, những đóng góp của nó trong thế kỷ 21 này sẽ còn to lớn hơn nhiều.

### **Câu hỏi:**

1. Khủng hoảng phần mềm là gì ?
2. Các yếu tố dẫn đến khủng hoảng của phần mềm. Việc hiểu rõ các yếu tố dẫn đến khủng hoảng phần mềm có đem lại lợi ích gì trong việc phát triển phần mềm?
3. Phân tích các khó khăn trong sản xuất phần mềm?

4. Hãy áp dụng kiến thức trong bài học này để lập ra bảng các yếu tố khó khăn gặp phải khi làm dự án nhóm.

### **Chương 3. CÔNG NGHỆ PHẦN MỀM**

#### **Mục tiêu:**

Mục tiêu của công nghệ phần mềm là sản xuất ra những phần mềm tốt, có chất lượng cao. Chương này cung cấp cho chúng ta kiến thức về lịch sử phát triển của công nghệ phần mềm (CNPM), hiểu được CNPM là gì, hiểu được vòng đời của phần mềm và quy trình phát triển phần mềm.

#### **3.1. Lịch sử phát triển ngành công nghiệp phần mềm**

Sự tiến hóa của phần mềm gắn liền với sự tiến hóa của phần cứng và có thể chia làm 4 giai đoạn:

##### ***Những năm đầu (từ 1950 đến 1960):***

Giai đoạn này phần cứng thay đổi liên tục, số lượng máy tính rất ít và phần lớn mỗi máy đều được đặt hàng chuyên dụng cho một ứng dụng đặc biệt.

Phương thức chính là xử lý theo lô (batch), tức là “gói” các chương trình có sử dụng kết quả của nhau lại thành một khối để tăng tốc độ thực hiện.

Thời kỳ này lập trình máy tính được coi là nghệ thuật “theo bản năng”, chưa có phương pháp hệ thống. Việc phát triển phần mềm chưa được quản lý.

Môi trường lập trình có tính chất cá nhân; thiết kế, tiến trình phần mềm không tường minh, thường không có dữ liệu. Sản xuất có tính đơn chiếc, theo đơn đặt hàng. Người lập trình thường là người sử dụng và kiêm cả việc bảo trì và sửa lỗi.

##### ***Thời kỳ trải rộng từ những năm 1960 đến giữa những năm 1970.***

Các hệ thống đa nhiệm, đa người sử dụng (ví dụ: Multics, Unix,...) xuất hiện dẫn đến khái niệm mới về tương tác người máy. Kỹ thuật này mở ra thế giới mới cho các ứng dụng và đòi hỏi mức độ tinh vi hơn cho cả phần mềm và phần cứng.

Nhiều hệ thống thời gian thực với các đặc trưng thu thập, phân tích và biến đổi dữ liệu từ nhiều nguồn khác nhau và phản ứng (xử lý, tạo output) trong một khoảng thời gian nhất định xuất hiện.

Tiến độ lưu trữ trực tuyến làm xuất hiện thế hệ đầu tiên của hệ quản trị CSDL.

Số lượng các hệ thống dựa trên máy tính phát triển, nhu cầu phân phối mở rộng thư viện phần mềm phát triển, quy mô phần mềm ngày càng lớn làm nảy sinh nhu cầu sửa chữa khi gặp lỗi, cần sửa đổi khi người dùng có yêu cầu hay phải thích nghi với những thay đổi của môi trường phần mềm (phần cứng, hệ điều hành, chương trình dịch mới). Công việc bảo trì phần mềm dần dần tiêu tốn nhiều công sức và tài nguyên đến mức báo động.

Năm 1968: Tại Tây Đức, Hội nghị khoa học của NATO đã đưa ra từ “Software Engineering”. Bắt đầu bàn luận về không khoảng phần mềm và xu hướng hình thành CNPM như một chuyên môn riêng.

Nửa cuối 1960: IBM đưa ra chính sách phân biệt giá cả giữa phần cứng và phần mềm. Từ đó, ý thức về phần mềm ngày càng cao. Bắt đầu những nghiên cứu cơ bản về phương pháp luận lập trình.

### ***Thời kỳ từ giữa những năm 1970 đến đầu những năm 1990.***

Hệ thống phân tán (bao gồm nhiều máy tính, mỗi máy thực hiện một chức năng và liên lạc với các máy khác) xuất hiện làm tăng quy mô và độ phức tạp của phần mềm ứng dụng trên chúng.

Mạng toàn cục và cục bộ, liên lạc số giải thông cao phát triển mạnh làm tăng nhu cầu thâm nhập dữ liệu trực tuyến, nảy sinh yêu cầu lớn phát triển phần mềm quản lý dữ liệu.

Công nghệ chế tạo các bộ vi xử lý tiên bộ nhanh khiến cho máy tính cá nhân, máy trạm để bàn, và các thiết bị nhúng (dùng cho điều khiển trong robot, ô tô, thiết bị y tế, đồ điện gia dụng,...) phát triển mạnh khiến cho nhu cầu về phần mềm tăng nhanh.

Thị trường phần cứng đi vào ổn định, chi phí cho phần mềm tăng nhanh và có khuynh hướng vượt chi phí mua phần cứng.

Nửa đầu những năm 1970: Nhằm nâng cao chất lượng phần mềm, không chỉ có các nghiên cứu về lập trình, kiểm thử, mà có cả những nghiên cứu đảm bảo tính tin cậy trong quy trình sản xuất phần mềm. Kỹ thuật: lập trình cấu trúc hóa, lập trình môđun, thiết kế cấu trúc hóa...

Giữa những năm 1970: Hội nghị quốc tế đầu tiên về CNPM được tổ chức (1975): International Conference on SE (ICSE).

Nửa sau những năm 1970: Quan tâm đến mọi pha trong quy trình phát triển phần mềm, nhưng tập trung chính ở những pha đầu. ICSE tổ chức lần 2, 3 và 4 vào 1976, 1978 và 1979.

- Nhật Bản có “Kế hoạch phát triển kỹ thuật sản xuất phần mềm” từ năm 1981
- Cuộc “cách tân sản xuất phần mềm” đã bắt đầu trên phạm vi các nước công nghiệp.

Nửa đầu những năm 1980: Trình độ học vấn và ứng dụng CNPM được nâng cao, các công nghệ được chuyển vào thực tế. Xuất hiện các sản phẩm phần mềm và các công cụ khác nhau làm tăng năng suất sản xuất phần mềm đáng kể.

- ICSE tổ chức lần 5 và 6 năm 1981 và 1982 với trên 1000 người tham dự mỗi năm.
- Nhật Bản sang “Kế hoạch phát triển các kỹ thuật bảo trì phần mềm” (1981-1985).

### ***Thời kỳ sau 1990:***

Công nghệ hướng đối tượng là cách tiếp cận mới đang nhanh chóng thay thế nhiều cách tiếp cận phát triển phần mềm truyền thống trong các lĩnh vực ứng dụng.



Sự phát triển của Internet làm cho người dùng máy tính tăng lên nhanh chóng, nhu cầu phần mềm ngày càng lớn, quy mô và độ phức tạp của những hệ thống phần mềm mới cũng tăng đáng kể.

Phần mềm trí tuệ nhân tạo ứng dụng các thuật toán phi số như hệ chuyên gia, mạng nơ ron nhân tạo được chuyển từ phòng thí nghiệm ra ứng dụng thực tế mở ra khả năng xử lý thông tin và nhận dạng kiểu con người.

Chất lượng phần mềm tập trung chủ yếu ở tính năng suất, độ tin cậy và tính bảo trì. Nghiên cứu hỗ trợ tự động hóa sản xuất phần mềm.

- Nhật Bản có “Kế hoạch hệ thống công nghiệp hóa sản xuất phần mềm”(SIGMA: Software Industrialized Generator & Maintenance Aids, 1985-1990).
- Nhiều trung tâm, viện nghiên cứu CNPM ra đời. Các trường đưa vào giảng dạy SE.

### ***Hiện nay:***

Công nghiệp hóa sản xuất phần mềm bằng cách đưa những kỹ thuật công nghệ học (Engineering techniques) thành cơ sở khoa học của CNPM.

Thế chế hóa lý luận trong sản xuất phần mềm và ứng dụng những phương pháp luận một cách nhất quán.

Tăng cường nghiên cứu và tạo công cụ trợ giúp sản xuất phần mềm.

### **3.2. Sự phát triển của các phương pháp thiết kế phần mềm**

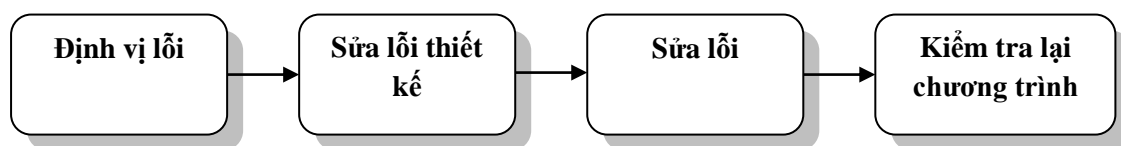
Một cách tiếp cận đối lập dựa trên việc tạo ra những mẫu đồ họa của hệ thống bằng cách dùng phương pháp cấu trúc, và trong một số trường hợp có thể tự động tạo mã từ những mô hình này. Phương pháp cấu trúc được phát minh vào những năm 1970 để ủng hộ thiết kế hướng vào chức năng (Constantine and Yourdon, 1979, Gane and Sarson, 1979). Nhiều phương pháp cạnh tranh ủng hộ thiết kế hướng vào đối tượng được đề xuất (Robinson, 1992; Booch, 1994) và chúng đã được hợp nhất vào những năm 1990 để cho ra đời Ngôn ngữ mô phỏng hợp nhất (Unified Modeling Language – UML) và quy trình thiết kế hợp nhất (Rumbaugh, 1991; Booch, 1999; Rumbaugh, 1999a; Rumbaugh, 1999b).

Một phương pháp cấu trúc bao gồm một mẫu về quá trình thiết kế, hệ thống ký hiệu để trình bày thiết kế, định dạng chuẩn, những quy tắc và hướng dẫn thiết kế. Phương pháp này có thể đáp ứng cho một vài hoặc tất cả những mô hình hệ thống dưới đây:

1. Một mô hình đối tượng cho biết những lớp đối tượng được sử dụng trong hệ thống và độ tin cậy của chúng.
2. Một mô hình dãy cho biết bằng cách nào những đối tượng trong hệ thống tương tác được với nhau khi hệ thống đang hoạt động.
3. Một mô hình về trạng thái quá độ cho biết trạng thái của hệ thống và nguyên nhân tạo ra sự quá độ từ trạng thái này đến trạng thái khác.

4. Một mô hình cấu trúc nơi những bộ phận cấu thành hệ thống và sự liên kết, thống nhất giữa chúng được chứng minh, xác nhận.

5. Một mô hình luồng dữ liệu nơi mà hệ thống được mô hình hoá thông qua sự thay đổi dữ liệu trong quá trình hoạt động. Mẫu này thường không được sử dụng trong những phương pháp hướng đối tượng nhưng lại thường xuyên được sử dụng trong thiết kế thời gian thực và hệ thống kinh doanh.



**Hình 3.1 Quá trình chỉnh sửa lỗi**

Trong thực tế, phương pháp cấu trúc thật sự là những hệ thống ký hiệu chuẩn mực và những biểu hiện cho sự hoạt động có hiệu quả. Một thiết kế hợp lý có thể ra đời từ việc làm theo những phương pháp này và áp dụng những sự chỉ dẫn. Việc quyết định phân chia hệ thống và khẳng định rằng thiết kế đã nắm bắt được những đặc tả hệ thống một cách thích đáng luôn đòi hỏi người thiết kế phải có khả năng sáng tạo cao. Việc nghiên cứu những nhà thiết kế trên cơ sở quan sát thực nghiệm (Bansler and Bodker, 1993) đã chỉ ra rằng họ hiếm khi làm theo những phương pháp trên một cách đơn thuần. Họ chỉ chọn lọc những sự chỉ dẫn từ những tình huống cục bộ.

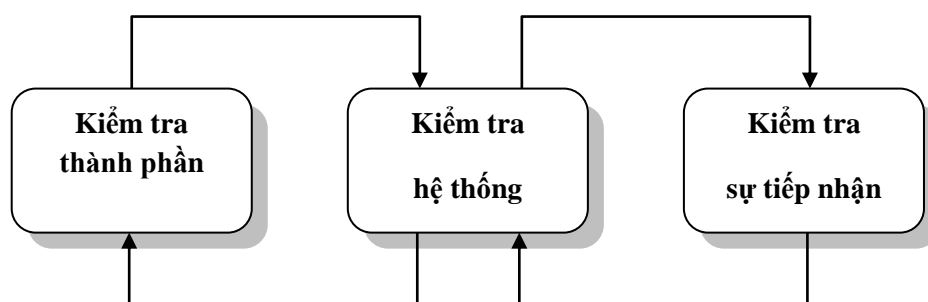
Việc phát triển một chương trình để đưa một hệ thống vào sử dụng luôn theo sau quá trình thiết kế hệ thống. Mặc dù một vài chương trình, chẳng hạn như các hệ thống đề cao tính an toàn, thường được thiết kế chi tiết trước khi việc thực hiện bắt đầu, những khâu cuối của thiết kế và việc phát triển chương trình nói chung là thường được xen kẽ. Công cụ CASE có thể được sử dụng để tạo nên khung chương trình từ bản thiết kế. Nó bao gồm mã định dạng và thi hành giao diện, và trong một số trường hợp người ta chỉ cần bổ sung một số chi tiết cho sự hoạt động của mỗi bộ phận cấu thành chương trình.

Việc lập trình là một hoạt động cá nhân và thường không có quá trình nào kèm theo. Một vào nhà lập trình thường bắt đầu với những bộ phận mà họ hiểu rõ, phát triển chúng rồi chuyển sang những bộ phận mà họ ít am hiểu hơn. Những người khác thì lại có cách tiếp cận ngược lại, họ để lại những phần quen thuộc bởi họ biết làm thế nào để phát triển chúng. Một vài nhà phát triển chương trình thích định dạng dữ liệu trước sau đó dùng chúng để phát triển chương trình; những người khác lại bỏ lại những dữ liệu không rõ ràng lâu nhất có thể.

Thông thường, những nhà lập trình kiểm tra mã mà họ vừa phát triển. Điều này thường để lộ ra những khuyết điểm phải bị loại bỏ khỏi chương trình. Nó được gọi là “sự gỡ rối”. Kiểm tra khuyết điểm và loại bỏ chúng là hai quá trình khác nhau. Việc kiểm tra xác minh sự tồn tại của những thiếu sót. Việc loại bỏ lại liên quan đến việc định vị và sửa chữa những thiếu sót này.

Hình 3.1 minh họa quá trình sửa lỗi này. Lỗi trong mã cần được định vị và chương trình được thay đổi để đáp ứng những yêu cầu. Việc kiểm tra sau đó phải được làm lại để chắc chắn rằng việc thay đổi là đúng đắn. Do đó quá trình loại bỏ là một phần của việc kiểm tra và phát triển phần mềm.

Khi loại bỏ sai sót chúng ta tạo ra những giả thuyết về những hoạt động có thể theo dõi được của chương trình, sau đó chúng ta kiểm tra những giả thuyết này với hy vọng tìm ra những sai sót làm cho đầu ra trở nên bất bình thường. Kiểm tra những giả thuyết này có thể kéo theo việc truy vết mã chương trình. Chúng ta có thể tạo ra những bài kiểm tra theo từng tình huống để khoanh vùng vấn đề. Công cụ loại bỏ sai sót tương tác có thể chỉ ra những giá trị trung bình của những biến số của chương trình và những dấu vết của sự trình bày đã được thể hiện. Công cụ này có thể rất hữu ích cho quá trình loại bỏ sai sót.



**Hình 3.2 Quá trình kiểm tra**

### **3.3. Định nghĩa công nghệ phần mềm**

Công nghệ phần mềm là một ngành công nghệ tập trung hướng vào các sản phẩm phần mềm ngay từ những bước ban đầu về chỉ định kỹ thuật hệ thống cho đến quá trình bảo trì hệ thống sau khi nó đã được đưa vào sử dụng. Trong định nghĩa này chúng ta thấy có hai điểm mấu chốt :

1. *Ngành công nghệ* - Người kỹ sư làm cho mọi thứ làm việc. Họ áp dụng những lý thuyết, những phương thức và những công cụ mà thích hợp với những công việc này, nhưng họ sử dụng chúng một cách có chọn lựa và luôn luôn cố gắng để tìm ra lời giải cho mọi vấn đề thậm chí ngay cả khi không hề có những phương thức hay lý thuyết có thể áp dụng được. Các kỹ sư cũng nhận ra rằng họ phải làm việc với sự gò ép về tài chính và tổ chức. Chính vì vậy tìm kiếm ra những cách giải quyết ngay trong những sự gò ép này.
2. *Toàn bộ tập trung hướng vào sản phẩm phần mềm* – Công nghệ phần mềm không chỉ quan tâm đến những tiến trình công nghệ của sự phát triển phần mềm mà nó còn quan tâm đến những công việc như quản lý dự án phần mềm và sự phát triển những công cụ, phương thức cùng với những lý thuyết hỗ trợ cho sản phẩm phần mềm.

Nhìn chung, các kỹ sư phần mềm chấp nhận cách tiếp cận có tổ chức và hệ thống đối với công việc của họ, bởi vì đây là cách làm hiệu quả nhất để sản xuất ra những phần mềm có

chất lượng cao. Tuy nhiên kỹ nghệ là tất cả mọi thứ về việc chọn lựa ra những phương thức thích hợp nhất cho một tập các tình huống và một cách tiếp cận sáng tạo hơn, ít thủ tục hơn để trong sự phát triển trong một vài tình huống có thể hiệu quả hơn. Sự phát triển ít thủ tục hơn đặc biệt thích hợp cho sự phát triển những hệ thống dựa vào cơ sở Web cái mà yêu cầu một hỗn hợp các phần mềm và những kỹ năng thiết kế đồ họa.

### **3.4. Vòng đời của sản phẩm phần mềm**

Vòng đời phần mềm là thời kỳ tính từ khi phần mềm được sinh (tạo) ra cho đến khi chết đi (từ lúc hình thành đáp ứng yêu cầu, vận hành, bảo dưỡng cho đến khi loại bỏ không dùng).

Quy trình phần mềm (vòng đời phần mềm) được phân chia thành các pha chính: phân tích, thiết kế, chế tạo, kiểm thử, bảo trì. Biểu diễn các pha có khác nhau theo từng người.

#### **Công nghệ và phân tích hệ thống (xác định yêu cầu hệ thống).**

Công nghệ và phân tích hệ thống bao gồm việc thu thập yêu cầu ở mức hệ thống với một lượng nhỏ thiết kế và phân tích ở mức đỉnh. Mục đích của bước này là xác định khái quát phạm vi, yêu cầu cũng như tính khả thi của phần mềm.

#### **Phân tích yêu cầu phần mềm (xác định yêu cầu phần mềm).**

Phân tích yêu cầu được tập trung việc thu thập và phân tích các thông tin cần cho phần mềm, các chức năng cần phải thực hiện, hiệu năng cần có và các giao diện cho người sử dụng.

Kết quả của phân tích là tư liệu về yêu cầu cho hệ thống và phần mềm (đặc tả yêu cầu) để khách hàng duyệt lại và dùng làm tài liệu cho người phát triển.

#### **Thiết kế (thiết kế hệ thống + thiết kế chi tiết)**

Là quá trình chuyển hóa các yêu cầu phần mềm thành các mô tả thiết kế.

Thiết kế gồm nhiều bước, thường tập trung vào một công việc chính: thiết kế kiến trúc phần mềm, thiết kế cấu trúc dữ liệu, thiết kế chi tiết các thủ tục, thiết kế giao diện và tương tác.

Lập tư liệu thiết kế (là một phần của cấu hình phần mềm) để phê duyệt.

#### **Mã hóa (lập trình).**

Biểu diễn thiết kế bằng một hay một số ngôn ngữ lập trình và dịch thành mã thực hiện được. Là bước chuyển hóa thiết kế chi tiết thành chương trình mà cuối cùng được biến đổi thành các lệnh mã máy thực hiện được.

#### **Kiểm thử**

Tiến trình kiểm thử bao gồm việc:

- Phát hiện và sửa lỗi phần logic bên trong chương trình hay còn gọi là lỗi lập trình.
- Kiểm tra xem phần mềm có hoạt động như mong muốn không, phát hiện và sửa lỗi về chức năng như thiếu hụt, sai sót về chức năng; và kiểm tra xem phần mềm có đảm bảo tính hiệu quả trong thực hiện hay không.

#### **Bảo trì**

Bao gồm các công việc sửa các lỗi phát sinh khi áp dụng chương trình hoặc thích ứng nó với thay đổi trong môi trường bên ngoài (hệ điều hành mới, thiết bị ngoại vi mới, yêu cầu người dùng) hoặc yêu cầu bổ sung chức năng hay nâng cao hiệu năng cần có.

***Suy nghĩ mới về vòng đời phần mềm.***

- Trước khi chuyển sang pha kế tiếp phải đảm bảo pha hiện tại đã được kiểm thử không còn lỗi.
- Cần có cơ chế kiểm tra chất lượng, xét duyệt giữa các pha nhằm đảm bảo không gây lỗi cho pha sau.
- Tư liệu của mỗi pha không chỉ dùng cho pha sau, mà chính là đối tượng quan trọng cho kiểm tra và đảm bảo chất lượng của từng quy trình và của chính phần mềm.
- Cần chuẩn hóa mẫu biểu, cách ghi chép tạo tư liệu cho từng pha, nhằm đảm bảo chất lượng phần mềm.
- Thao tác bảo trì phần mềm là việc xử lý quay vòng trở lại các pha trong vòng đời phần mềm nhằm biến đổi, sửa chữa, nâng cấp phần mềm.

### **3.5 Một số mô hình xây dựng phần mềm**

#### **3.5.1 Mô hình tuyến tính (The linear sequential model)**

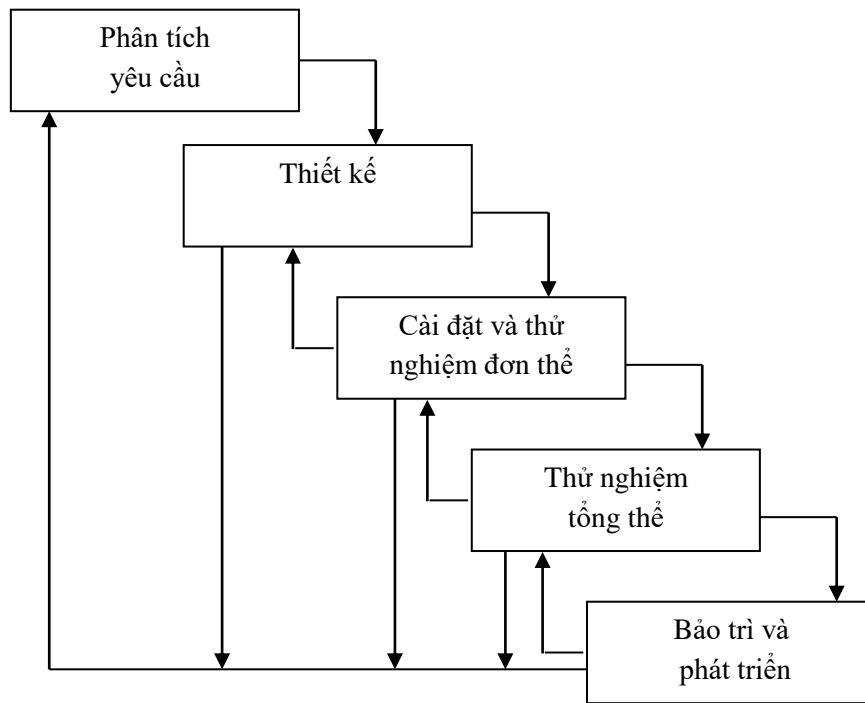
Đôi lúc còn được gọi là mô hình kinh điển (classic model) hay mô hình thác nước (waterfall model). Mô hình này xem quá trình xây dựng một sản phẩm phần mềm bao gồm nhiều giai đoạn tách biệt, sau khi hoàn tất một giai đoạn thì chuyển đến giai đoạn sau.

Có hai hoạt động phổ biến được thực hiện trong mỗi giai đoạn là: kiểm tra - phê chuẩn và quản lý cấu hình. Tổng kết mỗi giai đoạn là sự kiểm tra, phê chuẩn và quản lý cấu hình đây chính là mục tiêu của sản phẩm. Việc kiểm tra đưa ra khuôn mẫu đúng đắn tương ứng giữa sản phẩm phần mềm và các đặc tính của nó. Sự phê chuẩn đưa ra chuẩn mực về sự phù hợp hay chất lượng của sản phẩm phần mềm đối với mục đích của quá trình hoạt động.

Tuy vậy, thường thì các dự án có hàng ngàn trang tài liệu mà không ai ngoại trừ tác giả đọc đến nó. Thông tin ứng dụng chỉ nằm trong đầu mọi người và việc trao đổi thông tin là một trở ngại lớn để có được thành công của hệ thống. Kết luận là văn bản không phải là một phương tiện tốt để mô tả các yêu cầu phức tạp của ứng dụng. Thêm vào đó, mô hình bộc lộ một số nhược điểm quan trọng như:

- Mối qua hệ giữa các giai đoạn không được thể hiện
- Hệ thống phải được kết thúc ở từng giai đoạn do vậy rất khó thực hiện được đầy đủ những yêu cầu của khách hàng...

Mô hình này được tóm tắt như sau:

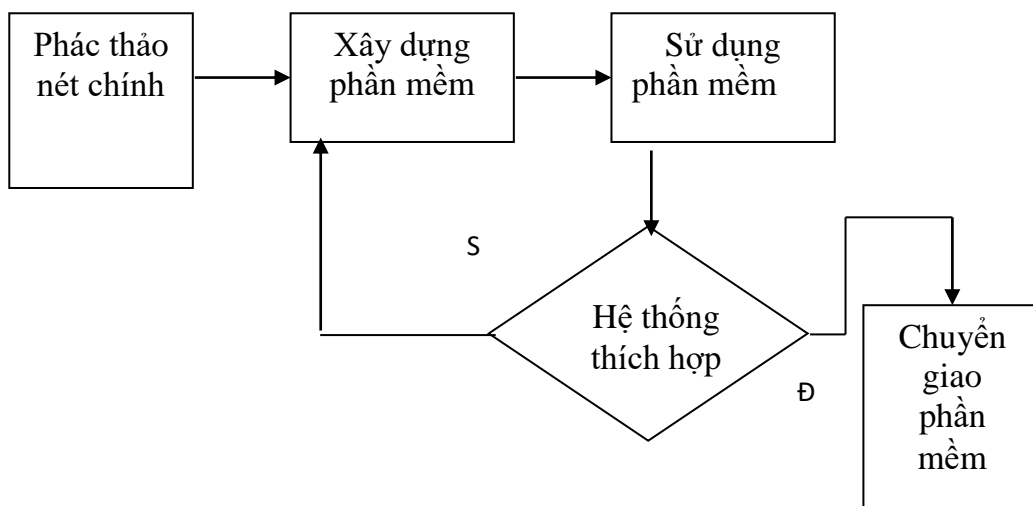


**Hình 3.3: Mô hình thác nước**

### 3.5.2 Mô hình mẫu (Prototyping model)

Thông thường, khách hàng sẽ đưa ra mục tiêu của họ một cách chung chung mà họ không biết hoặc không đưa ra một cách cụ thể những cái vào, cái ra và các tiến trình xử lý chúng. Thêm vào đó, chúng ta cũng không thể không quan tâm đến thuật toán sử dụng, tính tương thích của sản phẩm phần mềm với môi trường của nó như: phần cứng, hệ điều hành... Trong trường hợp này, mô hình mẫu có thể là sự lựa chọn tốt hơn cho người lập trình.

Những điểm chính của mô hình mẫu được tóm tắt theo sơ đồ sau:



### Hình 3.4: Mô hình Prototyping

Mô hình mẫu là một cách để phá vỡ sự khắt khe, cứng nhắc trong chu trình tuần tự của dự án. Tuy vậy, trong mô hình mẫu, sử dụng sai làm hỏng phân tích và thiết kế, không bao giờ hoàn thiện được mẫu thành các ứng dụng thực sự là các vấn đề cần quan tâm. Thêm vào đó là hệ thống có thể không bao giờ được chuẩn hóa, chi tiết của việc xử lý, việc kiểm tra tính hợp lệ của dữ liệu và các đòi hỏi kiểm toán có thể bị bỏ quên trong việc đưa mẫu vào sản xuất.

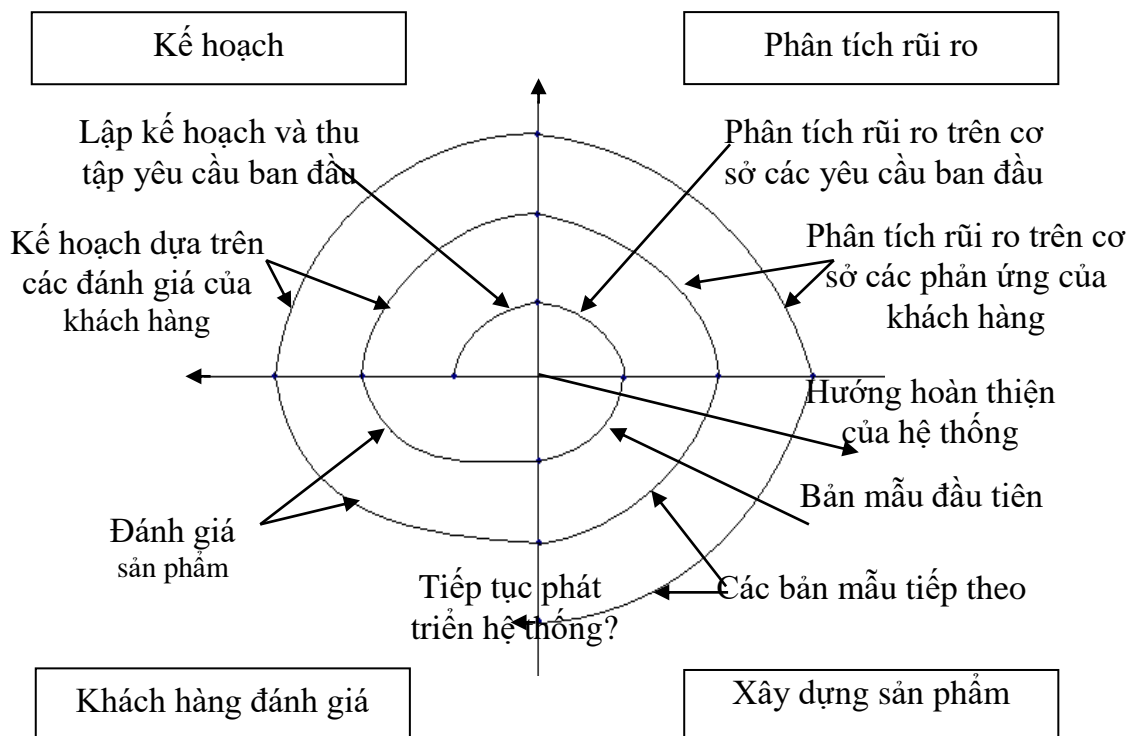
Trong tương lai, tạo mẫu thích hợp với đánh giá thiết kế, cải tiến cách dùng phần cứng và phần mềm mới. Tạo mẫu thường đi đôi với các ngôn ngữ lập trình bậc cao và ngày càng có nhiều công cụ đặt mẫu sẽ được tích hợp với CASE.

#### 3.5.3 Mô hình xoắn ốc (The spiral model)

Mô hình này được Boehm [1] đưa ra nên đôi lúc còn được gọi là mô hình Boehm's (The Boehm's spiral model). Nó có thể xem là sự kết hợp giữa mô hình thác nước và mô hình mẫu và đồng thời thêm một thành phần mới - phân tích rủi ro. Bao gồm bốn hoạt động chính:

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

Mô hình được tóm tắt như sau:



### Hình 3.5: Mô hình xoắn ốc (spiral)

Trong vòng đầu tiên của xoáy ốc, mục đích, lựa chọn, các ràng buộc được định nghĩa và các nguy cơ được xác định và phân tích. Nếu phân tích các lỗi chỉ ra rằng có một vài yêu cầu không chắc chắn, tạo mẫu có thể được tiến hành để giúp đỡ nhà phát triển và khách hàng. Mô phỏng và các mô hình khác có thể được sử dụng để xác định vấn đề và làm mịn các yêu cầu.

Khách hàng đánh giá công việc và đưa ra các gợi ý. Trên cơ sở ý kiến đó, phân tiếp theo của lập kế hoạch và phân tích lỗi xuất hiện.

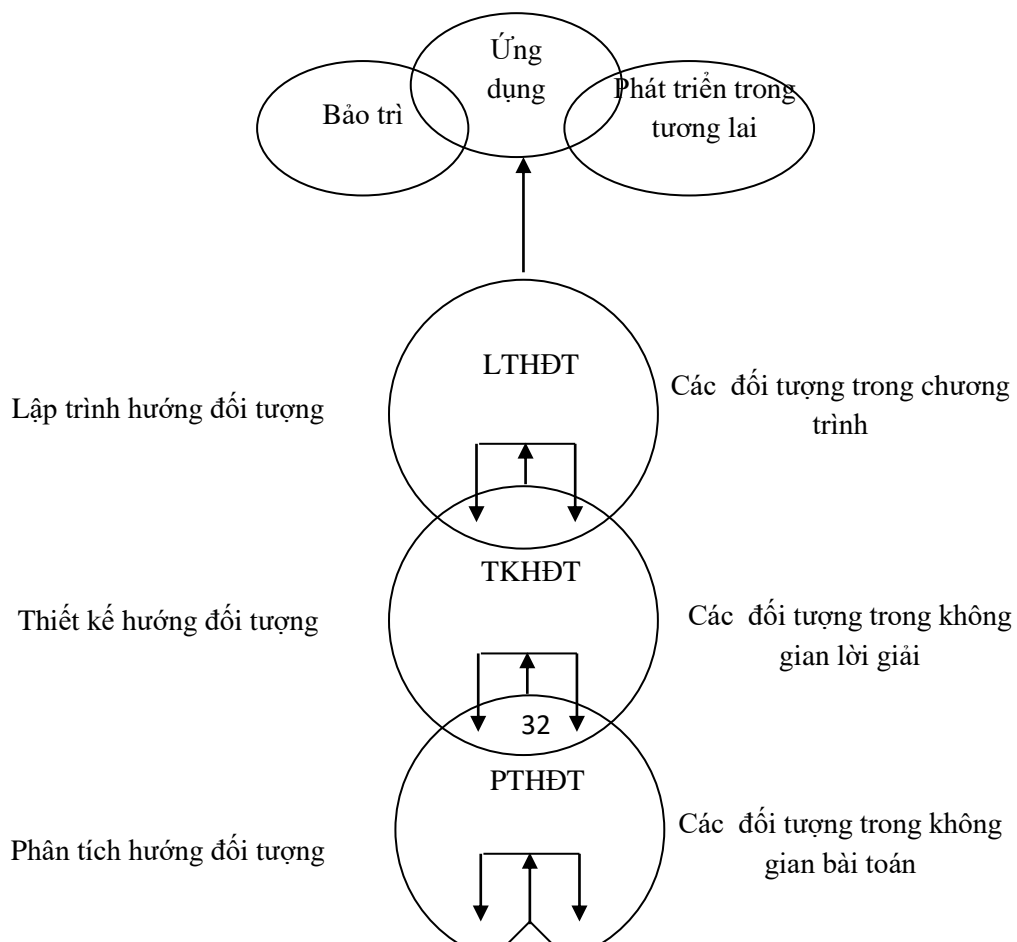
Mô hình xoáy ốc hiện nay là mô hình hướng tiếp cận hiện thực nhất để phát triển các hệ thống lớn. Nó sử dụng mô hình mẫu như là cơ chế loại trừ lỗi, cho phép nhà phát triển áp dụng mô hình mẫu tại mỗi chu trình phát triển. Nó kế thừa cách tiếp cận hệ thống từng bước từ chu kỳ sống cổ điển nhưng kết hợp với quá trình lặp lại phù hợp với thực tế.

Giống như các quy trình khác, mô hình xoáy ốc không phải là công cụ vạn năng. Đối với những hệ thống lớn, khó có thể điều khiển sự tiến hóa của phần mềm. Nó đòi hỏi phải có kỹ năng đánh giá lỗi. Cuối cùng là cần phải có thêm thời gian để kiểm nghiệm phương pháp mới này.

#### 3.5.4 Mô hình đài phun nước

Đây là mô hình của cách tiếp cận hướng đối tượng, hệ thống được xem như là một hệ thống các thực thể tác động qua lại để đạt được một mục đích nào đó. Mô hình này tương ứng với mô hình thác nước trong cách tiếp cận hướng thủ tục ở trên. Ở đây, ta thấy trong có những phần lặp và giao nhau giữa các bước phân tích, thiết kế và cài đặt.

Các điểm chính của mô hình được tóm tắt như sau:





**Hình 3.6: Mô hình đài phun nước**

### 3.5.5 Mô hình phát triển dựa trên thành phần

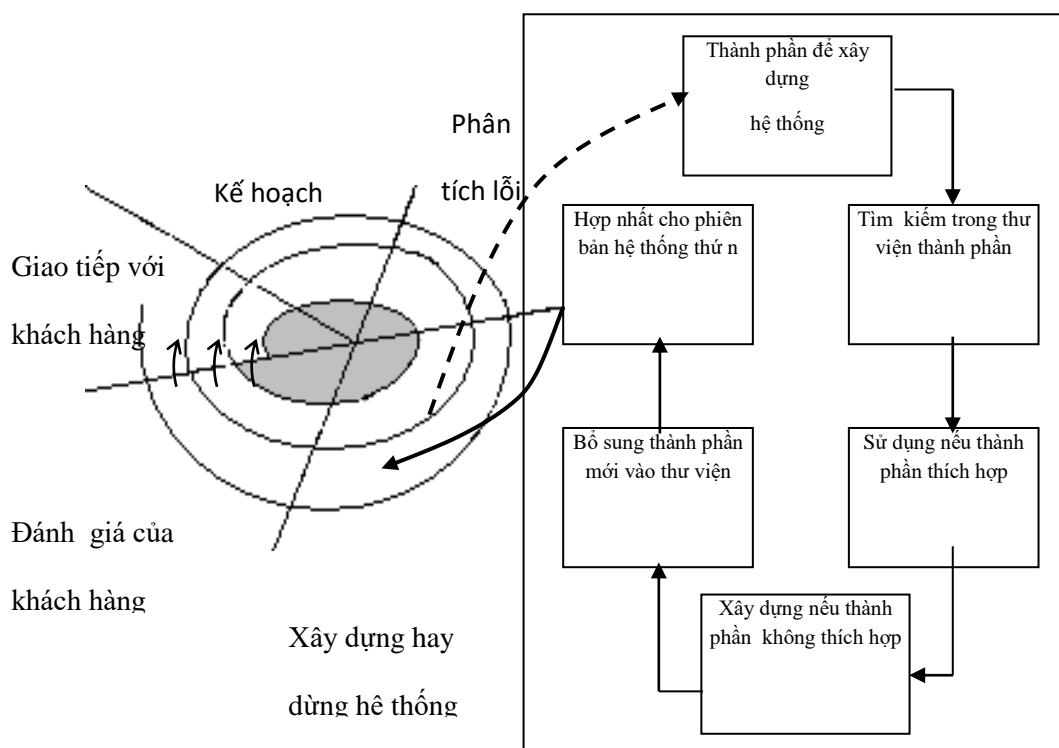
Xuất phát từ quan điểm: "Buy do not build", tư tưởng của phát triển dựa trên thành phần là lắp ráp hệ thống từ những thành phần đã có. Do vậy, kiến trúc phần mềm của hệ thống dựa vào kiến trúc phần mềm của các thành phần phần mềm tiêu chuẩn nên hệ thống đạt chất lượng cao hơn.

Phương pháp phát triển dựa trên thành phần gần tương tự như phương pháp phát triển hướng đối tượng. Hoạt động công nghệ bắt đầu với sự chỉ ra các lớp tham dự để phát triển hệ thống. Nếu các lớp này được tìm thấy trong thư viện và sự thích nghi là tốt, chúng sẽ được lấy ra và phát triển hệ thống. Ngược lại, chúng sẽ được phát triển để sử dụng và bổ sung vào thư viện sử dụng lại.

Thành phần phần mềm được sử dụng lại có độ chính xác cao và có thể nói là không chứa lỗi. Mặc dầu không thường xuyên được chứng minh về mặt hình thức nhưng với việc sử dụng lại, lỗi được tìm thấy và loại trừ; chất lượng của thành phần được cải thiện như là một kết quả.

Khi những thành phần sử dụng lại được ứng dụng thông qua tiến trình phần mềm, chúng ta ít tốn thời gian để tạo ra kế hoạch, mô hình, tài liệu, mã và dữ liệu mà chúng là cần thiết để tạo ra hệ thống. Thêm vào, chức năng cùng mức được phân phối cho người sử dụng với đầu vào ít công sức hơn, do vậy, hiệu suất phần mềm được cải thiện.

Những điểm chính của mô hình được tóm tắt như sau:



### ***Hình 3.7 Mô hình phát triển dựa trên thành phần***

#### **3.6 Phương pháp phát triển phần mềm**

Khác với thời kỳ đầu của tin học, các chương trình phụ thuộc nhiều vào thiết bị và người ta chỉ quan tâm đến các "mẹo vặt" lập trình, thì ngày nay người ta quan tâm đến nguyên lý và phương pháp để phát triển phần mềm. Các nguyên lý và phương pháp được đề xuất nhằm nâng cao năng suất lao động cho nhóm phát triển phần mềm. Năng suất ở đây bao gồm tính đúng đắn của sản phẩm, tính dễ đọc, dễ sửa đổi, dễ thực hiện, tận dụng được tối đa khả năng của thiết bị mà vẫn không bị phụ thuộc vào thiết bị.

Có nhiều phương pháp được đề cập như: phương pháp hướng chức năng, phương pháp hướng đối tượng, phương pháp ngữ nghĩa,... và thậm chí là không phương pháp để phát triển phần mềm. Bên cạnh các phương pháp để chỉ định cho việc tạo một bản phân tích và thiết kế, người ta còn chú ý đến phương pháp làm thế nào để đưa người dùng tham gia vào quy trình gọi là phương pháp luận xã hội.

#### **3.7 Vai trò của người dùng trong giai đoạn phát triển phần mềm**

Trong những ứng dụng trước kia được xây dựng thường xuyên không có sự bàn bạc với người sử dụng, sự cô lập của các công nghệ phần mềm đối với người dùng dẫn đến những hệ thống có khả năng làm việc về mặt kỹ thuật, nhưng thông thường không đáp ứng được nhu cầu của người sử dụng, và thường xuyên làm gián đoạn quá trình làm việc.

Để có sự tham gia của người sử dụng trong quá trình phát triển ứng dụng, phương thức này đòi hỏi những cuộc họp ngoài lề của tất cả những người sử dụng có liên quan và những người trong hệ thống - thường những người gặp nhau trong từ 5 đến 10 ngày để phát triển một mô tả chức năng chi tiết của những yêu cầu ứng dụng. Các cuộc họp ban ngày được sử dụng về những phân tích mới, những cuộc họp ban đêm lập tài liệu về những kết quả ban ngày để xem xét lại và tiếp tục chốt lại trong ngày tiếp theo.

Có rất nhiều lợi ích từ việc tham gia của người sử dụng trong phát triển ứng dụng.

- Trước tiên nó xây dựng sự cam kết của những người sử dụng - những người đương nhiên đảm nhiệm quyền sở hữu của hệ thống.
- Thứ hai, những người sử dụng là những chuyên gia thực sự của những công việc đang được tự động - lại được đại diện hoàn toàn thông qua sự phát triển.
- Thứ ba, những nhiệm vụ được người sử dụng thực hiện bao gồm việc thiết kế màn hình, các mẫu, các báo cáo, sự phát triển tài liệu của người sử dụng, sự phát triển và tiến hành của các cuộc kiểm tra công nhận,...

Sự tham gia của người sử dụng không chỉ là ước muốn mà còn là một mệnh lệnh đối với tiến trình và sản phẩm phát triển ứng dụng hoàn toàn hiệu quả. Khía cạnh quan trọng nhất của sự tham gia của người sử dụng là nó phải có ý nghĩa. Người sử dụng phải là những người

quyết định và là những người mong muốn tham gia vào quá trình phát triển. Sử dụng đội ngũ nhân viên ở cấp thấp hoặc chỉ định các nhà quản lý mở rộng không phải là cách để kéo người sử dụng vào các ứng dụng phát triển.

Mục tiêu của việc tham gia của người sử dụng là cho những người phát triển hệ thống và không phát triển hệ thống làm việc cùng với nhau như những đối tác chứ không phải như những kẻ thù. Khi những người sử dụng tham gia thì họ sẽ tạo ra những quy định không mang tính kỹ thuật. Những kỹ sư phần mềm giải thích và hướng dẫn người sử dụng tạo ra những quy định nữa kỹ thuật, ví dụ như việc thiết kế màn hình, và giải thích cả những tác động và suy luận của các quy định kỹ thuật chính yếu.

Việc tham gia của người sử dụng có nghĩa là người sử dụng sẽ điều khiển dự án, tạo nên phần lớn quy định và có tính quyết định cuối cùng đối với tất cả các quyết định lớn. Các kỹ sư phần mềm và các nhân viên của các hệ thống quản lý thông tin khác hoạt động như những kỹ thuật viên phục vụ, như là những chức năng của họ.

### **3.8. Qui trình phát triển phần mềm**

Một quy trình phần mềm là tập hợp các hoạt động dẫn đến sản xuất một sản phẩm phần mềm. Các hoạt động đó có thể bao gồm sự phát triển của phần mềm bắt đầu từ một ngôn ngữ lập trình chuẩn như Java hoặc C. Tuy nhiên, phần mềm mới được phát triển bởi sự mở rộng và thay đổi các hệ thống hiện hành và bởi sự biến dạng và tích hợp phần mềm dùng ngay hoặc các thành phần hệ thống.

Một lý do cho hiệu quả các công cụ CASE bị giới hạn là bởi sự đa dạng và rộng lớn của các quy trình phần mềm. Đó không phải là quy trình lý tưởng, và nhiều tổ chức cải tiến cách tiếp cận của chính họ đối với sự phát triển phần mềm. Các quy trình phát triển để khai thác các tiềm năng của con người trong một tổ chức và các đặc trưng của các hệ thống đang được phát triển. Với một vài hệ thống, như hệ thống quan trọng, yêu cầu một quy trình phát triển rất có cấu trúc. Với các hệ thống kinh doanh, có nhu cầu thay đổi nhanh chóng, một quy trình mềm dẻo và nhanh nhạy thì phù hợp hơn.

#### ***Bài tập***

- 1) Nêu các giá trị cốt lõi trong XP
- 2) Nêu vòng đời của một dự án XP
- 3) Nêu các công việc cốt lõi trong XP
- 4) RUP là gì? So sánh RUP với XP?
- 5) Kiến trúc của RUP là gì ?
- 6) Các luồng công việc trong RUP là gì ?
- 7) Scrum là gì ? Các công việc trong Sprint có gì khác so với một waterfall?
- 8) Nêu và giải thích các thành phần trong một khung làm việc Scrum?
- 9) So sánh các quy trình Waterfall, XP, RUP, Scrum.

## hương 4. DỰ ÁN PHẦN MỀM

### Mục tiêu

Mục tiêu của công nghệ phần mềm là sản xuất ra những phần mềm tốt, có chất lượng cao. Các nhân tố ảnh hưởng đến chất lượng phần mềm có thể được phân thành hai nhóm chính: các nhân tố có thể đo trực tiếp và các nhân tố chỉ có thể đo gián tiếp.

Tuỳ theo công dụng của sản phẩm và nhu cầu thực tế của người sử dụng, các chuẩn của quốc gia, quốc tế, nền văn minh của cộng đồng, thời điểm,... mà các tiêu chuẩn để lượng hoá phần mềm có thể thay đổi.

Chương này nhằm tìm hiểu các tiêu chuẩn hiện nay được dùng để đánh giá một sản phẩm phần mềm và cách thức để quản lý dự án phần mềm như :Lập kế hoạch dự án, tổ chức dự án, quản lý rủi ro, phát triển nhóm, quản lý chất lượng, lập kế hoạch làm việc chi tiết, kiểm soát và lập báo cáo dự án, quản lý thay đổi và quản lý cấu hình cuối cùng là hoàn tất dự án

### 4.1. Tổng quan về dự án phần mềm

#### Dự án phần mềm là gì ?

Theo từ điển bách khoa toàn thư Wikipedia thì Dự án là một tập hợp các công việc, được thực hiện bởi một tập thể, nhằm đạt được một kết quả dự kiến, trong một thời gian dự kiến, với một kinh phí dự kiến. ;

- Phải dự kiến nguồn nhân lực ;
- Phải có ngày bắt đầu, ngày kết thúc ;
- Phải có kinh phí thực hiện công việc ;
- Phải mô tả được rõ ràng kết quả (output) của công việc ;

#### Hoạt động và dự án khác nhau một cách cơ bản vì lý do sau :

Hoạt động là công việc diễn ra một cách thường xuyên, liên tục và có tính lặp lại ; trong khi dự án là những công việc mang tính chất tạm thời và duy nhất. Vì vậy, một dự án có thể được định nghĩa bằng tính đặc biệt của nó : « **Một dự án là một chuỗi các cố gắng nỗ lực mang tính tạm thời được thực hiện để tạo ra một sản phẩm hoặc một dịch vụ mang tính duy nhất** » (PMI).

Tính tạm thời được hiểu là các dự án có một điểm khởi đầu xác định và điểm kết thúc xác định. Tính duy nhất được hiểu là sản phẩm hay dịch vụ khác với tất cả các sản phẩm hay dịch vụ khác bởi một vài sự tiêu biểu nào đó.

Các dự án được thực hiện tại tất cả các mức khác nhau của tổ chức. Chúng có thể liên quan đến chỉ 1 người hoặc hàng nghìn người. Các dự án có thể mất ít hơn 100 giờ đồng hồ để hoàn thành hoặc cũng có thể mất hơn 10 triệu giờ. Các dự án có thể liên quan đến một đơn vị nhỏ trong tổ chức hoặc có thể vượt qua biên giới của tổ chức như trong các liên doanh.

Theo từ điển, dự án (project) là các hoạt động đã được thiết lập kế hoạch. Khái niệm này rất rộng, có thể nói tóm gọn là chúng ta có thể xác định được là làm thế nào để có thể tiến hành công việc trước khi bắt đầu. Việc lập kế hoạch thực chất là suy nghĩ rất cẩn thận về tất cả mọi vấn đề trước khi chúng ta thực hiện chúng.

**Phân biệt giữa dự án và dây chuyền sản xuất :**

***Bảng 4.1 : Phân biệt giữa hoạt động dự án và hoạt động nghiệp vụ***

<b>Hoạt động dự án</b>	<b>Hoạt động nghiệp vụ</b>
Tạo ra một sản phẩm xác định	Cho ra cùng một sản phẩm
Có ngày khởi đầu và ngày kết thúc	Liên tục
Đội ngũ nhiều chuyên môn khác nhau : <ul style="list-style-type: none"> <li>○ Khó trao đổi</li> <li>○ Ngại chi sẻ thông tin</li> </ul>	Các kỹ năng chuyên môn hóa
Đội hình tạm thời <ul style="list-style-type: none"> <li>○ Khó xây dựng ngay 1 lúc tinh thần đồng đội (teamwork)</li> <li>○ Khó có điều kiện đào tạo thành viên trong nhóm, trong khi cần phải sẵn sàng ngay</li> </ul>	Tổ chức ổn định <ul style="list-style-type: none"> <li>○ Có điều kiện đào tạo, nâng cấp các thành viên trong nhóm</li> </ul>
Dự án chỉ làm 1 lần	Công việc lặp lại và dễ hiểu
Làm việc theo kế hoạch trong một chi phí được phê duyệt	Làm việc trong một kinh phí thường xuyên hằng năm
Bị hủy nếu không đáp ứng mục tiêu, yêu cầu	Phải đảm bảo làm lâu dài
Ngày kết thúc và chi phí được tính theo dự kiến và phụ thuộc vào sự quản lý	Chi phí hàng năm được tính dựa trên kinh nghiệm trong quá khứ

Từ bảng so sánh trên ta thấy mỗi một dự án có một kết quả duy nhất ( được xác định), ví dụ tính duy nhất được so sánh trong bảng sau :

***Bảng 4.2 : Phân biệt giữa hoạt động dự án và hoạt động sản xuất***

<b>Hoạt động Dự án</b>	<b>Hoạt động sản xuất</b>
Xây nhà mới (cá nhân, doanh nghiệp)	Xây các căn hộ chung cư theo kế hoạch hàng năm của thành phố
Nghiên cứu một đề tài khoa học mới	Dạy học theo kế hoạch hằng năm của nhà trường.

	Hướng dẫn luận án sinh viên
Chế tạo bom nguyên tử, tàu vũ trụ	Sản xuất vũ khí hàng loạt
Xây dựng một phần mềm mới, do cơ quan đặt hàng	Áp dụng một phần mềm trong hoạt động thường ngày( quản lý kế toán, nhân sự, vật tư, sản xuất...)
Chế tạo một loại xe máy mới	Sản xuất hàng loạt xe máy theo thiết kế đã có sẵn, theo kế hoạch được giao
Thử nghiệm một dây chuyền sản xuất theo công nghệ mới	Sử dụng dây chuyền sản xuất giấy để sản xuất hàng loạt.

Theo như định nghĩa ở trên, một dự án phải có ngày bắt đầu và ngày kết thúc. Ngày kết thúc của dự án được xác định khi :

- Hoàn thành mục tiêu đề ra và nghiệm thu kết quả (kết thúc tốt đẹp) trước thời hạn ;
- Hết kinh phí trước thời hạn (kết thúc thất bại) ;
- Đến ngày cuối cùng (nếu tiếp tục nữa cũng không còn ý nghĩa) ;

#### **Tại sao các dự án phần mềm thường thất bại ?**

Một dự án được gọi là thất bại khi :

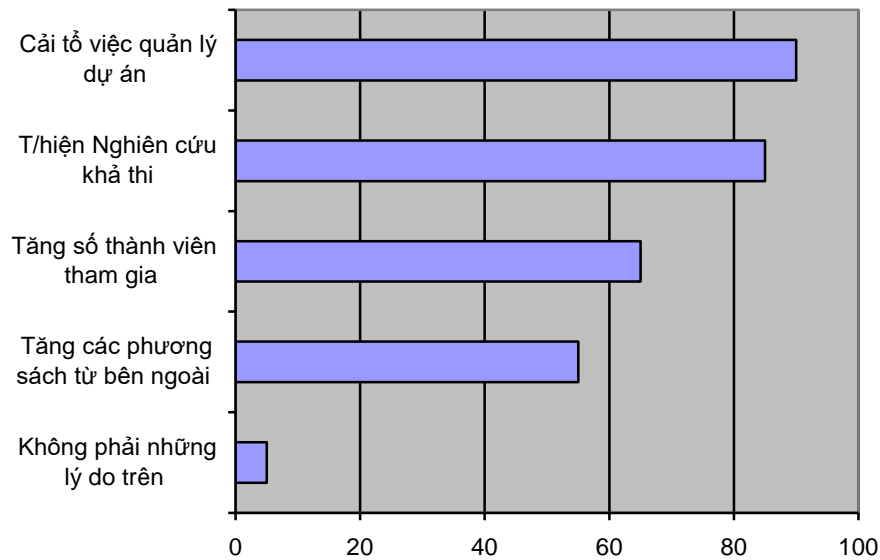
- Không đáp ứng được các mục tiêu ban đầu ;
- Không đáp ứng được thời hạn ;
- Vượt quá ngân sách cho phép (20-30%)

Vậy tại sao dự án phần mềm lại thất bại :

#### ***Hình 4.1: Tỷ lệ % các lý do khiến dự án thất bại***

- Các lý do khiến các dự án phần mềm thất bại là:
- (17%) không lường được phạm vi rộng lớn và tính phức tạp của công việc:
- (21%) Thiếu thông tin
- (21%) Không rõ mục tiêu
- (21%) Quản lý dự án kém
- (21%) Các lý do khác (mua phải thiết bị rởm, công nghệ quá mới đối tổ chức khiến cho không áp dụng được kết quả dự án, người bỏ ra đi...)

#### **Giải pháp nào để tránh sự thất bại của dự án**



**Hình 4.2. Các giải pháp tránh thất bại của dự án**

#### **4.1.2. Tỷ lệ thành công của dự án phần mềm**

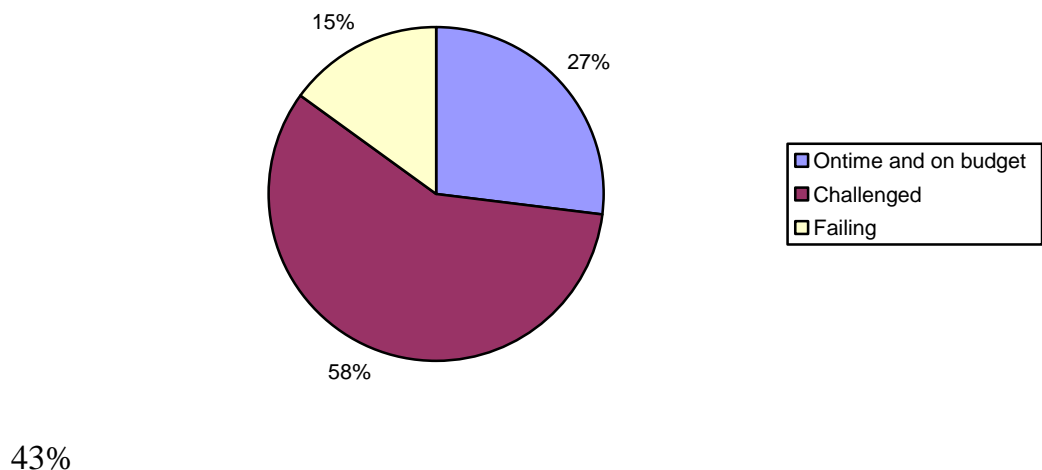
Năm 2001 Standish Group đã đưa ra báo cáo tỉ lệ thành công của các dự án công nghệ thông tin từ năm 1995 đến 2001 là:

- Các dự án thực hiện quá thời hạn: giảm từ 222% xuống còn 63%
- Thâm hụt chi phí giảm từ 189% xuống còn 45%
- Required features were up to 67% compared to 61%
- Có 78.000 U.S. Project thành công so với 28.000 trước đó
- 28% dự án IT thành công so với 16% của những năm trước đó:

Theo số liệu báo cáo mới của Standish Group CHAOS năm 2003, tỉ lệ thành công của các dự án phần mềm là rất ảm đạm.

- 15% các dự án phần mềm kết thúc trước thời hạn
- 66% được xem là thất bại

- Các dự án hoàn thành vượt mức chi phí trung bình là



**Hình 4.3. Tỷ lệ thành công của các dự án IT**

Và năm 2004, gần đây nhất Standish Group lại cho ra một báo cáo mới về tỷ lệ thành công của các dự án công nghệ thông tin.

Qua các số liệu trên cho thấy phải tiến hành cải tiến qui trình quản lý dự án, quản lý chất lượng, áp dụng qui trình quản lý một cách triệt để, cải tiến qui trình, tiêu chuẩn đánh giá chất lượng sản phẩm.

#### **4.1.3. Quản lý dự án là gì?**

##### **1. Định nghĩa:**

- Quản lý dự án (QLDA) là việc áp dụng các công cụ, kiến thức và kỹ thuật nhằm định nghĩa, lập kế hoạch, tiến hành triển khai, tổ chức, kiểm soát, đảm bảo chất lượng và kết thúc dự án.

- Một dự án quản lý tốt, tức là kết thúc phải thỏa mãn được chủ đầu tư về các mặt thời hạn, chi phí và chất lượng kết quả.

The Open University Software Project Management (1987) đưa ra khái niệm quản lý liên quan đến các hoạt động sau:

- Lập kế hoạch – Planning: quyết định những gì sẽ được thực hiện
- Tổ chức – Organizing – tạo ra sự sắp xếp, bố trí công việc, nhân lực,...
- Tổ chức đội ngũ- Staffing – chọn lựa đúng người để giao công việc
- Chỉ đạo – Directing – đưa ra các chỉ đạo thực hiện
- Giám sát – Monitoring – kiểm tra, giám sát tiến trình
- Kiểm soát – controlling – kiểm soát việc thực hiện để đưa ra những giải pháp hạn chế, đối phó, khắc phục
- Đổi mới – innovating – luôn cập nhật, hướng đến các giải pháp mới
- Đại diện – Representing – chịu trách nhiệm liên lạc với người dùng, cấp dưới.



## 2. Sơ lược về lịch sử ra đời khái niệm quản lý dự án.

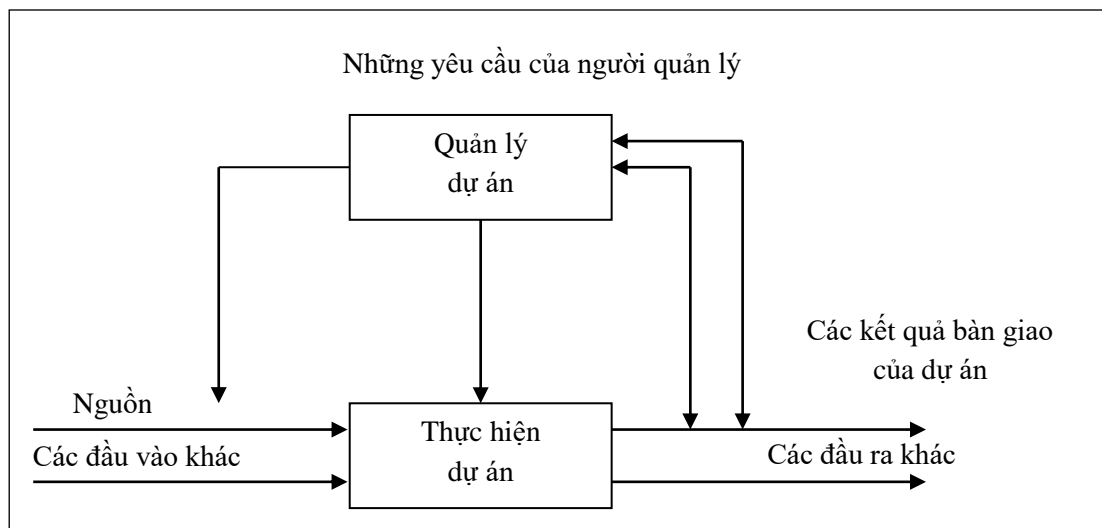
- Việc quản lý dự án đã có từ thời xưa: trong chiến tranh, xây dựng Kim tự tháp và các kỳ quan thế giới...
- Henry Gantt (đầu thế kỷ 20), đưa ra khái niệm sơ đồ Gantt dùng để quản lý các công việc của dự án và nay được phát triển và là một công cụ quản lý công việc của dự án trong các phần mềm công cụ quản lý dự án.
- Vào cuối những năm 50 ra đời sơ đồ mạng PERT (sơ đồ mạng công việc PERT) một kỹ thuật xem xét và đánh giá chương trình/dự án (PERT sẽ được trình bày thêm ở các phần sau)
- Sau này, bổ sung thêm những ý tưởng về tổ chức, kiểm soát, sử dụng tài nguyên trong quản lý dự án và dần dần khái niệm Quản lý dự án được hoàn thiện và là một trong những lý thuyết/nguyên lý cơ bản dành cho những người quản lý, quản trị.

## 3. 9 lĩnh vực kiến thức của PMI trong quản lý dự án

Viện quản lý dự án PMI.org đưa ra 9 lĩnh vực kiến thức cần quan tâm trong quản lý dự án nói chung và quản lý dự án phần mềm nói riêng. Khi nói đến dự án là phải nói đến 9 vấn đề sau và tất cả các vấn đề của dự án đều nằm trong chín (9) vấn đề cốt lõi này.

- Quản lý tích hợp (project integration management)
- Thời gian (Time)
- Chất lượng (Quality)
- Nguồn nhân lực (Human resource)
- Giao tiếp, truyền thông (Communications)
- Rủi ro (risk)
- Kết quả đạt được (procurement)

## 4. Sơ đồ dòng thông tin thể hiện việc quản lý và thực hiện dự án

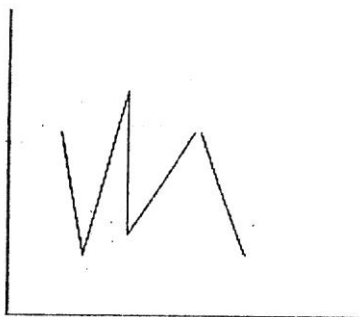


#### Hình 4.4. Sơ đồ dòng thông tin quản lý và thực hiện dự án

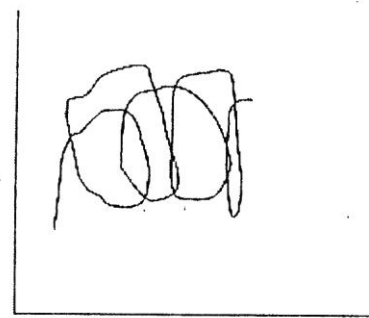
##### 5. Các phong cách quản lý dự án

Có 4 phong cách quản lý dự án của các nhà quản lý, lãnh đạo

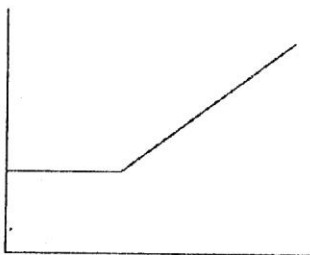
- (1) Sau khi vạch kế hoạch rồi, phó mặc cho mọi người thực hiện, không quan tâm theo dõi, khi có chuyện gì xảy ra mới nghĩ cách đối phó.
- (2) một đề tài nghiên cứu khoa học: Không có sáng kiến mới, cứ quanh quẩn với các phương pháp cũ, công nghệ cũ
- (3) Không lo lắng đến thời hạn giao nộp sản phẩm, đến khi dự án hết hạn thì mới lo huy động thật đông người cho xong.
- (4) quản lý chủ động, tích cực. Suốt quá trình thực hiện dự án không bị động về kinh phí, nhân lực và tiến độ đảm bảo (lý tưởng)



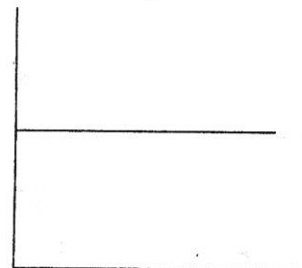
(1) Quản lý theo kiểu đối phó



(2) Quản lý theo kiểu mất phương hướng



(3) Quản lý nước đến chân mới nhảy



(4) Quản lý chủ động

Dưới đây là các đồ thị biểu diễn các phong cách quản lý ở trên

##### 6. Các đặc điểm của phong cách quản lý dự án thụ động

- Quản lý dự án luôn đứng sau các mục tiêu của dự án
- Hấp tấp, bị kích động, tương lai ngắn hạn
- Khi làm quyết định, chỉ nghĩ đến các khó khăn trở ngại tạm thời, trước mắt, không nghĩ đến liệu rằng đó là 1 bước đi đúng hay không.

- Không kiểm soát được tình thế. Nhiều khi phải thay đổi kế hoạch và tổ chức.

Với phong cách quản lý dự án thụ động như thế thì sẽ mang lại những hậu quả gì cho tổ chức, cho các bên liên quan đến dự án? Câu trả lời cho vấn đề này là:

- Kết quả thu được không ổn định

- Tinh thần làm việc không cởi mở, hợp tác
- Năng suất thấp, công việc không chạy
- Không sử dụng hiệu quả tài nguyên
- Người quản lý dự án bị dự án quản lý
- Hồ sơ dự án kém chất lượng
- Chậm tiến độ dự án, chỉ tiêu vượt quá kinh phí cho phép
- Chất lượng dự án không đảm bảo

### **7. Các thuộc tính của một dự án phần mềm**

Các dự án công nghệ phần mềm nói riêng và các dự án công nghệ thông tin nói chung có rất nhiều đặc điểm khác với các loại dự án khác. Các dự án CNPM bao gồm các thuộc tính.

- Kết quả bàn giao có thể là ít hữu hình
- Phạm vi có thể khó kiểm soát
- Kỹ năng, kinh nghiệm, thái độ và kỳ vọng trái ngược nhau
- Có thể bất đồng về mục tiêu kinh doanh
- Thay đổi quan trọng về tổ chức
- Các yêu cầu, phạm vi, và lợi nhuận chính xác có thể rất khó xác định
- Sự thay đổi nhanh chóng về công nghệ

### **8. Quản lý dự án phần mềm khác với các loại dự án khác**

Có rất nhiều kỹ thuật quản lý các dự án thông thường có thể áp dụng cho quản lý dự án phần mềm, tuy nhiên việc quản lý các dự án phần mềm có một số đặc điểm khác với các loại quản trị dự án khác.

Một các nhận biết sự khác nhau của quản lý dự án phần mềm là quá trình của việc tạo ra các hữu hình mà bản thân nó là sự vô hình

- Tính vô hình (invisibility): khi “một vật” được tạo ra như một cây cầu hay một con đường đang được xây dựng thì tiến trình (progress) thực sự có thể nhìn thấy được. Còn với phần mềm thì tiến trình không thể thấy ngay được

- Sự phức tạp (complexity): trên mỗi đồng dollar, pound hay euro được chi ra, sản phẩm phần mềm chứa đựng một sự phức tạp hơn nhiều so với các kỹ sư chế tạo khác.

- Sự phù hợp (comformity): một kỹ sư truyền thống thường làm việc với các hệ thống vật lý và các nguyên liệu vật lý như xi măng, sắt thép. Những hệ thống vật lý này có thể phức tạp nhưng được kiểm chế (quản lý) bởi các qui định (nguyên tắc) phù hợp. Còn người phát triển phần mềm phải làm theo các yêu cầu của khách hàng, nó không chỉ mang tính cá nhân mà còn không phù hợp.

- Tính linh hoạt (flexibility): do sự thoải mái (không bị ràng buộc) mà phần mềm có thể được thay đổi thường xuyên và đây được xem như là một thế mạnh của phần mềm. Tuy

nhiên, điều này có nghĩa là phần mềm phải luôn đáp ứng được các cấp độ thay đổi cao của người dùng, của khách hàng.

### **Một số vấn đề cần lưu ý về quản lý dự án**

- Việc quản lý dự án có thành công hay không đó là vấn đề về con người, yếu tố con người quyết định rất lớn sự thành bại của dự án
- Quản lý dự án là tìm ra các nguồn hỗ trợ và những vấn đề cản trở dự án
- Nhìn vào bản chất vấn đề/dự án, không tin hiện tượng
- Những người quản lý khác nhau có cách nhìn về dự án/về quản lý cũng khác nhau.
- Thiết lập kế hoạch chỉnh sửa dễ dàng
- Quản lý dự án là dám đối mặt với sự kiện, sự thật, nhìn thẳng vào vấn đề.
- Sử dụng quản trị để hỗ trợ cho các mục tiêu của dự án
- Sử dụng mục tiêu đối với từng nhiệm vụ không được giống như đã nêu trong kế hoạch
- Đọc lại phạm vi và các mục tiêu của dự án mỗi tuần 1 lần
- Không ngạc nhiên (phong cách của người quản lý chủ động, tích cực)

Một số vấn đề thông thường mà các dự án phần mềm hiện nay gặp phải

- Thiếu các tiêu chuẩn đo lường và tiêu chuẩn chất lượng
- Khó khăn trong việc giám sát tiến trình một cách trực quan
- Khả năng giao tiếp, truyền thông kém
- Các yêu cầu dự án thay đổi thường xuyên
- Vượt ngân sách và thời hạn bàn giao các sản phẩm không đúng hạn (muộn)

Các chiến lược trong quản lý dự án phần mềm

Trong quản lý dự án phần mềm, các nguyên tắc cơ bản sau cần phải được xác định rõ và thực hiện đầy đủ để đảm bảo dự án được thành công, đó là:

- Tránh xa các lỗi cổ điển
- Hiểu rõ các nguyên tắc cơ bản phát triển đó là
- Thực hiện quản lý rủi ro
- Thực hành lập lịch theo hướng đối tượng

Các chiến lược này sẽ làm cân bằng quá trình quản lý dự án

## **4.2. Người quản lý dự án và các tiêu chuẩn chọn lựa người QLDA**

Để xác định được các bên/người/bộ phận liên quan đến dự án thì với vai trò là một giám đốc dự án, bạn sẽ giao tiếp, làm việc với những ai. Trả lời được câu hỏi này điều đó xác định được các stakeholder của dự án. Việc xác định được các stakeholder của dự án là rất quan trọng, giúp cho việc phát triển dự án đúng mục tiêu, đúng yêu cầu của những người liên quan.

Các stakeholder của dự án gồm

- Nhà tài trợ cho dự án
- Người điều hành và quản lý dự án (giám đốc dự án)
- Đội dự án
- Khách hàng
- Nhà thầu
- Các giám đốc của các bộ phận chức năng

#### 4.2.1. Vai trò, trách nhiệm của từng thành viên trong dự án

- Người quản lý dự án (PM-Project Manager): Chịu trách nhiệm chính về kết quả của dự án. Có vai trò chủ chốt trong việc xác định các mục đích và mục tiêu, xây dựng các kế hoạch dự án, đảm bảo dự án được thực hiện có hiệu lực và hiệu quả.

- Người tài trợ dự án (PS-Project sponsor): Cấp tiền cho dự án thành công dự án, quyết định cho dự án đi tiếp hay cho dừng giữa chừng.

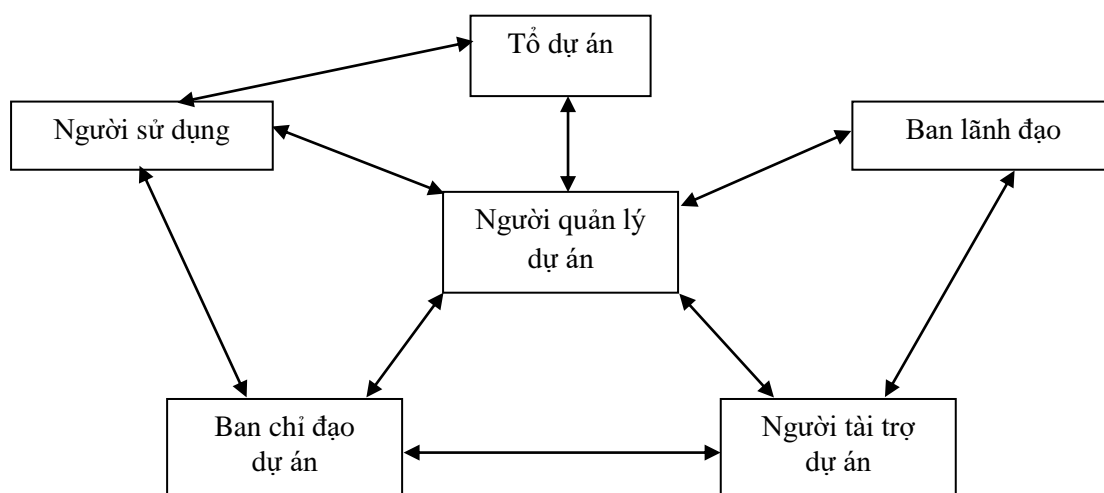
- Đội dự án (PT – Project team): Hỗ trợ cho PM để thực hiện thành công dự án. Bao gồm những người vừa có kỹ năng (skill) và năng lực (talent) để triển khai thực hiện dự án như: nhóm phát triển, nhóm phân tích và thiết kế, nhóm kiểm thử...

- Khách hàng (Client): Thụ hưởng kết quả dự án. Nêu yêu cầu, cử người hỗ trợ dự án. Là người chủ yếu nghiệm thu kết quả dự án.

- Ban lãnh đạo (Senior Mangement):Bổ nhiệm PM và PT, tham gia vào việc hình thành và xây dựng dự án.

- Các nhóm hỗ trợ (có thể có nhiều hay ít, tùy từng dự án). Ban điều hành (Steering Committee), nhóm kỹ thuật, nhóm thư ký...

*Sơ đồ sau biểu diễn mối quan hệ giữa các người có liên quan đến dự án:*



**Hình 4.5. Mối quan hệ giữa các người có liên quan đến dự án**

#### 4.2.2. Trách nhiệm của người quản lý dự án – PM

**Bảng 4.3. Trách nhiệm của người quản lý dự án**

<b>Trách nhiệm chính</b>	<b>Chi tiết</b>
Nêu ra những điểm bao quát chung	Về công việc, cấu trúc phân việc, kịch bản và ngân sách
Trao đổi với các tổ dự án, nhóm dự án	Bao gồm các báo cáo, biểu mẫu, bản tin, hội họp, và thủ tục làm việc, ý tưởng là trao đổi cởi mở và trung thực trên cơ sở đều đặn
Động viên, khuấy động tinh thần làm việc	Bao gồm khích lệ, phân việc, mời tham gia và ủy quyền
Hỗ trợ cho mọi người	

#### **15 chức năng của một PM (người quản lý dự án phần mềm)**

Mỗi một người quản trị dự án luôn có vai trò vị trí rất quan trọng trong dự án, chức năng, nhiệm vụ của PM rất nặng và sau đây là các công việc chính của một PM trong các dự án phần mềm.

- Xác định phạm vi của dự án
- Xác định các stakeholders, người đưa ra các quyết định về dự án
- Đưa ra bảng công việc chi tiết (work breakdown structures)
- Ước lượng thời gian cho các yêu cầu
- Xây dựng sơ đồ dòng thông tin quản lý dự án
- Xác định các yêu cầu về tài nguyên, nguồn lực và ngân sách
- Đánh giá các yêu cầu của dự án
- Xác định và đánh giá các rủi ro và chuẩn bị kế hoạch đối phó với các trường hợp bất ngờ
- Xác định các sự phụ thuộc lẫn nhau giữa các công việc
- Xác định và theo dõi các cột mốc hoàn thành các giai đoạn của dự án
- Tham gia vào các giai đoạn xem xét dự án
- Đảm bảo an toàn về nhu cầu nguồn lực cho dự án
- Quản lý các quá trình kiểm soát thay đổi
- Báo cáo trạng thái dự án cho các cấp lãnh đạo

#### 4.2.3. Chọn nhân sự cho dự án

Nhân sự thực hiện dự án trong những yếu tố rất quan trọng, người quản lý, tổ chức phải có một sự đánh đúng năng lực của từng thành viên dự án và phân công đúng người đúng việc.

Một số tiêu chí chọn nhân sự cho dự án bao gồm, nhân sự có:

- Kiến thức kỹ thuật tốt;
- Chuyên môn đặc biệt;
- Đã có kinh nghiệm, yếu tố kinh nghiệm rất quan trọng, bởi ngoài chuyên môn, kỹ năng thì cần phải có kinh nghiệm để giải quyết vấn đề một cách hiệu quả và nhanh chóng.
- Đã tham gia dự án nào chưa?
- Quyền lực của phòng, ban của người đó?
- Hiện có tham gia dự án nào khác không? Nhân sự chọn cho dự án cần phân biệt những người làm toàn thời gian (full-time) hay những người bán thời gian (part-time).
- Khi nào kết thúc?
- Dành bao nhiêu thời gian cho dự án?
- Khối lượng công việc chuyên môn hiện nay nhiều hay ít.
- Quan hệ đồng nghiệp, khả năng giao tiếp, truyền thông
- Có hăng hái tham gia, lòng yêu nghề, đam mê công việc;
- Có truyền thống làm việc với hiệu quả cao không?
- Có ngăn nắp và quản lý thời gian tốt không?
- Có tinh thần trách nhiệm không?
- Có tinh thần hợp tác không?
- Thủ trưởng của người đó có ủng hộ không.
- Và một số tiêu chí phụ khác

Với các tiêu chí trên sẽ chọn ra được những thành viên của nhóm dự án có đủ khả năng để hoàn thành dự án. Tuy nhiên, cũng có vấn đề đặt ra là cần phải xây dựng được tập thể này vững mạnh thì cần các yếu tố sau:

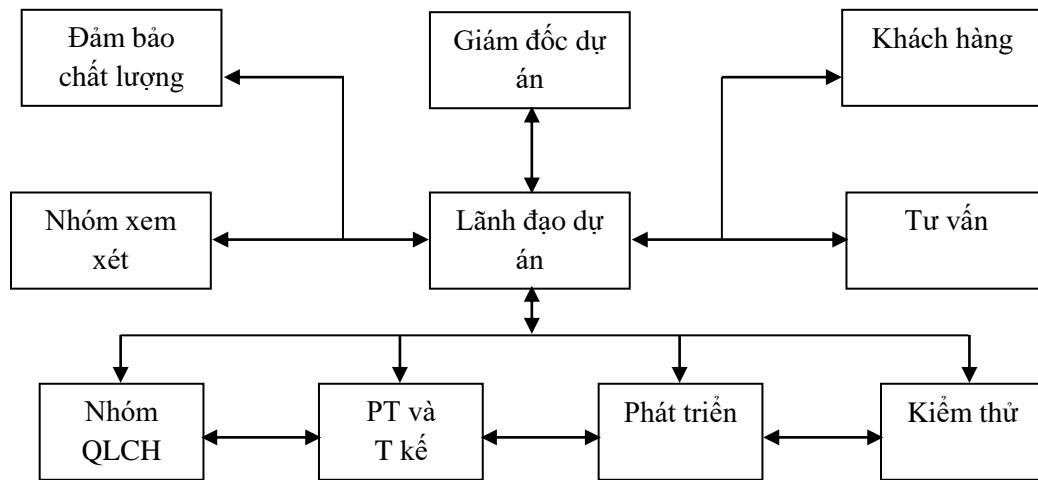
- Bổ nhiệm người phụ trách, lãnh đạo từng nhóm công việc, làm việc theo cấp.
- Phân bổ trách nhiệm: giao việc đúng người, đúng đối tượng và có trách nhiệm trên công việc được giao.
- Khuyến khích tinh thần đồng đội, làm việc nhóm, hỗ trợ giúp đỡ lẫn nhau.
- Làm phát sinh lòng nhiệt tình
- Thành lập sự thống nhất chỉ huy, phân cấp báo cáo, quản lý
- Cung cấp môi trường làm việc tốt, kỹ năng giao tiếp, truyền thông

Đối với các nhà quản lý dự án, không chỉ họ có sức ép từ phía khách hàng hay nhóm phát triển dự án mà còn từ rất nhiều phía, bao gồm: các tiêu chuẩn, nguồn lực, uy tín, công nghệ, thủ tục hành chính, môi trường kinh doanh...

Để đảm đương được áp lực đòi hỏi tổ chức phải chọn ra người đứng đầu cho dự án phải có những phẩm chất, năng lực nhất định để lãnh đạo dự án, đó là:

- Trung thực, phản ứng tích cực, toàn tâm toàn ý, Nhất quán, kiên quyết, nhìn xa trông rộng, khách quan, gương mẫu, lời cuốn, khả năng giao tiếp và xử lý tình huống, khả năng đàm phán và một số khả năng khác.

#### 4.2.4. Cấu trúc nhân dự nhóm dự án phần mềm



**Hình 4.6. Cấu trúc nhân sự nhóm dự án**

Sơ đồ này thể hiện được vị trí, mối quan hệ giữa các nhóm/ các thành viên trong dự án. Và thể hiện được sự quản lý trong dự án giúp cho phân bổ nhân sự một cách hợp lý vào các nhóm của dự án, vai trò và sự ảnh hưởng của các bên tham gia dự án.

#### Trách nhiệm chính của các thành viên trong đội dự án

**Bảng 4.4. Trách nhiệm của các thành viên đội dự án**

Tên	Vị trí	Trách nhiệm
<b>Nhóm Tư vấn:</b> chịu trách nhiệm về việc tư vấn tất cả các yêu cầu của nhóm phát triển		
<b>Nhóm phân tích &amp; Thiết kế:</b> chịu trách nhiệm phân tích và thiết kế hệ thống		
<b>Nhóm phát triển:</b> chịu trách nhiệm viết code, kiểm tra đơn vị xử lý lỗi		
<b>Đội kiểm thử:</b> có trách nhiệm chuẩn bị kế hoạch kiểm thử và thực hiện kiểm thử theo kế hoạch, xây dựng các kịch bản kiểm thử		
<b>Nhóm quản lý cấu hình:</b> có trách nhiệm kiểm soát, quản lý các thay đổi và các phiên bản, xuất bản của sản phẩm		
<b>Người xem xét:</b> phản hồi các thông tin về dự án quan việc xem xét các báo cáo của dự án		
<b>Đảm bảo chất lượng:</b> có trách nhiệm xem xét toàn bộ chất lượng của dự án thông qua các hoạt động bên trong của dự án		
<b>Khách hàng:</b> cung cấp các thông tin, yêu cầu cho dự án và là người chịu trách nhiệm kiểm thử hệ thống lần cuối cùng, và phê duyệt các tài liệu, sản phẩm, khoản mục chuyển giao		



### 4.3. Bốn “chiều” của dự án

Bất kỳ một dự án nào dù lớn hay nhỏ luôn luôn có 4 yếu tố (3P+1T) tác động/ảnh hưởng đến dự án, một dự án phải có **người** để thực hiện, phải được thực hiện theo một **quy trình** nhất định để tạo ra được một **sản phẩm** dựa trên **công nghệ** X để phát triển, bao gồm:

- People – con người
- Process – Quy trình
- Product – Sản phẩm
- Technology – công nghệ

P – Yếu tố con người rất quan trọng, dự án phát triển một sản phẩm chỉ có 1 người khác với dự án 10 người làm ra, một nhóm 3 hay 5 người phát triển một sản phẩm khác với 1 người.

P – Phát triển một sản phẩm theo một nền tảng hay quy trình cụ thể, cơ bản nào đó, có quản lý rủi ro, quản trị chất lượng, lifecycle...

P – Sản phẩm được phát triển phải hữu hình, phải định lượng, phải trực quan và mang lại lợi ích khi sử dụng sản phẩm.

T – Và cuối cùng là công nghệ, bất kỳ một sản phẩm nào được phát triển/ được tạo ra đều phải dựa trên một hoặc nhiều công nghệ dù đó là công nghệ gì.

Một dự án luôn tồn tại 4 yếu tố/ khía cạnh (3P + 1T), để hiểu rõ được từng yếu tố trên ta đi sâu xem xét từng khía cạnh một của vấn đề.

#### 4.3.1. Yếu tố con người – People

Yếu tố con người rất quan trọng trong sự thành bại của dự án. Việc chọn lựa, tổ chức nhóm dự án, phân công/ giao việc một cách hợp lý sẽ mang lại hiệu quả cho dự án.

- Năng suất của người phát triển dự án
- Sự cải tiến đổi mới:
- Chọn đội dự án (Team selection)
- Tổ chức độ dự án (Team organization)
- Thúc đẩy, động viên (Motivation)

Các nhân tố thành công khác:

- o Giao đúng việc đúng người
- o Phát triển/ phát huy nghề nghiệp
- o Cân bằng được: đâu là cái chung và đâu là riêng tư
- o Khả năng giao tiếp/ truyền đạt thông tin rõ ràng, mạch lạc

#### 4.3.2. Quy trình – Process

Để phát triển một sản phẩm có chất lượng phải thực hiện theo các tiến trình đó là:

- o Xác định xem quy trình có khó thực hiện?
- o Quy trình có 2 loại: quy trình quản lý và quy trình kỹ thuật
- o Các nền tảng quy trình phát triển dự án

- Quy trình đảm bảo chất lượng
- Quy trình quản lý rủi ro
- Lifecycle lập kế hoạch dự án
- Hướng khách hàng
- Các quy trình cải tiến
- Tránh làm lại các công việc đã thực hiện (rework)

#### **4.3.3. Sản phẩm – Product**

- Sản phẩm phải rõ ràng, xác thực, hiển nhiên
- Quản lý kích thước của sản phẩm
- Các yêu cầu và đặc tính của sản phẩm

#### **4.3.4. Công nghệ - Technology**

- Về lĩnh vực công nghệ thì đây là hướng ít quan trọng nhất đối với 3 lĩnh vực trên (3P). Công nghệ gì được sử dụng để phát triển sản phẩm, hay được ứng dụng công cụ, phương pháp gì để triển khai dự án, phát triển, thiết kế sản phẩm.
- Lựa chọn công cụ và ngôn ngữ
- Giá trị và chi phí của việc tái sử dụng các công nghệ gì, công cụ và ngôn ngữ cho việc triển khai dự án, phát triển sản phẩm

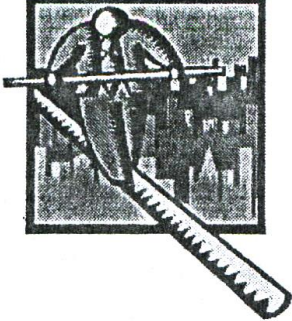

#### **4.4. Lập kế hoạch**

Ta cũng đã biết được tầm quan trọng của việc lập kế hoạch dự án, nhưng để có thể lập được kế hoạch của dự án thì sau khi nghiên cứu tính khả thi của dự án chúng ta cần phải:

- Xác định các yêu cầu của dự án, bởi có được yêu cầu thì mới có dự án, điều này cũng đồng nghĩa với việc xác định mục tiêu của dự án cần triển khai thực hiện.
- Xác định nguồn lực/ nguồn tài nguyên cho dự án bao gồm vật lực, nhân lực, chi phí
  - Lựa chọn mô hình phát triển (lifecycle model): waterfall hay incremental hay spiral... tùy thuộc vào từng loại dự án, kích thước...
  - Xác định cho được các đặc tính/ tính năng của sản phẩm để từ đó có được kế hoạch phát triển đúng hướng.

Những lợi ích của kế hoạch quản lý dự án

**Bảng 4.5. Lợi ích của việc lập kế hoạch quản lý**

Rủi ro khi không lập kế hoạch quản lý	Lợi ích khi lập kế hoạch quản lý
	
<ul style="list-style-type: none"> <li>○ Khởi đầu sai lệch</li> <li>○ Dễ bị nhầm lẫn</li> <li>○ Không đáp ứng được sự mong đợi của nhà tài trợ và/hoặc các mục tiêu</li> <li>○ Thông tin nghèo nàn</li> </ul>	<ul style="list-style-type: none"> <li>○ Đáp ứng các mục tiêu</li> <li>○ Gây dựng lòng tin của người góp vốn</li> <li>○ Thiết lập hướng làm việc chung</li> <li>○ Mở ra các kênh thông tin liên lạc</li> <li>○ Bắt đầu tự án với mọi phương thức có hệ thống</li> </ul>

#### 4.5. Sự giám sát dự án

Sau khi tiến hành lập kế hoạch xong, dự án được triển khai thực hiện theo kế hoạch thì cũng là lúc chuyển sang giai đoạn giám sát, điều khiển dự án. Giám sát theo kế hoạch đã lập để đảm bảo dự án được phát triển đúng kế hoạch đã xây dựng. Bao gồm:

- Chi phí, nỗ lực và lịch trình thực hiện
- Giám sát và so sánh giữa các hoạt động theo kế hoạch và thực tế
- Có phương án để giải quyết, làm thế nào để xử lý khi không đi đúng kế hoạch?

#### 4.6. Các định chuẩn và tiêu chuẩn đo lường

##### 4.6.1. Đo lường lịch trình và chi phí

- Xác định, đo lường chi phí dự án (mức độ chi tiêu cho dự án)
- Lịch trình thực hiện, xác định lịch trình mức độ hoàn thành (Schedule)
- Đánh giá sự nỗ lực thực hiện các công việc
- Đo lường, đánh giá hiệu quả, tính năng của sản phẩm

##### 4.6.2. Các tiêu chuẩn, công cụ đo lường khác

Ngoài việc sử dụng các phép đo lường trên để xác định sự thành công của dự án còn có các công cụ, phương pháp hay tiêu chuẩn khác để đánh giá sự thành công của dự án. Đó là:

- Phân tích các giá trị thu được – EVA
- Tỷ lệ/ tần suất xảy ra các lỗi
- Hiệu suất (ex: SLOC)

- Mức độ phức tạp đứng trên quan điểm các chức năng của sản phẩm (ex: function points)

#### **4.7. Các nguyên tắc cơ bản về kỹ thuật**

Về phương diện kỹ thuật, sản phẩm được phát triển dựa trên các kỹ thuật:

- Xác định các yêu cầu
- Phân tích các yêu cầu
- Thiết kế hệ thống
- Xây dựng/ coding sản phẩm
- Đảm bảo chất lượng cho sản phẩm
- Và triển khai sản phẩm

#### **4.8. Các giai đoạn phát triển dự án**

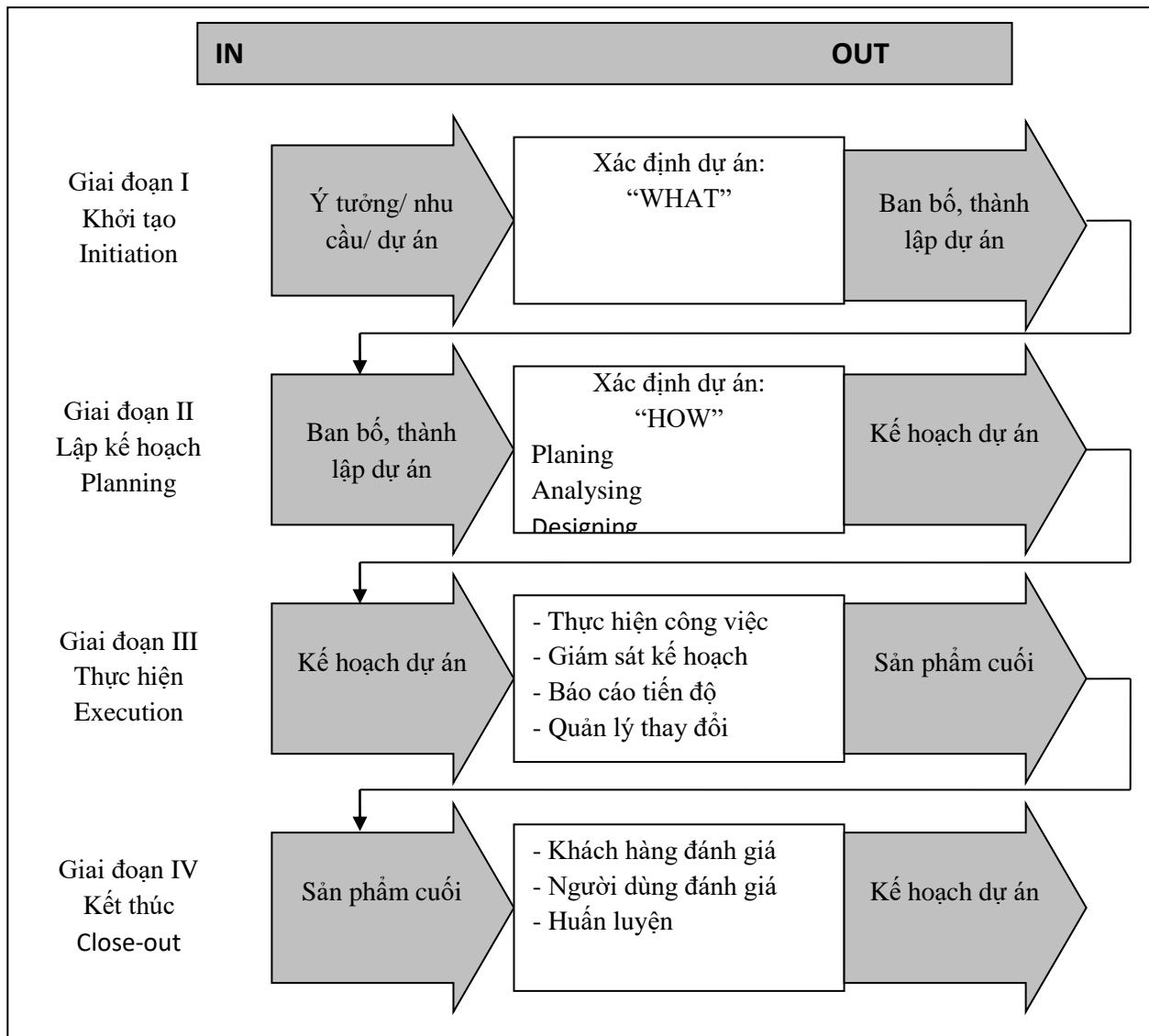
Bất kỳ một dự án nào khi được phát triển / xây dựng cũng đều được chia ra thành các giai đoạn nhỏ để thực hiện. Một điều cũng rõ ràng rằng, một khi dự án được chia thành nhiều giai đoạn nhỏ một cách hợp lý sẽ giúp cho việc quản lý, kiểm soát và điều hành dự án càng dễ dàng và khả năng thành công của dự án sẽ được nâng cao, đặc biệt đối với các dự án phức tạp, dự án lớn.

Đối với các dự án phần mềm, các giai đoạn phát triển của dự án mà chúng ta cũng đã được biết đến như là chu trình sống của dự án – Project Life Cycle. Mỗi một giai đoạn được đánh dấu bởi một hoặc nhiều sản phẩm (Deliverables) hoàn thiện hay cũng có thể gọi là các cột mốc, các khoản mục chuyển giao.

Ví dụ: Kết thúc giai đoạn khảo sát và phân tích yêu cầu sẽ có được sản phẩm là tài liệu đặc tả yêu cầu, đặc tả chức năng của sản phẩm. Sau giai đoạn thiết kế là sản phẩm tài liệu đặc tả chương trình (thiết kế chi tiết) để giai đoạn tiếp theo sử dụng sản phẩm này làm đầu vào.

Từ các giai đoạn phát triển đó chúng ta sẽ xác định giai đoạn nào là chính, là quan trọng nhất trong dự án phần mềm để từ đó người quản trị dự án có được phân bổ nguồn tài nguyên một cách hợp lý nhất.

Sau đây là các giai đoạn quản lý phát triển dự án (nhìn theo MSF hoặc RUP)



#### 4.8.1. Quản lý dự án phần mềm là gì ?

Quản lý dự án phần mềm là các hoạt động trong lập kế hoạch, giám sát và điều khiển tài nguyên dự án (ví dụ như kinh phí, con người), thời gian thực hiện, các rủi ro trong dự án và cả quy trình thực hiện dự án; nhằm đảm bảo thành công cho dự án.

#### 4.8.2. Tam giác quản lý dự án phần mềm

Là một tam giác mà ba cạnh thể hiện ba yếu tố không chế của dự án là: phạm vi công việc (khối lượng và các yêu cầu kỹ thuật), thời gian hoàn thành (tiến độ thực hiện) và ngân sách (mức vốn đầu tư) cho phép. Sự cân đối giữa ba yếu tố này bảo đảm cho tam giác không bị hở ở bất kỳ góc nào chính là thể hiện chất lượng của công tác quản lý dự án. Vì vậy, người ta còn gọi đây là tam giác chất lượng.

#### 4.8.3. Cơ cấu phân chia công việc

Cơ cấu phân chia công việc (work breakdown structure - WBS) là biểu liệt kê (bóc tách, thống kê), trong đó thể hiện đầy đủ các công việc cần thực hiện để dự án được hoàn thành toàn bộ, đạt được các mục tiêu đề ra. Tùy theo mục đích sử dụng mà biểu liệt kê này có phạm vi và mức độ chi tiết khác nhau.

#### **4.9. Lập kế hoạch dự án**

Người quản trị dự án và kỹ sư phần mềm xác định nhân tố con người, máy tính và các tài nguyên tổ chức yêu cầu để phát triển ứng dụng. Kế hoạch dự án chính là sơ đồ các nhiệm vụ, thời gian và các mối quan hệ giữa chúng. Việc lên kế hoạch, nói chung, thường gồm các bước sau:

- Liệt kê các nhiệm vụ: gồm các nhiệm vụ phát triển ứng dụng, các nhiệm vụ đặc trưng của dự án, các nhiệm vụ về tổ chức giao diện, sự xem xét lại và các việc phê chuẩn.
- Định danh phụ thuộc giữa các công việc.
- Xác định nhân viên dựa vào kỹ năng và kinh nghiệm.
- Ấn định thời gian hoàn thành cho mỗi công việc bằng các tính toán thời gian hợp lý nhất cho mỗi công việc.
- Định danh hướng đi tới hạn.
- Xem xét lại các tài liệu theo khía cạnh đầy đủ, nội dung, độ tin cậy và độ chắc chắn.
- Thương lượng, thỏa thuận và cam kết ngày bắt đầu và kết thúc công việc.
- Xác định các giao diện giữa các ứng dụng cần thiết, đặt kế hoạch cho việc thiết kế giao diện chi tiết.

Các nhiệm vụ trong lập kế hoạch dự án thường bao gồm:

- Do tất cả các tài liệu, kế hoạch và công việc của nhóm là phụ thuộc vào người sử dụng, do vậy tổ chức này bao gồm người quản lý, người sử dụng, kiểm toán,...phải đưa các kiến thức chuyên ngành của mình vào những tài liệu ứng dụng một cách thích hợp.
- Cần đạt được sự đồng ý, cam kết từ các ngành, phòng ban bên ngoài trong quá trình cung cấp tài liệu. Bên cạnh đó, bộ phận đảm bảo chất lượng phải xem xét để tìm ra các sai sót và không đồng nhất của tài liệu và tất cả các hoạt động này đều phải đạt kế hoạch.
- Xác định các đòi hỏi về giao diện ứng dụng.
- Đánh giá khối lượng công việc. Thời gian cho mỗi công việc phụ thuộc vào tính phức tạp và mục tiêu của nó - có ba loại thời gian cần tính đến: thời gian bị quan (P), thời gian thực tế (R), thời gian lạc quan (O). Thời gian lịch trình được tính  $= (O+2R+P)/4$ .
- Vấn đề tiếp theo là xác định kỹ năng và kinh nghiệm cần có của người thi hành nhiệm vụ để xác định dùng bao nhiêu người và có kỹ năng gì cho dự án. Sau đó xác định lịch

trình làm việc và người quản lý dự án xác định ngân sách. Ở đây cần có sự trao đổi để hạn chế các trục trặc có thể xảy ra.

- Sau khi hoàn tất, kế hoạch, lịch trình và dự toán ngân sách được đưa cho người sử dụng và người quản lý hệ thống để bổ sung hoặc thông qua.

Chú ý rằng bản kế hoạch không nên đóng cứng, nó có thể thay đổi khi công đoạn nào đó có sự cố xảy ra hoặc thời hạn tỏ ra không phù hợp hay có những thay đổi quan trọng trong mục tiêu của dự án.

#### **4.10. Tổ chức dự án**

Việc tổ chức dự án là rất quan trọng, đây là yếu tố chính quyết định cho sự thành công. Tổ chức dự án bao gồm các hoạt động sau :

- Chọn nhân sự thích hợp.
- Chọn cấu trúc nhóm.
- Chọn kích thước của nhóm.
- Xác định vai trò của các thành viên trong nhóm.
- Quản lý giao tiếp giữa các thành viên.

##### **4.10.1. Chọn nhân sự thích hợp**

Các yếu tố cần xem xét khi chọn nhân sự:

- Kinh nghiệm
  - Hiểu biết lĩnh vực ứng dụng
  - Kinh nghiệm với môi trường phát triển
  - Hiểu biết về ngôn ngữ lập trình
- Đào tạo
- Khả năng
  - Khả năng giao tiếp
  - Khả năng thích ứng
  - Khả năng học hỏi
- Thái độ
- Tính cách

##### **4.10.2. Chọn cấu trúc nhóm**

Cấu trúc nhóm thường chia theo các nhóm sau:

- Nhóm phi hình thức (egoless team)
  - Các thành viên của nhóm có vai trò như nhau.
  - Nhóm có quy mô nhỏ.
  - Các thành viên đều có kinh nghiệm và năng lực.
  - Nhóm được dùng trong những dự án khó.
- Nhóm chief-programmer: gồm có
  - Trưởng nhóm: thực hiện phân tích, thiết kế, mã hóa, kiểm thử.

- Trợ lý: hỗ trợ trưởng nhóm phát triển, kiểm thử.
- Thư ký: quản lý thông tin.
- Các chuyên gia hỗ trợ: quản lý tài liệu, lập trình, kiểm thử.
- Nhóm phụ thuộc chủ yếu vào trưởng nhóm vì thế trưởng nhóm phải là người có năng lực.
- Nhóm phân cấp
  - Dự án được chia thành nhiều dự án nhỏ.
  - Mỗi dự án nhỏ được thực hiện bởi 1 nhóm.
  - Mỗi nhóm có 1 trưởng nhóm.
  - Mỗi thành viên cấp dưới phải báo cáo công việc với người quản lý trực tiếp.
  - Mỗi thành viên phải được đào tạo có kỹ năng để thực hiện vai trò của mình.

#### **4.10.3. Chọn kích thước của nhóm**

Khi chọn kích thước dự án ta cần chú ý các vấn đề sau:

- Kích thước nhóm nên tương đối nhỏ: dưới 8 người để giảm thiểu thời gian giao tiếp và dễ dàng làm việc cùng nhau.
- Nhóm không nên quá nhỏ để đảm bảo tiếp tục làm việc nếu có thành viên ra đi.
- Đối với một số dự án, số người trong nhóm có thể thay đổi.
- Khi một dự án chậm trễ việc thêm người vào dự án không bao giờ giải quyết được vấn đề.

#### **4.10.4. Xác định vai trò của các thành viên trong nhóm**

- Nhà tài trợ dự án: “Người đứng đầu”, chủ sở hữu và nhà tài chính của dự án.
- Giám đốc dự án: đại diện cho nhà tài trợ dự án.
- Ban chỉ đạo: giám sát hoạt động dự án, hỗ trợ trong việc lập phương hướng để đưa ra quyết định.
- Nhóm nghiệp vụ: thừa nhận các kết quả bàn giao chủ yếu về nghiệp vụ.
- Trưởng dự án:
  - Chịu trách nhiệm một dự án.
  - Bảo đảm nhóm có đầy đủ thông tin và nguồn tài nguyên cần thiết.
  - Phân công công việc cho các thành viên.
  - Kiểm tra thời hạn các công việc.
  - Giao tiếp với khách hàng.
- Các thành viên nhóm: hoàn thành nhiệm vụ được giao.

#### **4.10.5. Quản lý giao tiếp giữa các thành viên**

Các đặc điểm trong giao tiếp nhóm:

- Các thành viên có vị trí cao thường áp đặt các cuộc trao đổi.
- Nhóm thường có nam và nữ thường giao tiếp tốt hơn.
- Giao tiếp phải qua một người điều phối trung tâm thường không hiệu quả.



- Tất cả các thành viên nên có tham gia vào các quyết định ảnh hưởng toàn bộ nhóm.
- Tính cách của các thành viên.
  - o Quá nhiều thành viên có cùng tính cách cũng có thể không tốt : hướng công việc – mỗi người đều muốn thực hiện công việc riêng, hướng cá nhân – mỗi người đều muốn làm ông chủ, hướng tương tác – nhiều hợp hành mà ít thực hiện công việc.
  - o Một nhóm nên cân bằng giữa các tính cách.

#### 4.11. Quản lý rủi ro

Rủi ro là những vấn đề chưa bao giờ xảy ra trước đây nhưng có nguy cơ xảy ra trong tương lai.

Quản lý rủi ro là dự đoán các vấn đề có thể xảy ra trong tương lai dự án, phân tích sự ảnh hưởng của các vấn đề này và đưa ra các phương pháp đối phó, kiểm soát khi các vấn đề này thực sự xảy ra. Mục tiêu là tránh “Khủng hoảng”.

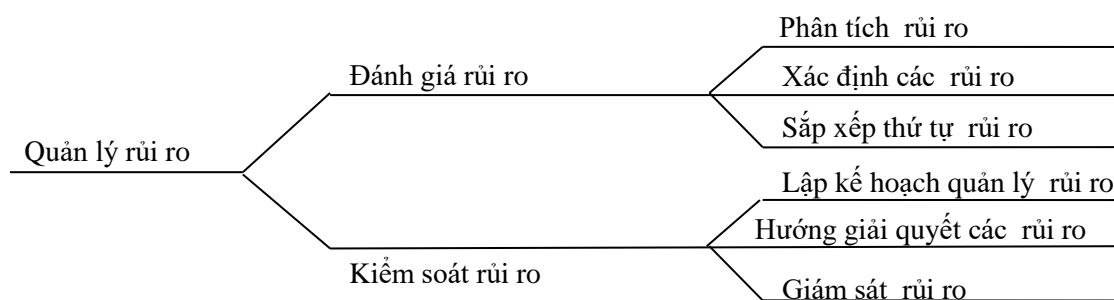
##### *Các đặc điểm của rủi ro dự án:*

- Xác suất xảy ra là  $0 \leq x \leq 1$ .
- Khi rủi ro xảy ra kèm theo sự mất mát tiền của, con người và danh tiếng, uy tín.
- Có một số rủi ro có thể quản lý, kiểm soát được.

##### 4.11.1. Các loại rủi ro

- Rủi ro lập lịch trình: dồn nén lịch trình do khách hàng hoặc do thị trường...
- Rủi ro về chi phí: ngân sách không đảm bảo, vượt quá ngân sách.
- Rủi ro về yêu cầu của sản phẩm:
  - o Các yêu cầu không đúng.
  - o Không hoàn thành các yêu cầu, các yêu cầu không trọn vẹn.
  - o Các yêu cầu không rõ ràng, mâu thuẫn lẫn nhau.
  - o Các yêu cầu hay thay đổi, thay đổi nhanh.
- Rủi ro về chất lượng
- Rủi ro khi vận hành

##### 4.11.2. Quy trình quản lý rủi ro



**Hình 4.7 Quy trình quản lý rủi ro**

Qua quy trình trên ta có thể thấy được các bước cần thực hiện để có thể phát hiện được tối đa các rủi ro như sau:

- Xác định các rủi ro :
  - Cùng với các bên liên quan xác định các rủi ro của dự án.
  - Liệt kê các rủi ro tiềm ẩn có thể phát vỡ kế hoạch thực hiện dự án.
- Phân tích các rủi ro
  - Xác định ảnh hưởng của từng vấn đề rủi ro của dự án.
  - Thông báo các rủi ro cho mọi người đều biết để đối phó, phòng ngừa.
  - Ước lượng kích thước thiệt hại.
  - Ước lượng khả năng xảy ra các tổn thất đó.
- Sắp xếp thứ tự xảy ra của các rủi ro, dựa vào sự phân tích tầm ảnh hưởng và nguy cơ xảy ra mà sắp xếp thứ tự cho các rủi ro theo mức từ cao đến thấp để từ đó có được chiến lược phòng chống các rủi ro khi chúng xảy ra.
  - Có 2 loại rủi ro đó là rủi ro có thể dự đoán được và rủi ro không thể dự đoán được.
  - Lập kế hoạch kiểm soát và quản lý rủi ro tiềm ẩn trong dự án.

#### **4.11.3. Quản lý hiệu quả rủi ro**

Để quản lý hiệu quả rủi ro cần có những chiến lược cụ thể, những sách lược an toàn để đối phó lại với các trường hợp xảy ra bất ngờ.

##### ***Muốn quản lý hiệu quả cần:***

Phòng ngừa hơn là chữa trị.

Đánh giá rủi ro theo thời kỳ trong suốt vòng đời của dự án.

Kết hợp chặt chẽ một quy trình liên tục về xác định rủi ro, phân tích, quản lý và rà xét.

Không đi quá giới hạn và kết thúc không chính xác.

Mức hợp lý của quản lý rủi ro chuẩn sẽ không tốn những nỗ lực vô lý.

##### ***Cần có nhật ký ghi lại các thay đổi, các rủi ro có thể xảy ra:***

***Bảng 4.6 Bảng ghi nhật ký rủi ro***

Mô tả	Độ quan trọng	Người chịu trách nhiệm	Ngày giải quyết
[1]	[2]	[3]	[4]

##### ***Một số vấn đề lưu ý trong quản lý rủi ro:***

Dự án càng lớn thì rủi ro càng nhiều.

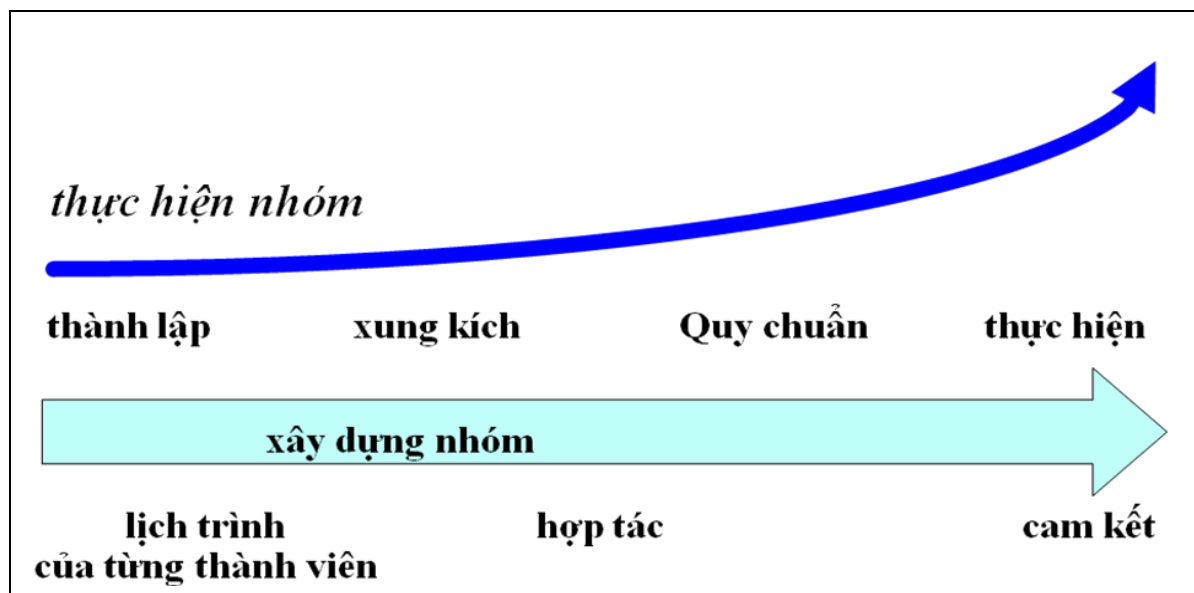
Việc dự báo rủi ro phụ thuộc vào kinh nghiệm quản lý dự án của người PM (người quản lý dự án).

Kiểm soát rủi ro không nhằm loại bỏ rủi ro, chỉ nhằm hạn chế tối thiểu thiệt hại của rủi ro, đặc biệt đối với các rủi ro không lường trước được, không dự đoán trước được.

Không phải cứ tập trung hết sức để ngăn chặn và đề phòng rủi ro đã là tốt, vì có thể phải trả giá đắt nếu rủi ro không xảy ra.

#### 4.12. Phát triển nhóm

Để phát nhóm hiệu quả ta có quy trình phát triển nhóm như sau:



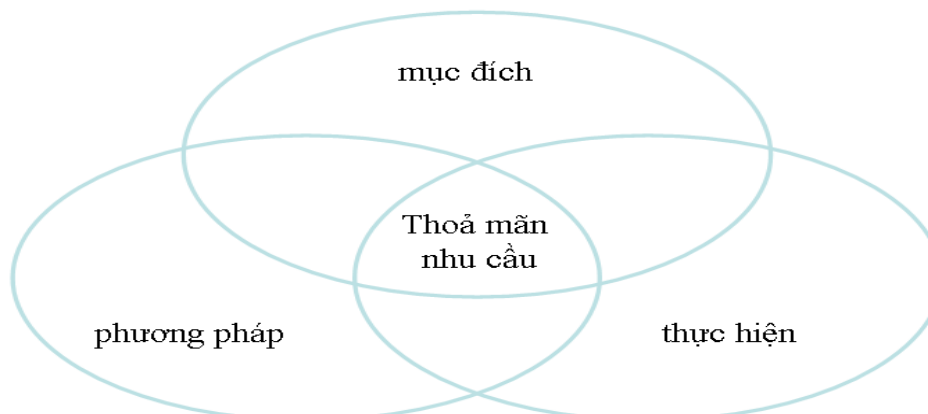
*Hình 4.8. Quy trình xây dựng nhóm*

Một số điểm chú ý khi thành lập nhóm:

- Mục tiêu rõ ràng
- Trách nhiệm rõ ràng
- Thông tin liên lạc hiệu quả
- Phản hồi có tính xây dựng
- Lãnh đạo linh hoạt
- Cán bộ đủ khả năng
- Xác định với nhóm

#### 4.13. Quản lý chất lượng

##### 4.13.1. Biểu đồ biểu diễn sự cân bằng chất lượng



*Hình 4.9. Biểu diễn sự cân bằng về chất lượng sản phẩm*

Chất lượng là :

- Mức độ/cấp độ của sự hoàn hảo, vượt trội.
- Thích hợp cho việc sử dụng.
- Khả năng thỏa mãn nhu cầu của người dùng.
- Có thể đoán trước được.
  - o Tính đồng bộ/ đồng dạng
  - o Đáng tin cậy
  - o Chi phí thấp
  - o Hợp thời
- Được định nghĩa bởi khách hàng.

Vậy chất lượng phần mềm là gì ?

Là sự đáp ứng đúng yêu cầu và sự kì vọng vào khoảng chi phí hợp lý và thời gian cho phép.

Chúng ta thấy rất rõ:

- Đạt được yêu cầu (các yêu cầu của khách hàng được trình bày rõ ràng).
- Đáp ứng được kì vọng của khách hàng .

***Các hoạt động được thực hiện để đạt được chất lượng phần mềm.***

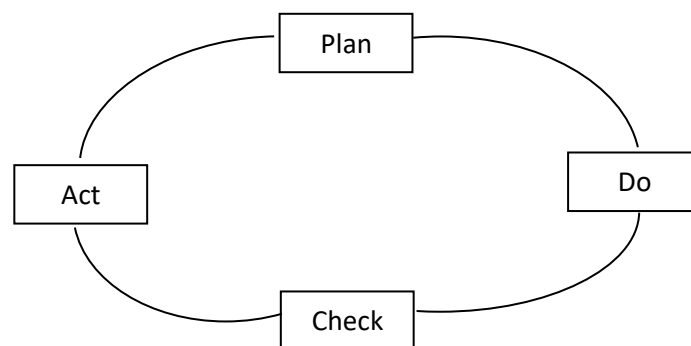
Lập kế hoạch cho các hoạt động chất lượng phần mềm.

Xác định các thước đo.

Triển khai thực hiện các hoạt động đảm bảo chất lượng.

Giám sát sự thành công.

Xác định cải tiến chất lượng theo yêu cầu.



***Hình 4.10. Các hoạt động kiểm soát chất lượng***

- o To PLAN: lập kế hoạch quản lý chất lượng, xác định các thước đo.
- o To DO: triển khai thực hiện theo kế hoạch, theo các tiêu chuẩn hay thước đo chất lượng.

- To CHECK: kiểm tra thực hiện có đúng theo các thước đo, kế hoạch hay tiêu chuẩn chất lượng đã đưa ra.
- To ACT: các hoạt động hiệu chỉnh và phê duyệt.

Ví dụ: Khi chúng ta cài đặt một TV mới vừa mua.

To Plan: ta đưa ra quyết định chọn mua TV cỡ nào, màu sắc và tính năng của nó, sau mua TV mang về.

To Do: ta cài đặt chương trình và đặt nó vào một chỗ ở phòng khách mà bạn đã chọn trước.

To Check: bật nó lên và kiểm tra xem nó hoạt động thế nào.

To Act: thực hiện lần cuối xem chỉnh sửa các tính năng, kênh và vị trí để đảm bảo mọi thứ đều đã thỏa mãn.

#### **4.13.2. Lập kế hoạch quản lý chất lượng.**

##### ***Ở mức lập kế hoạch quản lý, cần quyết định:***

Các tiêu chuẩn, thước đo chất lượng.

Các nhóm có trách nhiệm đối với việc ngừng hoạt động, nếu cần tách nhóm kiểm soát chất lượng và thẩm quyền của họ.

Các hình thức xem xét (không chính thức, chính thức).

Xem xét thường xuyên (ví dụ: tất cả các kết quả chuyển giao theo công việc hoặc chỉ kết quả bàn giao dự án).

##### ***Ở mức độ lập kế hoạch làm việc, cho phép thời gian đối với:***

Kiểm soát và phương pháp quản lý chất lượng.

Thiết lập quy trình quản lý chất lượng.

Thống nhất người (chính xác) sẽ ký nhận:

- Người chịu trách nhiệm
- Giám đốc dự án/ trưởng nhóm
- Đại diện người sử dụng có ảnh hưởng
- Người kiểm soát chất lượng

##### ***Đánh giá kế hoạch chất lượng:***

Kế hoạch quản lý có xác định được các phương pháp, tiêu chuẩn, quy trình và hướng dẫn sử dụng cho từng giai đoạn hoặc hoạt động của dự án không?

Các lý do có cho thấy những điểm này là rõ ràng hợp lý không?

Những tiêu thức kiểm soát được xác định để giám sát hiệu quả có sử dụng các phương pháp đã lựa chọn không?

##### ***Kiểm soát chất lượng***

- Về nội dung.
  - Rà xét, kiểm tra
- Thẩm định tính chấp nhận.

- Rà xét quản lý nhóm
- Thẩm định việc phê chuẩn.
  - Rà xét ban điều hành
- Thẩm định việc triển khai.
  - Quản lý lợi ích
  - Điều tra người sử dụng/các câu hỏi

Phương pháp kiểm soát chất lượng phải được thành lập thành văn bản trong kế hoạch chất lượng. Kế hoạch làm việc chi tiết bao gồm việc thẩm định các nhiệm vụ và các nguồn lực.

#### ***Các hoạt động điều chỉnh***

Các hoạt động điều chỉnh diễn ra :

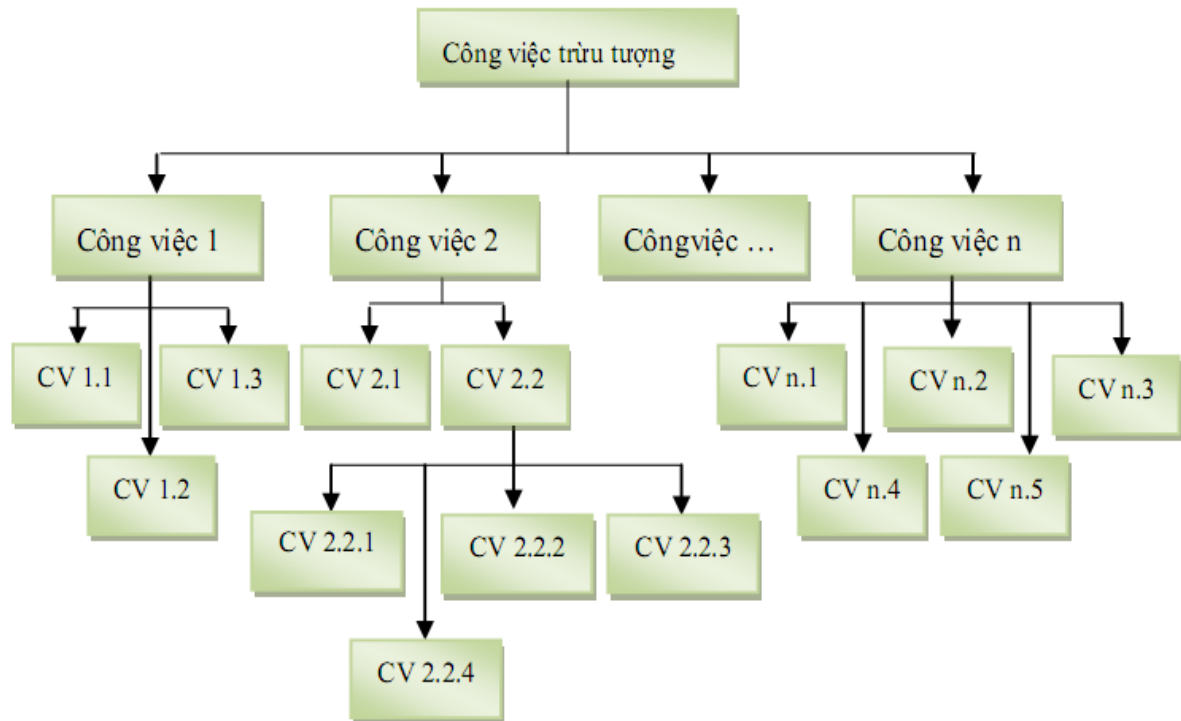
Khi việc thực hiện dự án không diễn ra theo kế hoạch, hoặc chất lượng sản phẩm/công việc chưa đạt yêu cầu.

Khi chi phí cho dự án có nguy cơ tăng lên.

Khi chất lượng công việc/ sản phẩm có nguy cơ giảm

#### **4.14. Lập kế hoạch làm việc chi tiết**

Sau khi đã công bố dự án, trưởng dự án sẽ bắt tay vào việc lên kế hoạch bắt đầu bằng việc xác định các công việc cụ thể và ước lượng tài nguyên cho chúng. Phân rã công việc (Work Breakdown Structure -WBS) công việc được phát biểu ở mức cao, nghĩa là mức khá trừu tượng (what), do đó khó có thể hình dung công việc làm như thế nào (how) để có thể ước lượng được chính xác tối đa tài nguyên cho nó. Vì vậy, để có thể ước lượng được công việc đó làm như thế nào, phải dùng kỹ thuật phân rã công việc hay WBS để cụ thể hóa các công việc của dự án ở mức chi tiết. WBS là một danh sách các công việc ở dạng phân cấp, ở mỗi cấp là chi tiết hóa của công việc ở mức trên, trên cây phân cấp. Đây là kỹ thuật hiện thực hóa một công việc trừu tượng.



**Hình 4.11. Biểu diễn hiện thực hóa một công việc trừu tượng**

Nói cách dễ hiểu hơn, công việc sẽ được phân rã cho tới khi có thể hình dung được nó làm như thế nào, từ đó mới có thể ước lượng tài nguyên, thời gian thực hiện và chi phí.

Một định nghĩa khác, WBS là một công cụ để chi tiết hóa một công việc đến mức có thể ước lượng tài nguyên, thời gian cho công việc đó.

WBS là đầu vào (input) của bản kế hoạch, do đó nếu WBS sai thì bản kế hoạch tiếp theo đó của dự án sẽ bị sai và nguy cơ dự án bị thất bại sẽ rất cao.

WBS có thể được trình bày ở 2 dạng:

- Dạng danh sách: dạng này rất được hay sử dụng. Ví dụ: dự án xây dựng WebSite.

## 1.0 Khảo sát

### 1.1 Khảo sát hệ thống hiện hành

### 1.2 Xác định yêu cầu.

#### 1.2.1 Của người dùng

#### 1.2.2 Về nội dung trang web

#### 1.2.3 Về hệ thống

#### 1.2.4 Của chủ server

### 1.3 Xác định các chức năng

### 1.4 xác định rủi ro và cách tiếp cận quản lý rủi ro.

### 1.5 Lên kế hoạch dự án.

### 1.6 Nhóm phát triển trang web

## 2.0 Thiết kế Web Site.

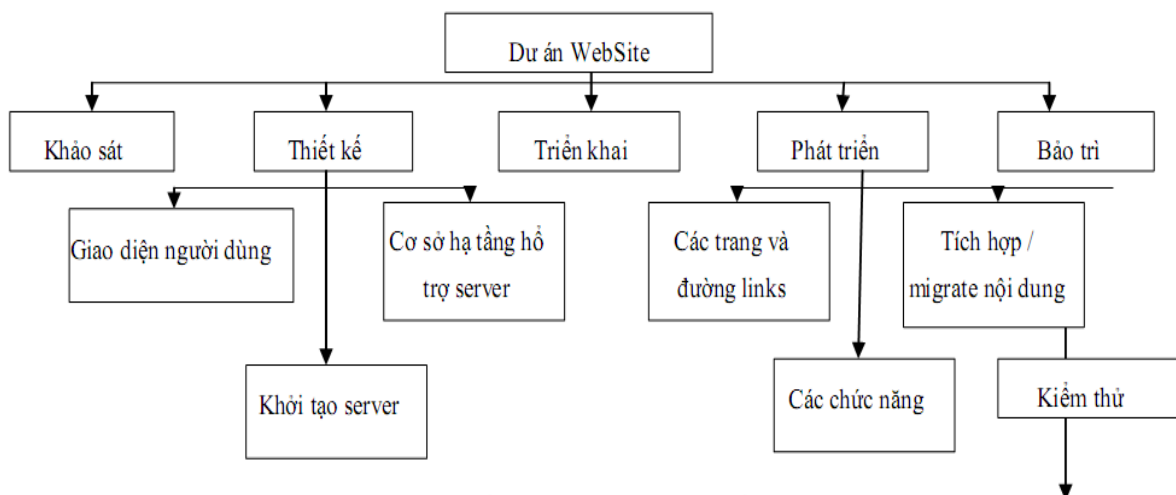
## 3.0 Phát triển Web Site.

## 4.0 Triển khai

## 5.0 Bảo trì

**Hình 4.12: Ví dụ WBS của xây dựng dự án website**

- Dạng đồ họa: dạng này trực quan nhưng tốn chỗ vẽ, do đó chỉ thích hợp cho những dự án nhỏ và đơn giản. Ví dụ: với dự án xây dựng web site của công ty ABC.



**Hình 4.13: Ví dụ dạng đồ họa WBS của xây dựng dự án website**



Phải bảo đảm trong WBS có gồm luôn các công việc:

- Công việc của quản trị dự án: chi phí và tài nguyên cần thiết cho việc quản trị dự án như: lương cho trưởng dự án, văn phòng làm việc, máy tính,..
- Viết tài liệu: hồ sơ phát triển sản phẩm, các kinh nghiệm quản trị dự án này,...
- Cài đặt sản phẩm: huấn luyện người dùng, lên kế hoạch truyền thông, kế hoạch marketing,...
- Đóng dự án: gồm thời gian, chi phí và tài nguyên cần để đóng cửa văn phòng dự án, tái phân công nhân sự và đóng các tài khoản ngân hàng.
- Thu hồi sản phẩm: gồm kế hoạch thu hồi sản phẩm sau khi đã hết vòng đời sử dụng.

#### Lưu ý:

Có thể phân rã theo bất kỳ phạm trù nào miễn là có ý nghĩa cho dự án. Phạm trù đó có thể là các thành phần của sản phẩm, các chức năng, các đơn vị của tổ chức, lãnh thổ địa lý, các chi phí, lịch biểu, hoặc các công việc. WBS thường được phân rã theo công việc. Các ví dụ trên minh họa cho phân rã theo công việc.

Các thành phần được phân rã ở mức cuối cùng – mức lá hay còn gọi là gói công việc, phải thỏa các tiêu chí sau:

1. Tình trạng/tính hoàn tất của công việc có thể đo được.
2. Thời gian, tài nguyên và chi phí dễ ước lượng.
3. Luật 80 giờ: công việc nên thực hiện trong khoảng tối thiểu là 8 giờ, tối đa 80 giờ trong khoảng 2 tuần.
4. Thời gian hoàn thành công việc trong giới hạn cho phép.
5. Công việc được phân công độc lập. Nghĩa là một khi công việc đó được thực hiện thì nó sẽ được thực hiện cho đến hết mà không bị dừng giữa chừng để chờ kết quả của công việc khác. Trong lúc phân rã, nếu có một tiêu chí không thỏa thì phải phân rã tiếp.

Tránh đưa ra một WBS không đủ chi tiết. Nếu công việc được xác định ở mức độ thô (còn mơ hồ) thì việc ước lượng nhân sự, thời gian thực hiện, ngân sách cho công việc sẽ trở nên "phán". Những thông tin được ước lượng đó không đáng tin cậy, nó là nguyên nhân khiến kế hoạch không khớp với thực tế. WBS càng chi tiết, việc lên kế hoạch càng chính xác hơn và khả năng theo dõi quá trình thực hiện tốt hơn. Một phương pháp phổ biến được sử dụng là phương pháp 80 giờ: mỗi công việc thuộc tầng thấp nhất trong WBS phải không được vượt quá 80 giờ lao động. Nếu như công việc cần nhiều giờ hơn, trưởng dự án phải chia nhỏ công việc đó thành những công việc nhỏ hơn. Do đó WBS được liên tục cải tiến khi trưởng dự án ước lượng thời gian thực hiện của công việc.

Khi hoàn tất, trưởng dự án nên cùng với khách hàng và người bảo trợ duyệt lại (review) để chắc chắn rằng WBS là đầy đủ và có đề cập đến các chi tiết quan trọng. Trưởng dự án có thể dùng WBS để thương lượng lại với khách hàng, cấp trên về ngân sách, lịch biểu,...

Các lợi ích khi lên WBS:

- WBS bắt đầu từ dự án, thành viên nhóm, và khách hàng ngồi lại với nhau để phân tích những bước cần thiết trong xây dựng và chuyển giao sản phẩm hay dịch vụ. Điều này tự nó đã khuyến khích một cuộc đối thoại giúp khoanh vùng được phạm vi của dự án; sớm chỉ ra được các vấn đề then chốt; làm rõ các điều còn mơ hồ; phơi bày ra các giả định ngầm - các giả định này phải được liệt kê một cách tường minh bởi chúng chính là nguồn gốc của các hiểm nguy tiềm ẩn.
- Mức độ thấp nhất trong WBS là mức độ gói công việc (work package level). Đó là những công việc lá sẽ được sử dụng để phân công, xây dựng lịch biểu và theo dõi tiến độ. WBS được xem như là một danh sách phân cấp không lồi. Mỗi mức thấp hơn trong danh sách là sự chia nhỏ của các ở mức độ cao hơn.
- Những đối tượng ở mức thấp hơn tạo thành cái ở mức độ cao hơn. Đôi khi mức độ cao hơn của một.
- WBS là mức độ quản lý, các chi tiết được tổng quát lên ở mức độ quản lý để dùng trong mục đích báo cáo. Mức độ thấp hơn được gọi là mức độ kỹ thuật.
- Nó là cơ sở cho việc tạo ra một lịch biểu hiệu quả và một kế hoạch về ngân sách tốt. Một WBS tốt cho phép tài nguyên được cấp cho từng công việc rõ ràng, giúp cho việc đưa ra một lịch biểu hợp lý, và làm cho việc tính toán ngân sách đáng tin cậy hơn.
- Mức độ chi tiết trong WBS giúp nó trở nên hiệu quả hơn trong việc phân công nhân sự với công việc cụ thể, mọi người không thể nấp dưới "vỏ bọc chung chung".
- Một WBS lớn (có hàng ngàn công việc khác nhau) có thể tốn nhiều tuần để thực hiện. Phạm vi dự án càng lớn, WBS càng lớn, một mình trưởng dự án thực hiện sẽ bị thiếu sót và sai rất cao. Do đó trưởng dự án nên mời thêm các thành viên trong nhóm tham gia xây dựng WBS. Sự tham gia này khiến các thành viên ý thức rõ hơn vai trò và trách nhiệm của mình trong dự án.
- WBS phải được cập nhật khi dự án có thay đổi. Một WBS tốt khiến cho việc thực hiện và lên kế hoạch được dễ dàng hơn.

#### **4.15. Kiểm soát và lập báo cáo dự án**

##### **.1.1. Mục tiêu**

Để có thể :

- Chứng minh sự cần thiết trong việc lập báo cáo và kiểm soát dự án hiệu quả và giải thích các lợi ích.
- Để nhận biết các phương pháp và kỹ năng khác nhau có thể được sử dụng cho việc lập báo và kiểm soát dự án.
- Đánh giá tầm quan trọng của việc trình bày các thông tin hiện trạng hợp lý cho các thành phần khác nhau.
- Đánh giá tầm quan trọng của một chu kỳ kiểm soát dự án được xác định.

##### **.1.2. Định nghĩa**

- Kiểm soát dự án: Nắm bắt và quản lý tiến trình.
- Lập báo cáo dự án: Truyền bá hiệu quả những kiến thức này.

### **.1.3. Lập báo cáo và kiểm soát dự án là nền tảng để quản lý dự án.**

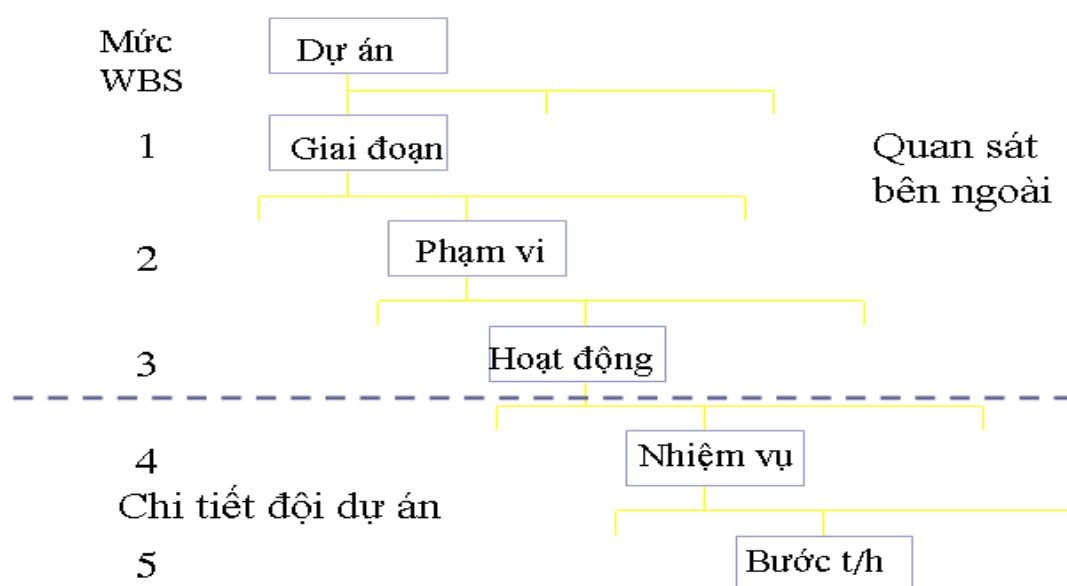
Quản trị viên dự án có thể:

- Báo cáo khách quan về thực trạng dự án.
- Xác định những cản trở và hành động hiệu chỉnh.
- Triển khai các giải pháp.
- Hiểu sự ảnh hưởng của công việc tương lai.
- Đưa ra những quyết định hợp lý dựa trên thông tin xác thực.

Quản trị viên dự án, trưởng nhóm và thành viên nhóm phải:

- Lắng nghe tin nhắn chuyển đến.
- Chấp nhận tin xấu và tốt.
- Hỗ trợ tích cực các thành viên trong nhóm để vượt qua trở ngại.

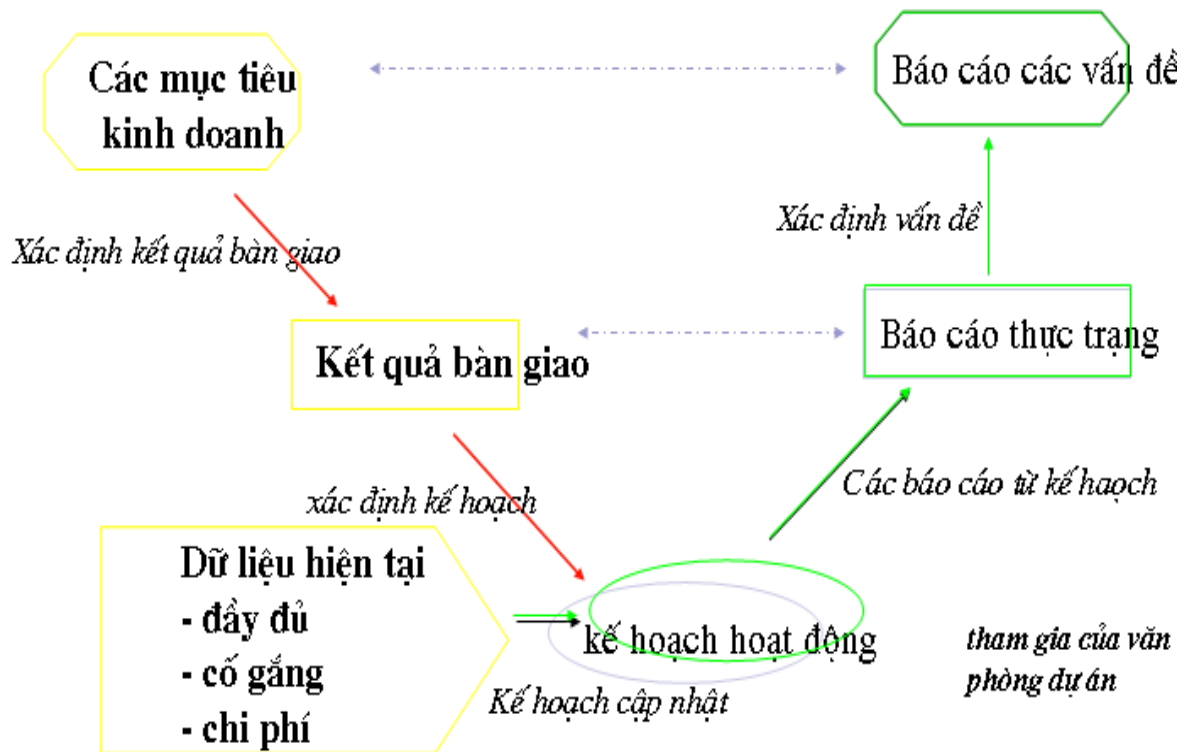
### **.1.4. Lập báo cáo – the WBS**



**Hình 4.14: Mức báo cáo WBS**

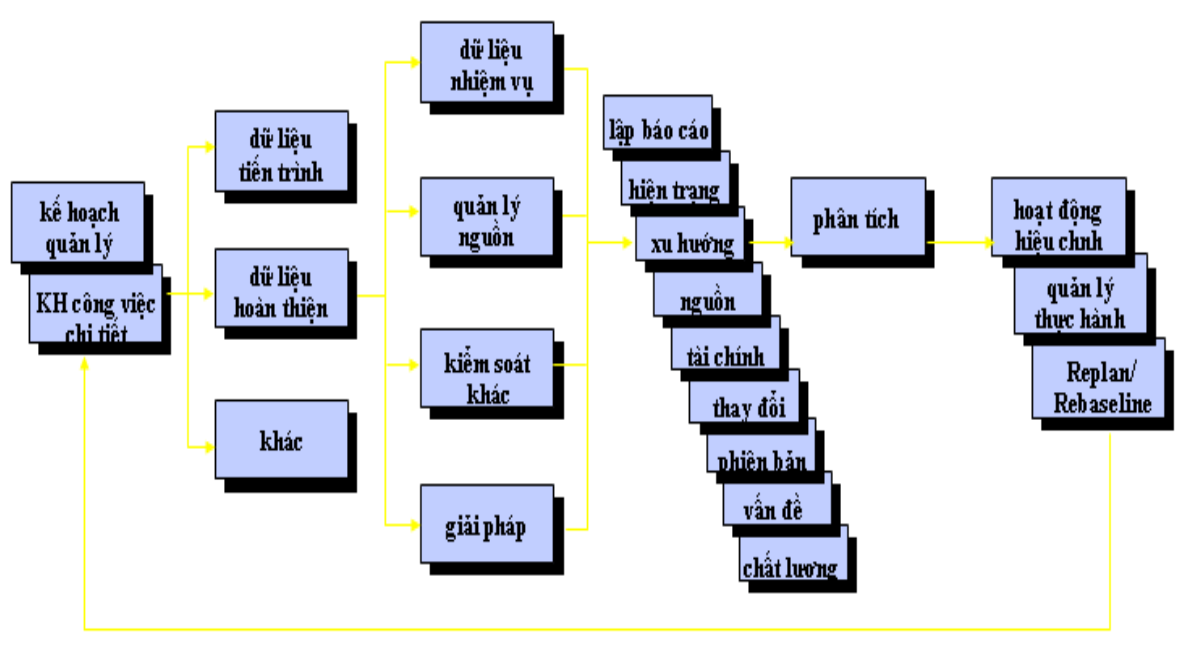
Kế hoạch thực tế, đúng lúc dựa trên thông tin được báo cáo ở mức nhóm, quản lý, điều hành.

Các phương pháp trình bày đa dạng thích hợp với từng đối tượng và công việc được giao.



**Hình 4.15: Mô tả quy trình lập báo cáo, kiểm soát dự án (a)**

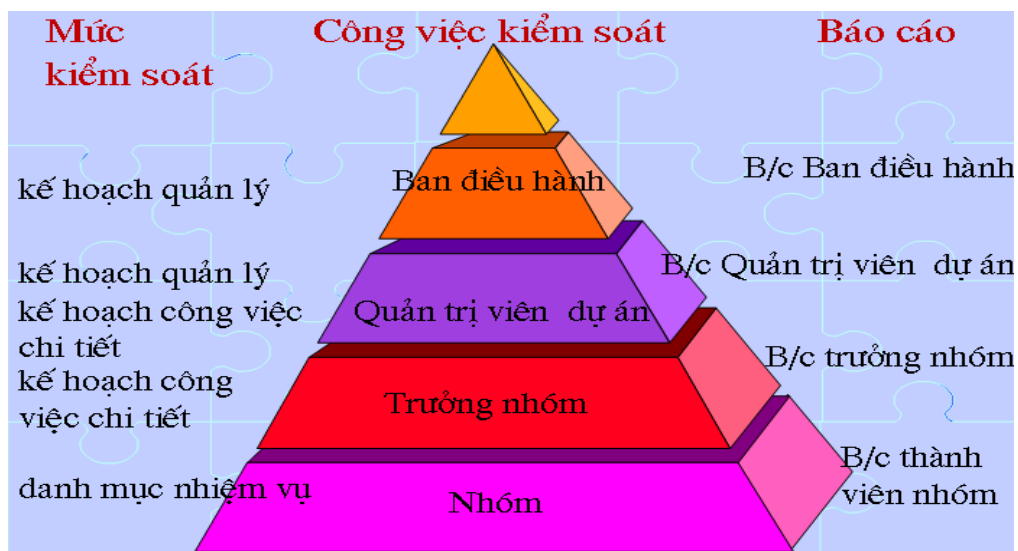
Lập báo cáo và kiểm soát dự án



**Hình 4.16: Mô tả quy trình lập báo cáo, kiểm soát dự án (b)**

### 1.5. Khuôn khổ kiểm soát dự án

Hình vẽ sau cho chúng ta thấy được khuôn khổ/phạm vi của việc kiểm soát dự án. Việc kiểm soát dự án bao gồm các mức kiểm soát, công việc thực hiện kiểm soát thuộc thẩm quyền của những ai và các báo cáo kết quả, báo cáo ghi nhận cho các lãnh đạo cấp trên.



**Hình 4.17. Kiểm soát dự án**

Theo hình vẽ trên, mức đáy là mức thấp nhất và phạm vi của nhóm phát triển dự án thực hiện các công việc của dự án. Báo cáo của nhóm này là các thành viên nhóm phải báo cáo cho trưởng nhóm đó.

Giai đoạn/ mức/phạm vi lập kế hoạch chi tiết dự án thuộc thẩm quyền của Trưởng nhóm sẽ thực hiện báo cáo cho lãnh đạo cấp trên (trên 1 cấp). Và tương tự cho các mức kiểm soát tiếp theo.

## Chương 5. YÊU CẦU PHẦN MỀM

### Mục tiêu

Với sản phẩm phần mềm được xây dựng, việc hiểu đầy đủ các đặc điểm của nó là điều không dễ. Quá trình xác định các chức năng và các ràng buộc của hệ thống gọi là tìm hiểu và xác định yêu cầu. Để có được điều này thì cần phải trả lời câu hỏi "cái gì-what" chứ không phải là "như thế nào-how". Tìm hiểu, xác định và phân tích yêu cầu là bước hình thành bài toán, do vậy các yêu cầu của bài toán cần phải được tìm hiểu và phân tích theo các nguyên lý, tài liệu kỹ thuật và các bước cụ thể. Chương này cung cấp kiến thức tổng quát về yêu cầu phần mềm, kỹ thuật xác định yêu cầu, nội dung xác định yêu cầu, các nguyên lý phân tích yêu cầu

### 5.1. Tổng quát về yêu cầu phần mềm

Những yêu cầu của một hệ thống là những miêu tả về dịch vụ được cung cấp bởi hệ thống và những ràng buộc về chức năng của hệ thống đó. Các yêu cầu này phản ánh mong muốn của khách hàng về một hệ thống giúp giải quyết các vấn đề như kiểm soát một thiết bị, đặt hàng hay tìm kiếm thông tin. Quá trình tìm kiếm thông tin, phân tích, báo cáo và kiểm tra các dịch vụ và các ràng buộc được gọi là xác định yêu cầu (RE). Trong chương này, chúng ta tập trung chính vào những yêu cầu và cách thức để miêu tả chúng.

Thuật ngữ “requirement” không được sử dụng một cách nhất quán trong nền công nghiệp sản xuất phần mềm. Trong một số trường hợp, yêu cầu đơn giản là một khái niệm ở mức độ cao, trừu tượng của một dịch vụ mà hệ thống cung cấp hoặc một ràng buộc của hệ thống. Ở một số trường hợp khác, nó lại là một khái niệm cụ thể về chức năng của một hệ thống. Davis (Davis, 1993) giải thích tại sao những khác biệt này lại tồn tại:

Nếu một công ty muốn ký một hợp đồng với một dự án phát triển phần mềm lớn, họ phải đưa ra những yêu cầu của họ một cách đầy đủ nhưng là theo lối trừu tượng một giải pháp không được định nghĩa trước. Những yêu cầu phải được viết ra để những người đầu thầu có thể bỏ thầu, và họ cũng phải đề nghị những giải pháp khác nhau để đáp ứng yêu cầu của khách hàng. Khi hợp đồng đã được ký kết, người nhận thầu phải viết một mô tả về hệ thống cho khách hàng một cách chi tiết để khách hàng có thể hiểu và có thể phê chuẩn những chức năng tương lai của phần mềm. Tất cả những tài liệu này được gọi là tài liệu xác định yêu cầu của hệ thống.

### 5.2. Kỹ thuật xác định yêu cầu

Một số vấn đề nảy sinh trong quá trình xác định yêu cầu phần mềm là kết quả từ việc xác định sai sự khác nhau rõ ràng giữa các mức độ của việc mô tả. Chúng ta phân biệt sự khác nhau giữa chúng bằng việc sử dụng thuật ngữ “yêu cầu người sử dụng” để chỉ những yêu cầu mức cao và “yêu cầu hệ thống” để chỉ những mô tả cụ thể những gì mà hệ thống có thể thực hiện. Hai khái niệm trên có thể được định nghĩa như sau:

1. ***Yêu cầu người sử dụng*** là những yêu cầu (bằng ngôn ngữ tự nhiên và các sơ đồ) về những dịch vụ mà một hệ thống được mong đợi sẽ cung cấp và những ràng buộc của môi trường chúng được cài đặt.
2. ***Yêu cầu hệ thống*** được xây dựng từ các chức năng, dịch vụ và những ràng buộc một cách cụ thể. Tài liệu xác định yêu cầu hệ thống (thường được gọi là bản mô tả chức năng) yêu cầu thật chính xác. Nó cần miêu tả một cách chính xác những gì sẽ được thực hiện. Nó có thể là một phần trong hợp đồng giữa người dùng và công ty phần mềm.

Ví dụ: Thay đổi hệ thống thư viện.

Định nghĩa yêu cầu người dùng

1. TLIBSYS sẽ theo dõi tất cả các dữ liệu được yêu cầu bản quyền trong UK cũng như bất cứ đâu .

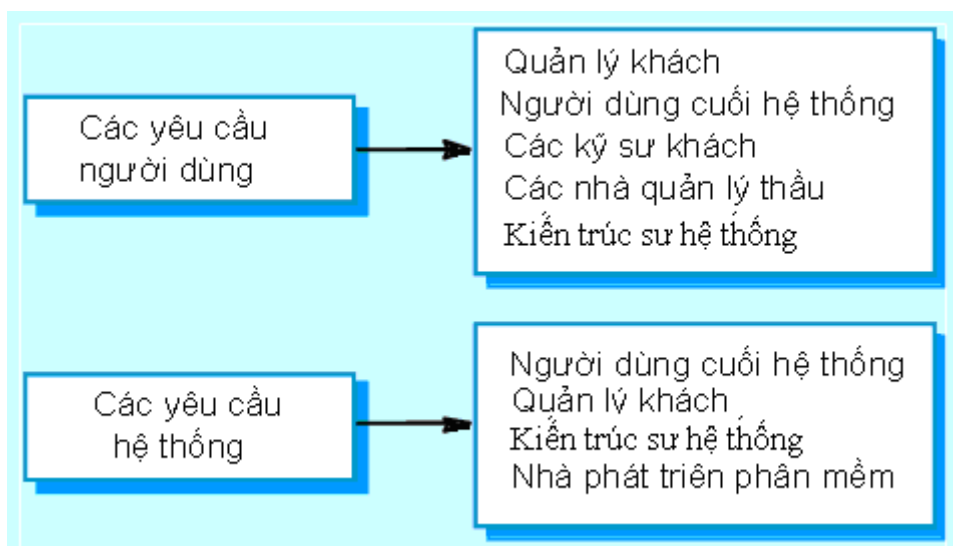
- 1.1 Để thực hiện một yêu cầu cho một bản tài liệu từ LIBSYS, người yêu cầu sẽ được biểu diễn với các dạng báo cáo chi tiết của người dùng
- 1.2 Các mẫu yêu cầu LIBSYS chứa trên hệ thống 5 năm, kể từ ngày được yêu cầu
- 1.3 Tất cả các mẫu yêu cầu của LIBSYS phải được chỉ dẫn bởi người dùng bởi tên của các dạng yêu cầu
- 1.4 LIBSYS sẽ duy trì một bản ghi của tất cả các yêu cầu mà được thực hiện tới hệ thống
- 1.5 ...

### ***Hình 5.1: Yêu cầu người dùng và yêu cầu hệ thống***

Những mức độ khác nhau của bản mô tả chức năng rất có ích bởi chúng tập hợp thông tin về hệ thống cho những người đọc khác nhau. Hình 5.1 minh họa sự khác nhau giữa yêu cầu người sử dụng và yêu cầu hệ thống. Ví dụ với một hệ thống thư viện sẽ chỉ ra cách một yêu cầu người sử dụng được mở rộng thành một số yêu cầu của hệ thống. Bạn có thể thấy từ trong phần này rằng yêu cầu người sử dụng thì ngắn gọn hơn, và yêu cầu hệ thống sẽ có thêm những chi tiết, những lý giải về dịch vụ và chức năng sẽ được cung cấp bởi hệ thống trong tương lai.

Bạn nên viết những yêu cầu ở các mức độ cụ thể khác nhau cho từng đối tượng người đọc khác nhau với các mục đích khác nhau. Hình 5.2 sẽ chỉ ra các loại người đọc khác nhau cho hai loại tài liệu này. Người sử dụng tài liệu xác định yêu cầu người dùng thường không quan tâm đến cách thức hệ thống vận hành và có thể những nhà quản lý-người lại không để ý đến những tiện ích chi tiết của hệ thống. Người sử dụng tài liệu xác định yêu cầu của hệ thống là những người cần biết chính xác những gì mà hệ thống sẽ thực hiện bởi họ quan tâm

đến cách thức nó sẽ hỗ trợ quá trình làm việc hoặc bởi họ liên quan đến sự hoạt động của hệ thống.



**Hình 5.2: Các dạng người đọc**

### 5.3. Nội dung xác định yêu cầu

Yêu cầu của hệ thống phần mềm thường được phân chia thành yêu cầu chức năng, yêu cầu phi chức năng và yêu cầu về phạm vi.

1. Yêu cầu chức năng: Là những yêu cầu về dịch vụ mà hệ thống sẽ cung cấp, cách thức hệ thống trả lời các đầu vào cụ thể và cách thức hệ thống hoạt động trong những trường hợp đặc biệt. Trong một số trường hợp, những yêu cầu chức năng phải xác định rõ ràng những gì mà hệ thống không thể thực hiện.

Ví dụ: "Hệ thống cần lưu trữ tất cả chi tiết về đơn đặt hàng của khách hàng."

2. Yêu cầu phi chức năng: Là những ràng buộc về dịch vụ và chức năng cho hệ thống. Đó là những ràng buộc thời gian, ràng buộc trong quá trình phát triển phần mềm và những tiêu chuẩn. Những yêu cầu này thường áp dụng cho hệ thống một cách tổng thể. Chúng không thường áp dụng cho từng chức năng cụ thể, đặc tính cụ thể của hệ thống. Có thể nhìn nhận yêu cầu phi chức năng là các ràng buộc về các loại giải pháp thỏa mãn các yêu cầu chức năng. Các yêu cầu này mô tả về chính hệ thống, và về việc nó cần thực hiện các chức năng của mình tốt đến mức độ nào, chẳng hạn yêu cầu loại này là yêu cầu về tính sẵn có, khả năng kiểm thử, khả năng bảo trì, và tính dễ sử dụng. Có thể chia các yêu cầu phi chức năng thành hai loại:

1. Ràng buộc về hiệu năng: chẳng hạn "hệ thống cần phục vụ liên tục từ 5 giờ sáng đến 9 giờ tối.", "mỗi đơn đặt hàng cần được lưu trữ trong tối thiểu 7 năm".
2. Ràng buộc về quá trình phát triển hệ thống: thời gian, tài nguyên, chất lượng. Ví dụ: khi nào hệ thống cần hoàn thành (thời gian); tổng chi phí cho phát triển hệ thống (tài nguyên); cần áp dụng các tiêu chuẩn nào cho quá trình phát triển hệ thống, trong đó có các phương pháp quản lý dự án và phát triển hệ thống (chất lượng).



3. Yêu cầu về phạm vi: Là những yêu cầu xuất phát từ những phạm vi ứng dụng của hệ thống, phản ánh các đặc tính và ràng buộc của phạm vi đó. Chúng có thể là những yêu cầu chức năng hoặc phi chức năng.

Trong thực tế, sự khác nhau giữa các loại yêu cầu này không rõ ràng như những định nghĩa trên. Một tài liệu xác định yêu cầu người sử dụng liên quan đến tính bảo mật có thể được coi là một yêu cầu phi chức năng. Tuy nhiên, khi được phát triển cụ thể hơn, yêu cầu này có thể tạo ra những yêu cầu khác là những yêu cầu chức năng, ví dụ như yêu cầu phải có danh sách phân quyền người sử dụng trong hệ thống.

#### **5.4. Các nguyên lý phân tích yêu cầu**

Phần tiếp theo của các tiến trình yêu cầu kỹ thuật là phát hiện và phân tích các yêu cầu. Trong hoạt động này, các kỹ sư phần mềm làm việc với khách hàng và người sử dụng hệ thống cuối để tìm ra lĩnh vực ứng dụng, giúp cho việc chuẩn bị hệ thống, yêu cầu thực hiện hệ thống, các ràng buộc về phần cứng ...

Phát hiện và phân tích các yêu cầu có thể cần phải nhiều người trong nhiều bộ phận khác nhau trong một tổ chức. Thuật ngữ *người liên đới* (stakeholder) được dùng để đề cập đến bất kỳ người nào hay nhóm người nào mà sẽ bị ảnh hưởng bởi hệ thống, theo một cách trực tiếp hoặc gián tiếp. Những *người liên đới* bao gồm người sử dụng cuối mà tương tác với hệ thống và bất kỳ người nào trong một tổ chức chịu sự ảnh hưởng của sự tương tác đó. Nhóm những *stakeholder* có thể là các kỹ sư phát triển và duy trì sự liên kết các hệ thống, hay là những người quản lý kinh doanh, các chuyên gia về lĩnh vực này và những người đại diện của các công đoàn.

Phát hiện và hiểu được các yêu cầu của những stakeholder là rất khó khăn bởi các lý do sau :

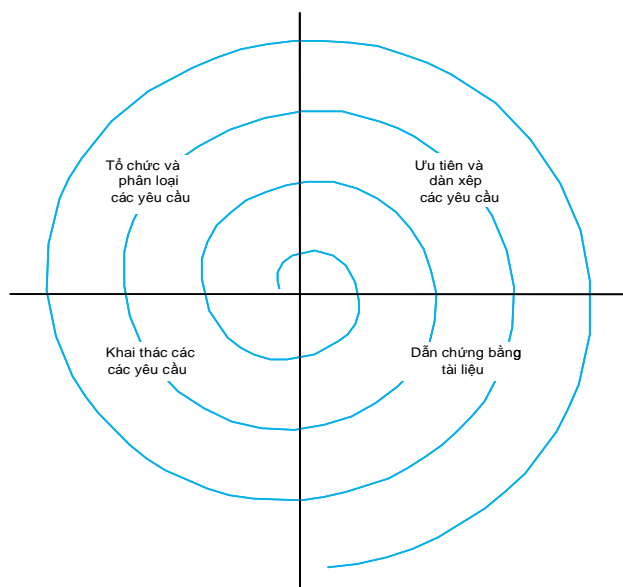
1. Những *stakeholder* thường không biết họ muốn gì từ hệ thống máy tính trừ các trường hợp thông thường. Họ có thể khó khăn trong việc tìm một cách nói cho rõ ràng về những gì mà họ muốn hệ thống thực hiện những yêu cầu không chân thực bởi vì họ không ý thức được về chi phí của các yêu cầu của họ.
2. Những *stakeholder* sẽ tự nhiên bày tỏ các yêu cầu của mình trong giới hạn của họ và với các hiểu biết ngầm trong công việc của họ. Điều này yêu cầu các kỹ sư công nghệ dù không nhiều kinh nghiệm trong lĩnh vực của khách hàng vẫn phải hiểu những yêu cầu trên.
3. Những stakeholder khác nhau sẽ có những yêu cầu khác nhau, điều mà họ có thể bày tỏ bằng nhiều cách khác nhau. Điều này yêu cầu các kỹ sư phải xem xét tất cả các yêu cầu tiềm tàng có thể xảy ra và khai thác các điểm tương đồng hay xung khắc giữa chúng.

4. Những người quản lý nhân sự có thể ảnh hưởng đến các yêu cầu của hệ thống. Ví dụ như người quản lý có thể chỉ định yêu cầu hệ thống để tăng ảnh hưởng của họ trong tổ chức.
5. Nền kinh tế và môi trường kinh doanh luôn luôn biến động, nó chắc chắn sẽ thay đổi suốt trong quá trình phân tích. Vì lý do đó mà các yêu cầu riêng có thể bị thay đổi. Các yêu cầu mới sẽ xuất hiện từ những stakeholder mới.

Một mô hình quá trình phổ biến của việc phát hiện và phân tích các yêu cầu được biểu diễn như **Hình 5.3**. Mỗi tổ chức sẽ có các phiên bản hay cài đặt của mô hình trên, phụ thuộc vào các nhân tố cục bộ như sự thành thạo của nhân viên, dạng của hệ thống đang được phát triển và các chuẩn mực sử dụng. Trở lại một chút bạn xem các hoạt động trên trong vòng xoắn ốc mà xen kẽ nhau như các quá trình xuất phát từ trong ra ngoài của một vòng xoắn ốc.

Quá trình các hoạt động là :

1. *Khai thác các yêu cầu* : Đây là quá trình tương tác với những stakeholder trong hệ thống để thu thập các yêu cầu của họ. Phạm vi các yêu cầu của những stakeholder và các tài liệu cũng sẽ được khai thác suốt trong hoạt động.
2. *Phân loại và tổ chức các yêu cầu* : Đây là hoạt động phân loại tập các yêu cầu ,nhóm các yêu cầu có liên hệ với nhau và tổ chức chúng trong một nhóm có liên hệ chặt chẽ.
3. *Sự ưu tiên và dàn xếp các yêu cầu* : Một điều chắc chắn là trong nhiều loại stakeholder khác nhau các yêu cầu sẽ bị xung đột. Hoạt động này đề cập đến việc ưu tiên các yêu cầu, tìm và giải quyết xung đột giữa các yêu cầu.
4. *Dẫn chứng bằng tài liệu* : Các yêu cầu sẽ được báo cáo bằng tài liệu và đưa vào vòng tiếp theo của hình xoắn ốc. Các tài liệu về các yêu cầu trang trọng hay không trang trọng sẽ được công bố.



**Hình 5.3** *Quá trình phân tích và phát hiện các yêu cầu.*

**Hình 5.3** biểu diễn phát hiện và phân tích các yêu cầu như một quá trình lặp với các thông tin phản hồi liên tục từ mỗi hoạt động tới các hoạt động khác. Một chu kì quá trình bắt đầu với việc khai thác các yêu cầu và kết thúc với việc hoàn chỉnh tư liệu các yêu cầu. Hiểu biết của các nhà phân tích được phát triển qua từng chu kì.

Trong chương này, chúng ta tập trung trước tiên vào việc khai thác các yêu cầu và các kĩ thuật khác nhau được dùng để hỗ trợ việc này. Sự ưu tiên và dàn xếp các yêu cầu là mối quan tâm trước tiên với việc nhận dạng các yêu cầu chồng chéo nhau từ những stakeholder khác nhau và nhóm các yêu cầu có quan hệ lại với nhau. Cách nhóm các yêu cầu phổ biến nhất là sử dụng mô hình của hệ thống kiến trúc để nhận dạng các hệ thống con và liên kết các yêu cầu với mỗi hệ thống con. Điều này nhấn mạnh rằng công nghệ xác định các yêu cầu và thiết kế kiến trúc là không thể tách rời.

Một điều chắc chắn là những stakeholder có cách nhìn khác nhau về tầm quan trọng và ưu thế của các yêu cầu, và cũng có lúc các cách nhìn đó là xung đột nhau. Trong một quá trình, bạn nên tổ chức những stakeholder liên tục để có thể liên lạc tới được. Tất nhiên là không thể thỏa mãn mọi khách hàng, nhưng nếu một vài stakeholder cảm thấy cách nhìn của họ không được xem xét, họ có thể sẽ cố gắng phá hoại quá trình công nghệ xác định các yêu cầu.

Trong công đoạn làm tư liệu về các yêu cầu, các yêu cầu sẽ được phát hiện để làm tài liệu theo cách mà chúng có thể được sử dụng để giúp đỡ việc khai thác các yêu cầu tương lai. Trong công đoạn này, một phiên bản sớm của hệ thống các tư liệu về các yêu cầu có thể sẽ được tạo ra, nhưng nó sẽ bỏ qua một số mục và chưa hoàn thành các yêu cầu. Một cách chọn khác là các yêu cầu sẽ được soạn thảo thành các bảng trong tài liệu hay trên các tấm phiếu. Viết các yêu cầu trên phiếu có thể rất hiệu quả, và dễ dàng cho những stakeholder cầm tay, tay đổi hay sắp xếp.

### 5.5 Tài liệu đặc tả yêu cầu

Kết quả của bước phân tích là tạo ra bản đặc tả yêu cầu phần mềm (Software Requirement specification - SRS). Đặc tả yêu cầu phải chỉ rõ được phạm vi của sản phẩm, các chức năng cần có, đối tượng người sử dụng và các ràng buộc khi vận hành sản phẩm. Có nhiều chuẩn khác nhau trong xây dựng tài liệu, dưới đây là một định dạng RSR thông dụng (theo chuẩn IEEE 830 1984).

- **Giới thiệu**

- **Mục đích**

*Mục này giới thiệu mục đích của tài liệu yêu cầu. Thường chỉ đơn giản là định nghĩa “đây là tài liệu yêu cầu về phần mềm XYZ”.*

- **Phạm vi**

Nêu phạm vi đề cập của tài liệu yêu cầu.

- **Định nghĩa:**

Định nghĩa các khái niệm, các từ viết tắt, các chuẩn được sử dụng trong tài liệu yêu cầu.

- **Tài liệu tham khảo.**

Nêu danh mục các tài liệu tham khảo dùng để tạo ra bản đặc tả yêu cầu.

- **Mô tả chung về tài liệu.**

Mô tả khái quát cấu trúc tài liệu, gồm có các chương, mục, phụ lục chính nào.

- **Mô tả chung.**

- **Tổng quan về sản phẩm**

Mô tả khái quát về sản phẩm

- **Chức năng sản phẩm.**

Khái quát về chức năng sản phẩm.

- **Đối tượng người dùng.**

Mô tả đối tượng người dùng.

- **Ràng buộc tổng thể.**

Khái quát về các ràng buộc của phần mềm: ràng buộc phần cứng, môi trường sử dụng, yêu cầu kết nối với các hệ thống khác,...

- **Giả thiết về sự lệ thuộc.**

Mô tả các giả thiết khi áp dụng tài liệu: ví dụ như tên phần cứng, phần mềm, hệ điều hành cụ thể.

- **Yêu cầu chi tiết.**

Mô tả các yêu cầu

- **Yêu cầu chức năng**

Mô tả chi tiết về các yêu cầu chức năng.

- **Yêu cầu chức năng 1.**

- **Giới thiệu**

- **Dữ liệu vào**

- **Xử lý**

- **Kết quả**

- **Yêu cầu chức năng 2.**

....

- **3.1.n. Yêu cầu chức năng thứ n**

- **Yêu cầu giao diện ngoài.**

Mô tả giao diện của phần mềm với môi trường bên ngoài

- **Giao diện người dùng**

- *Giao diện phần cứng*
- *Giao diện phần mềm*
- *Giao diện truyền thông*
- *Yêu cầu hiệu suất*

*Mô tả về hiệu suất, ví dụ như thời gian phản hồi với sự kiện, số giao dịch được thực hiện/ giây,...*

- *Ràng buộc thiết kế*

*Mô tả các ràng buộc thiết kế, ví dụ các ràng buộc về ngôn ngữ, về công nghệ, về cơ sở dữ liệu và về chuẩn giao tiếp.*

- *Thuộc tính*

*Mô tả các thuộc tính của phần mềm*

- *Tính bảo mật, an toàn và khả năng phục hồi*

*Mức độ bảo mật dữ liệu, cách thức truy cập vào hệ thống. Độ an toàn của hệ thống đối với các trường hợp bất thường như mất điện...Khả năng phục hồi của hệ thống sau khi gặp sự cố.*

- *Tính bảo trì.*

*Các chức năng, giao diện đòi hỏi phải dễ sử dụng (dễ bảo trì)*

- *Các yêu cầu khác*

*Các yêu cầu khác liên quan đến sản phẩm.*

## **5.6.Các bước phân tích và đặc tả yêu cầu**

### **5.6.1 Phân tích bài toán**

Mô tả nghiệp vụ

- Mô tả các luồng nghiệp vụ, các xử lý và vai trò của con người trong hệ thống hiện tại.
- Hiểu được nghiệp vụ.
- Chủ yếu tập trung vào các vùng cần tự động hóa.
- Hỗ trợ cho việc xác định các thay đổi và cải tiến yêu cầu trong hệ thống mới.

Mô tả hệ thống

- Mô tả luồng thông tin giữa hệ thống đề xuất và môi trường của nó.
- Đáp ứng được mô tả nghiệp vụ.
- Cải tiến nghiệp vụ hiện tại.
- Dựa trên mô tả nghiệp vụ hiện tại.

### **5.6.2 Thu thập yêu cầu**

Khi một công ty muốn ký một hợp đồng cho một dự án phát triển một phần mềm, công ty sẽ phát biểu các yêu cầu ở mức trừu tượng để không bắt buộc định nghĩa trước các giải pháp.

Các yêu cầu phải được viết sao cho các nhà phát triển phần mềm có thể đưa ra các giải pháp khác nhau. Sau khi đã trúng thầu và ký hợp đồng, yêu cầu phải được làm rõ hơn để khách hàng có thể hiểu và đánh giá được phần mềm. Cả hai tài liệu nói trên đều gọi là tài liệu yêu cầu người dùng.

Theo mức độ chi tiết có thể chia ra các loại tài liệu yêu cầu:

3. Xác định yêu cầu: đây là một khẳng định, bằng ngôn ngữ tự nhiên hơn là các sơ đồ, về các dịch vụ hệ thống cần cung cấp và các ràng buộc mà hệ thống phải tuân theo. Tài liệu này cung cấp cho các thành phần: người quản lý của bên khách hàng, người dùng cuối của hệ thống, kỹ sư của khách hàng, người quản lý ký kết hợp đồng, các kiến trúc sư hệ thống.

4. Đặc tả yêu cầu: là tài liệu được cấu trúc mô tả hệ thống các dịch vụ chi tiết hơn. Đôi khi tài liệu này được gọi là đặc tả chức năng. Đây có thể coi là hợp đồng ký kết giữa người mua và kẻ bán phần mềm. Tài liệu này cung cấp cho các thành phần: người dùng cuối của hệ thống, kỹ sư của khách hàng, các kiến trúc sư hệ thống, người phát triển phần mềm.

5. Đặc tả phần mềm: là mô tả trừu tượng hơn của phần mềm làm cơ sở cho thiết kế và triển khai. Tài liệu này cung cấp cho các thành phần: kỹ sư của khách hàng, các kiến trúc sư hệ thống, người phát triển phần mềm.

6. Xác định yêu cầu: là mô tả trừu tượng các dịch vụ mà hệ thống được mong đợi phải cung cấp và các ràng buộc mà hệ thống phải tuân thủ khi vận hành. Nó chỉ có các đặc tả sản phẩm bên ngoài của hệ thống mà không liên quan đến các đặc tính thiết kế. Nó phải được viết sao cho người ta có thể hiểu được mà không cần một kiến thức chuyên môn đặc biệt nào.

### 5.6.3 Phân tích yêu cầu

Nghiên cứu kỹ các yêu cầu của người sử dụng và của hệ thống phần mềm để xây dựng các đặc tả về hệ thống là cần thiết, nó sẽ xác định hành vi của hệ thống.

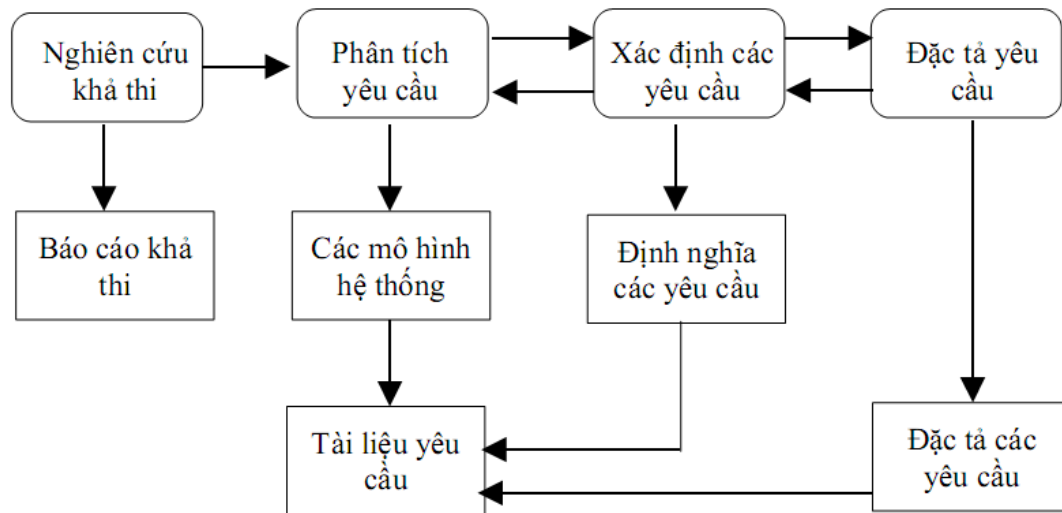
Nhiệm vụ của giai đoạn này là phải trả lời được các câu hỏi sau:

- a. Đầu vào của hệ thống là gì.
- b. Những quá trình cần xử lý trong hệ thống, hay hệ thống phần mềm sẽ phải xử lý những cái gì.
- c. Đầu ra: kết quả xử lý của hệ thống là gì.
- d. Những ràng buộc trong hệ thống, chủ yếu là mối quan hệ giữa đầu vào và đầu ra như thế nào.

Trả lời được câu hỏi trên, nghĩa là phải xác định được chi tiết các yêu cầu làm cơ sở để đặc tả hệ thống. Đó là kết quả của sự trao đổi, thống nhất giữa người đầu tư, người sử dụng với người xây dựng hệ thống. Mục tiêu là xây dựng các hồ sơ mô tả chi tiết các yêu cầu của bài toán nhằm nêu bật được hành vi, chức năng cần thực hiện của hệ thống dự kiến.

Như vậy, phân tích yêu cầu là quá trình suy luận các yêu cầu hệ thống thông qua quan sát hệ thống hiện tại, thảo luận với các người sử dụng, phân tích công việc.

Việc này có thể liên quan với việc tạo một hay nhiều mô hình khác nhau. Nó giúp các phân tích viên hiểu biết hệ thống. Các mẫu hệ thống cũng có thể được phát triển để mô tả các yêu cầu. Ta có quy trình để có các chức năng của hệ thống.



Trong quá trình phân tích cần lưu ý đến tính khả thi của dự án.

7. Khả thi về kinh tế: chi phí phát triển phải cân xứng với lợi ích mà hệ thống đem lại, gồm có:

a. Chi phí

- i. Mua sắm: thiết bị, vật tư (phần cứng), tư vấn, cài đặt thiết bị, quản lý và phục vụ,...
- ii. Chi phí cho khởi công: phần mềm phục vụ cho hệ thống, hệ thống liên lạc (truyền dữ liệu), nhân sự ban đầu: đào tạo - huấn luyện, cải tổ tổ chức cho phù hợp,...
- iii. Chi phí liên quan: chi phí nhân công phục vụ thu nhập dữ liệu, sửa đổi, cập nhật hệ thống, chuẩn bị tài liệu,...
- iv. Chi phí liên tục là tốn kém nhất, gồm: bảo trì, thuê bao, khấu hao phần cứng, chi phí phục vụ cho vận hành,...

b. Lợi nhuận do sử dụng hệ thống

- i. Nhiệm vụ xử lý thông tin: giảm chi phí do xử lý tự động, tăng độ chính xác và kết quả tốt hơn, thời gian trả lời rút ngắn,...
- ii. Có được từ hệ thống: thu thập và lưu trữ dữ liệu tự động, đầy đủ, dữ liệu được chuẩn hóa, bảo đảm an toàn và an ninh dữ liệu, tương thích và chuyển đổi giữa các bộ phận, truy cập và tìm kiếm nhanh, kết nối và trao đổi diện rộng,

8. Khả thi về kỹ thuật: đây là vấn đề cần lưu ý vì các mục tiêu, chức năng và hiệu suất của hệ thống theo một cách nào đó là còn "mơ hồ" do vậy xét:

- a. Rủi ro xây dựng: các phần tử hệ thống (chức năng, hiệu suất) khi thiết kế và phân tích có tương đương hay không?
  - b. Có sẵn tài nguyên: có sẵn con người và tài nguyên cần thiết để phát triển hệ thống?
  - c. Công nghệ: các công nghệ liên quan cho việc phát triển hệ thống đã có sẵn hay chưa?
9. Khả thi về hợp pháp: có sự xâm phạm, vi phạm hay khó khăn nào gây ra khi xây dựng hệ thống hay không?
10. Các phương án: đánh giá về phương án tiếp cận đến việc xây dựng hệ thống.

#### 5.6.4 Đặc tả yêu cầu

Khi đã xác định rõ bài toán thì bước tiếp theo là tìm hiểu xem hệ thống dự kiến sẽ yêu cầu làm cái gì. Điều quan trọng ở đây là phải xây dựng được danh sách các yêu cầu của người sử dụng. Dựa trên những yêu cầu của người sử dụng, người phát triển đưa ra các đặc tả cho hệ thống.

Người xây dựng hệ thống phải trả lời được các yêu cầu sau đây:

- a. Đầu ra của hệ thống là cái gì.
- b. Hệ thống sẽ phải làm cái gì để có kết quả mong muốn, nghĩa là phải xử lý những cái gì.
- c. Những tài nguyên mà hệ thống yêu cầu là gì.

Hiểu rõ nguồn gốc, các dạng thông tin cần cung cấp cho hệ thống hoạt động. Hệ thống sẽ phải giải quyết những vấn đề gì, những kết quả cần phải có là gì. Xác định được mối quan hệ giữa cái vào và cái ra cho quá trình hoạt động của hệ thống. Các đặc tả chi tiết phục vụ cho việc xây dựng và trải nghiệm về hệ thống để kiểm tra xem những nhiệm vụ đã đặt ra có hoàn tất được hay không.

Ở đây, chúng ta cần chú ý là trong một số trường hợp, sẽ nảy sinh những yêu cầu mới mà có thể là ta phải xây dựng lại hệ thống, tất nhiên điều này sẽ làm chậm tiến trình xây dựng và làm tăng giá thành do một vài lý do để không thể hoàn chỉnh các đặc tả đối với các hệ thống như:

- d. Các hệ thống phần mềm lớn luôn đòi hỏi cải tiến từ hiện trạng. Mặc dù các khó khăn của hệ thống hiện tại có thể xác định được nhưng các ảnh hưởng và hiệu ứng của hệ thống mới khó có thể dự đoán trước được.
- e. Hệ thống lớn thường có nhiều cộng đồng sử dụng khác nhau. Họ có các yêu cầu và ưu tiên khác nhau. Các yêu cầu hệ thống cuối cùng không tránh khỏi các thỏa hiệp.
- f. Người trả tiền cho hệ thống và người sử dụng thường khác nhau. Các yêu cầu đưa ra do ràng buộc của các tổ chức và tài chính có thể tranh chấp với yêu cầu của người sử dụng.



Do các đặc tả yêu cầu thêm các thông tin vào định nghĩa yêu cầu nên các đặc tả thường được biểu diễn cùng với các mô hình hệ thống được phát triển trong quá trình phân tích yêu cầu. Nó cần bao gồm mọi thông tin cần thiết về yêu cầu chức năng và ràng buộc của hệ thống. Phân tích yêu cầu được tiếp tục xác định và đặc tả khi các yêu cầu mới nảy sinh. Đây là tài liệu thường xuyên thay đổi và nên được kiểm soát chặt chẽ.

Ngôn ngữ tự nhiên không hoàn toàn thuận tiện cho các thiết kế viên hoặc các hợp đồng giữa các khách hàng và cán bộ phát triển hệ thống vì có một số lý do như sau:

- g. Nhầm lẫn do cách hiểu các khái niệm khác nhau giữa hai bên.
- h. Đặc tả yêu cầu ngôn ngữ tự nhiên quá mềm dẻo. Một vấn đề có thể được mô tả bằng quá nhiều cách khác nhau.
- i. Các yêu cầu không được phân hoạch tốt, khó tìm các mối quan hệ,...

Do vậy người ta thường dùng các thay thế khác để đặc tả các yêu cầu như:

- j. Ngôn ngữ tự nhiên có cấu trúc,
- k. Ngôn ngữ mô tả thiết kế, giống ngôn ngữ lập trình nhưng có mức trừu tượng cao hơn,
- l. Ngôn ngữ đặc tả yêu cầu,
- m. Ghi chép graphic,
- n. Đặc tả toán học,...

Có thể chia đặc tả yêu cầu ra làm hai loại: đặc tả phi hình thức (ngôn ngữ tự nhiên) và đặc tả hình thức (dựa trên kiến trúc toán học).

#### 1. Đặc tả phi hình thức

Đặc tả phi hình thức là đặc tả sử dụng ngôn ngữ tự nhiên. Tuy nó không được chặt chẽ bằng đặc tả hình thức nhưng được nhiều người biết và có thể dùng để trao đổi với nhau để làm chính xác hóa các điểm chưa rõ, chưa thống nhất giữa các bên phát triển hệ thống.

#### 2. Đặc tả hình thức

Đặc tả hình thức là đặc tả mà ở đó các từ ngữ, cú pháp, ngữ nghĩa được định nghĩa hình thức dựa vào toán học. Đặc tả hình thức có thể coi là một phần của hoạt động đặc tả phần mềm. Các đặc tả yêu cầu được phân tích chi tiết. Các mô tả trừu tượng của các chức năng chương trình có thể được tạo ra để làm rõ yêu cầu.

Đặc tả phần mềm hình thức là một đặc tả được trình bày trên một ngôn ngữ bao gồm: từ vựng, cú pháp và ngữ nghĩa được định nghĩa. Định nghĩa ngữ nghĩa đảm bảo ngôn ngữ đặc tả không phải là ngôn ngữ tự nhiên mà dựa trên toán học. Các chức năng nhận các đầu vào trả lại các kết quả. Các chức năng có thể định ra các điều kiện tiên tố và hậu tố. Điều kiện tiên tố là điều kiện cần thỏa mãn để có dữ liệu vào, điều kiện hậu tố là điều kiện cần thỏa mãn sau khi có kết quả.

Có hai hướng tiếp cận đặc tả hình thức để phát triển các hệ thống tương đối phức tạp

+ Tiếp cận đại số, hệ thống được mô tả dưới dạng các toán tử và các quan hệ.

+ Tiếp cận mô hình, mô hình hệ thống được cấu trúc sử dụng các thực thể toán học như là các tập hợp và các thứ tự.

Sử dụng đặc tả hình thức, ta có các thuận lợi:

- o. Cho phép chúng ta thấy và hiểu được bản chất bên trong của các yêu cầu, đây là cách tốt nhất để làm giảm các lỗi, các thiếu sót có thể xảy ra và giúp cho công việc thiết kế được thuận lợi.
- p. Do chúng ta sử dụng toán học cho việc đặc tả nên có thể dựa vào các công cụ toán học khi phân tích và điều này làm tăng thêm tính chắc chắn và tính đầy đủ của hệ thống.
- q. Đặc tả hình thức, bản thân nó cho chúng ta một cách thức cho việc kiểm tra hệ thống sau này.

Tuy vậy, đặc tả hình thức cũng bộc lộ một vài khó khăn:

- r. Quản lý phần mềm có tính bảo thủ cố hữu của nó, không sẵn sàng chấp nhận các kỹ thuật mới.
- s. Chi phí cho việc đặc tả hình thức thường cao hơn so với các đặc tả khác (tuy phần cài đặt sẽ thấp hơn), nên khó để chứng minh rằng chi phí tương đối cao cho đặc tả sẽ làm giảm tổng chi phí dự án.
- t. Phần lớn, những người đặc tả hệ thống không được đào tạo một cách chính quy về việc sử dụng đặc tả hình thức cho việc đặc tả hệ thống mà dựa trên thói quen của họ.
- u. Thông thường, nhiều thành phần của hệ thống là khó cho việc đặc tả bằng ngôn ngữ hình thức. Thêm vào đó là khách hàng không thể hiểu được nó.
- v. Khách hàng không thích các đặc tả toán học.

### 5.6.5 Hợp thức hóa yêu cầu

Các yêu cầu hệ thống được trình bày trong tài liệu các yêu cầu phần mềm cho biết những thứ cần bộ phát triển hệ thống cần biết. Tài liệu này bao gồm các định nghĩa về yêu cầu và các đặc tả về các yêu cầu. Trong một số trường hợp, chúng không được trình bày riêng biệt mà được tích hợp làm một. Đôi khi, định nghĩa yêu cầu được trình bày như là một giới thiệu tới đặc tả yêu cầu. Cách tiếp cận hiệu quả nhất là trình bày các đặc tả chi tiết như là phụ lục của yêu cầu.

Tài liệu yêu cầu phần mềm không phải tài liệu đặc tả. Nó cần phải mô tả cái hệ thống cần phải làm chứ không phải làm thế nào. Tài liệu này cần dễ dàng được đặc tả và ánh xạ sang các phản tương ứng của thiết kế hệ thống. Nếu các dịch vụ, ràng buộc và các đặc tả thuộc tính trong tài liệu yêu cầu phần mềm được thỏa mãn bởi thiết kế thì thiết kế này được coi là giải pháp thích hợp với vấn đề.

Về nguyên tắc, các yêu cầu cần được hoàn chỉnh và chắc chắn. Mọi chức năng hệ thống cần được đặc tả và các yêu cầu không được mâu thuẫn. Tuy nhiên các thiếu sót là không thể

tránh khỏi, do vậy tài liệu nên được cấu trúc dễ cho việc thay đổi. Nội dung nên được chia thành các chương. Sáu yêu cầu cần được thỏa mãn là :

11. Nó cần mô tả các hành vi hệ thống bên ngoài.
12. Nó cần mô tả các ràng buộc về thực hiện.
13. Nó cần phải dễ thay đổi.
14. Nó phải là công cụ tham chiếu cho người bảo trì hệ thống.
15. Nó cần ghi được vòng đời của hệ thống.
16. Nó cần biểu thị được các đáp ứng chấp nhận được với các sự kiện không dự kiến.

Cấu trúc chung của tài liệu yêu cầu phần mềm gồm các phần như sau:

17. Giới thiệu mô tả sự cần thiết của hệ thống. Nó cần sự mô tả sơ lược các chức năng của mình và giải thích cách làm việc với các hệ thống khác. Nó cũng cần mô tả làm thế nào hệ thống đáp ứng được toàn bộ các mục tiêu chiến lược và nghiệp vụ.

18. Thuật ngữ: nó cần định nghĩa các khái niệm kỹ thuật được sử dụng trong tài liệu này. Không được giả định người đọc đã có kinh nghiệm.

19. Mô hình hệ thống: phần này lập một hoặc nhiều mô hình hệ thống cho biết các quan hệ giữa các cấu thành hệ thống với hệ thống và môi trường của nó. Nó cần bao gồm các mô hình đối tượng, mô hình luồng dữ liệu và ngữ nghĩa dữ liệu.

20. Định nghĩa yêu cầu chức năng: các dịch vụ cung cấp cho người dùng cần được mô tả trong mục này. Mô tả có thể dùng ngôn ngữ tự nhiên, sơ đồ hoặc các dạng ghi chép khác cho phép khách hàng có thể hiểu được. Các dịch vụ cung cấp cho người dùng cần được mô tả trong mục này. Mô tả có thể dùng ngôn ngữ tự nhiên, sơ đồ hoặc các dạng ghi chép khác cho phép khách hàng có thể hiểu được.

21. Định nghĩa yêu cầu phi chức năng: các ràng buộc về phần mềm và các hạn chế đối với thiết kế cần phải được mô tả trong phần này. Nó có thể bao gồm các chi tiết của biểu diễn dữ liệu, thời gian đáp ứng và yêu cầu bộ nhớ,...Các tiêu chuẩn về sản phẩm và quy trình cần tuân thủ cũng được mô tả.

22. Tiến triển hệ thống: phần này mô tả các giả thiết căn bản làm cơ sở cho hệ thống và dự đoán các thay đổi về phát triển phần cứng, yêu cầu người dùng.

23. Đặc tả yêu cầu: mô tả các yêu cầu cơ bản chi tiết hơn. Nếu cần các chi tiết hơn có thể được thêm vào các yêu cầu phi chức năng, ví dụ giao diện với các hệ thống có thể được định nghĩa.

24. Ngoài ra, tài liệu yêu cầu phần mềm có thể bao gồm thêm các phần sau:

- a. Phần cứng: nếu hệ thống được phát triển trên một phần cứng đặc biệt, phần cứng này và giao diện cần được mô tả. Nếu phần cứng bán sẵn được sử dụng, các cấu hình cực tiểu và cực đại phải được mô tả.
- b. Yêu cầu dữ liệu: tổ chức logic của dữ liệu được sử dụng bởi hệ thống và các quan hệ giữa chúng được mô tả, có thể dùng sơ đồ thực thể liên kết.

- c. Chỉ mục có thể được cung cấp. Ví dụ chỉ mục theo chữ cái, chỉ mục theo chương, theo chức năng....

Do hệ thống được vận hành trong thời gian dài, nên môi trường hệ thống và mục đích nghiệp vụ có thể thay đổi. Khi đó tài liệu yêu cầu cũng cần phải thay đổi.

Với mục đích tiến triển, tài liệu yêu cầu thường được chia theo hai phân loại:

- d. Các yêu cầu ổn định: được suy dẫn từ các hoạt động cốt lõi của tổ chức tương đối liên quan trực tiếp tới miền hệ thống.
- e. Các yêu cầu bất thường: các yêu cầu có thể thay đổi khi phát triển hệ thống sau này như: các yêu cầu xuất hiện như là sự hiểu biết của khách hàng về sự phát triển của hệ thống trong quá trình xây dựng hệ thống, các yêu cầu được sinh ra do sự xuất hiện của việc tin học hóa làm thay đổi các quy trình nghiệp vụ,...

## **Chương 6. PHƯƠNG PHÁP THIẾT KẾ HỆ THỐNG**

### **Mục tiêu**

Xây dựng ứng dụng phần mềm là một dây chuyền các chuyển đổi, mà ở đó phân tích nhằm xác định ứng dụng sẽ thực hiện cái gì (what) còn thiết kế nhằm để trả lời câu hỏi phần mềm cụ thể sẽ như thế nào (how)? Tức là xác định cách thức thực hiện những gì đã được đặt ra ở phần phân tích.

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

Chương 6 và 7 này sẽ cung cấp kiến thức về khái niệm thiết kế hệ thống và một số phương pháp thiết kế hệ thống cũng như giới thiệu một vài công cụ hiện đang được sử dụng để hỗ trợ việc phân tích thiết kế hệ thống

### **6.1. Khái niệm thiết kế hệ thống**

Thiết kế hệ thống là một quá trình áp dụng nhiều kỹ thuật và các nguyên lý để tạo ra mô hình của một thiết bị, một tiến trình hay một hệ thống đủ chi tiết mà theo đó có thể chế tạo ra sản phẩm vật lý tương ứng với nó.

Bản chất thiết kế phần mềm là một quá trình chuyển hóa các yêu cầu phần mềm thành một biểu diễn thiết kế. Từ những mô tả quan niệm về toàn vẹn phần mềm, việc làm mịn (chi tiết hóa) liên tục dẫn tới một biểu diễn thiết kế rất gần với cách biểu diễn của chương trình nguồn để có thể ánh xạ vào một ngôn ngữ lập trình cụ thể.

### **6.2. Phương pháp thiết kế hệ thống**

Do các hệ phần mềm lớn là phức tạp nên người ta thường dùng các phương pháp tiếp cận khác nhau trong việc thiết kế các phần khác nhau của một hệ thống. Chẳng có một chiến lược tốt nhất nào cho các dự án. Hai chiến lược thiết kế hiện đang được dùng rộng rãi trong việc phát triển phần mềm đó là thiết kế hướng chức năng và thiết kế hướng đối tượng. Mỗi chiến lược thiết kế đều có ưu và nhược điểm riêng phụ thuộc vào ứng dụng phát triển và nhóm phát triển phần mềm.

Cách tiếp cận hướng chức năng hay hướng đối tượng là bổ sung và hỗ trợ cho nhau chứ không phải là đối kháng nhau. Kỹ sư phần mềm sẽ chọn cách tiếp cận thích hợp nhất cho từng giai đoạn thiết kế.

### **6.2.1. Thiết kế hướng chức năng**

Thiết kế hướng chức năng là một cách tiếp cận thiết kế phần mềm trong đó bản thiết kế được phân giải thành một bộ các đơn thể được tác động lẫn nhau, mà một đơn thể có một chức năng được xác định rõ ràng. Các chức năng có các trạng thái cục bộ nhưng chúng chia sẻ với nhau trạng thái hệ thống, trạng thái này là tập trung và mọi chức năng đều có thể truy cập được.

Có người nghĩ rằng thiết kế hướng chức năng đã lỗi thời và nên được thay thế bởi cách tiếp cận hướng đối tượng. Thế nhưng, nhiều tổ chức đã phát triển các chuẩn và các phương pháp dựa trên sự phân giải chức năng. Nhiều phương pháp thiết kế kết hợp với các công cụ CASE đều là hướng chức năng và có nhiều hệ thống đã được phát triển bằng cách sử dụng phương pháp tiếp cận hướng chức năng. Các hệ thống đó sẽ phải được bảo trì cho một tương lai xa xôi. Bởi vậy thiết kế hướng chức năng vẫn sẽ còn được tiếp tục sử dụng rộng rãi, đó là của thiên hạ. Còn người Việt Nam chúng ta, chúng ta chưa có tập quán dùng một phương pháp thiết kế nào, liệu chúng ta có nhất thiết phải đi theo trào lưu đó hay chúng ta nên đi thẳng vào phương pháp nào hữu hiệu nhất?

Người ta dùng các biểu đồ dòng dữ liệu - mà nó mô tả việc xử lý dữ liệu logic, các lược đồ cấu trúc - nó chỉ ra cấu trúc của phần mềm và các mô tả PDL (page description language) - nó mô tả thiết kế chi tiết. Khái niệm dòng dữ liệu đã bị cải biên làm cho nó thích hợp hơn việc sử dụng một hệ thống vẽ biểu đồ tự động và sử dụng một dạng lược đồ cấu trúc có kèm thêm các thông tin điều khiển.

Chiến lược thiết kế hướng chức năng dựa trên việc phân giải hệ thống thành một bộ các chức năng có tương tác nhau với trạng thái hệ thống tập trung dùng chung cho các chức năng đó. Các chức năng này có thể có các thông tin trạng thái cục bộ nhưng chỉ dùng cho quá trình thực hiện chức năng đó mà thôi.

Thiết kế hướng chức năng gắn với các chi tiết của một thuật toán của chức năng đó nhưng các thông tin trạng thái hệ thống là không bị che dấu. Điều này có thể gây ra một vấn đề vì rằng một chức năng có thể thay đổi trạng thái theo một cách mà các chức năng khác

không ngờ tới. Việc thay đổi một chức năng và cách nó sử dụng trạng thái hệ thống có thể gây ra những tương tác bất ngờ đối với các chức năng khác.

Do đó cách tiếp cận chức năng để thiết kế là thắng lợi nhất khi mà khối lượng thông tin trạng thái hệ thống là được làm nhỏ nhất và thông tin dùng chung nhau là rõ ràng.

### **6.2.2. Thiết kế hướng đối tượng**

Hệ thống được nhìn nhận như một bộ các đối tượng (chứ không phải là bộ các chức năng). Hệ thống được phân tán, mỗi đối tượng có những thông tin trạng thái riêng của nó. Đối tượng là một bộ các thuộc tính xác định trạng thái của đối tượng đó và các phép toán của nó. Nó được thừa kế từ một vài lớp đối tượng lớp cao hơn, sao cho dễ định nghĩa nó chỉ cần nêu đủ các khác nhau giữa nó và các lớp cao hơn nó.

Che dấu thông tin là chiến lược thiết kế dấu càng nhiều thông tin trong các thành phần càng hay. Cái đó ngầm hiểu rằng việc kết hợp điều khiển logic và cấu trúc dữ liệu được thực hiện trong thiết kế càng chậm càng tốt. Liên lạc thông qua các thông tin trạng thái dùng chung (các biến tổng thể) là ít nhất, nhờ vậy khả năng hiểu là được tăng lên. Thiết kế là tương đối dễ thay đổi vì sự thay đổi một thành phần không thể không dự kiến các hiệu ứng phụ trên các thành phần khác.

Thiết kế hướng đối tượng là dựa trên việc che dấu thông tin, nhìn hệ phần mềm như là một bộ các đối tượng tương tác với nhau chứ không phải là một bộ các chức năng như cách tiếp cận chức năng. Các đối tượng này có một trạng thái được che dấu và các phép toán trên các trạng thái đó. Thiết kế biểu thị các dịch vụ được yêu cầu và được cung cấp bởi các đối tượng có tương tác với nó.

Thiết kế hướng đối tượng có ba đặc trưng:

i) Vùng dữ liệu dùng chung là bị loại bỏ. Các đối tượng liên lạc với nhau bằng cách trao đổi thông báo chứ không phải bằng các biến dùng chung.

ii) Các đối tượng là các thực thể độc lập mà chúng sẵn sàng được thay đổi vì rằng tất cả các trạng thái và các thông tin biểu diễn là chỉ ảnh hưởng trong phạm vi chính đối tượng đó thôi. Các thay đổi về biểu diễn thông tin có thể được thực hiện không cần sự tham khảo tới các đối tượng hệ thống khác.

iii) Các đối tượng có thể phân tán và có thể hành động tuần tự hoặc song song. Không cần có quyết định về tính song song ngay từ một giai đoạn sớm của quá trình thiết kế.

#### Ưu điểm:

i) Dễ bảo trì vì các đối tượng là độc lập. Các đối tượng có thể hiểu và cải biến như là một thực thể độc lập. Thay đổi trong thực hiện một đối tượng hoặc thêm các dịch vụ sẽ không làm ảnh hưởng tới các đối tượng hệ thống khác.

ii) Các đối tượng là các thành phần dùng lại được thích hợp (do tính độc lập của chúng). Một thiết kế có thể dùng lại được các đối tượng đã được thiết kế trong các bản thiết kế trước đó.

iii) Đối với một vài lớp hệ thống, có một phản ánh rõ ràng giữa các thực thể có thực (chẳng hạn như các thành phần phần cứng) với các đối tượng điều khiển nó trong hệ thống. Điều này cải thiện được tính dễ hiểu của thiết kế.

#### Nhược điểm:

Sự nhận minh các đối tượng hệ thống thích hợp là khó khăn. Cách nhìn tự nhiên nhiều hệ thống là cách nhìn chức năng và việc thích nghi với cách nhìn hướng đối tượng đôi khi là khó khăn.

Phương pháp thiết kế hướng đối tượng vẫn còn là tương đối chưa chín mùi và đang thay đổi mau chóng.

Ở đây, cần phân biệt hai khái niệm là thiết kế hướng đối tượng và lập trình (cài đặt) hướng đối tượng:

+ Thiết kế hướng đối tượng là một chiến lược thiết kế nó không phụ thuộc vào một ngôn ngữ thực hiện cụ thể nào. Các ngôn ngữ lập trình hướng đối tượng và các khả năng bao gói đối tượng làm cho thiết kế hướng đối tượng được thực hiện một cách đơn giản hơn. Tuy nhiên một thiết kế hướng đối tượng cũng có thể được thực hiện trong một ngôn ngữ kiểu như Pascal hoặc C mà không có các đặc điểm như vậy.

+ Việc chấp nhận thiết kế hướng đối tượng như là một chiến lược hữu hiệu đã dẫn đến sự phát triển phương pháp thiết kế hướng đối tượng. Như Ada không phải là ngôn ngữ lập trình hướng đối tượng vì nó không trợ giúp sự thừa kế của các lớp, nhưng lại có thể thực hiện các đối tượng trong Ada bằng cách sử dụng các gói hoặc các nhiệm vụ, do đó Ada được dùng để mô tả các thiết kế hướng đối tượng.

+ Thiết kế hướng đối tượng là một chiến lược thiết kế, nó không phụ thuộc vào ngôn ngữ để thực hiện. Các ngôn ngữ lập trình hướng đối tượng và khả năng bao gói dữ liệu làm cho dễ thực hiện một thiết kế hướng đối tượng hơn. Tuy nhiên cũng có thể thực hiện một thiết kế hướng đối tượng trong một ngôn ngữ kiểu như Pascal hoặc C.

#### **Câu hỏi:**

1. Khái niệm thiết kế hệ thống ?
2. Phân biệt thiết kế hướng đối tượng và thiết kế hướng chức năng, so sánh ưu và nhược điểm của cả hai.
3. Áp dụng kiến thức của chương cho phần mềm các anh chị đang nghiên cứu, lựa chọn phương án hướng đối tượng hay hướng chức năng, vì sao ?

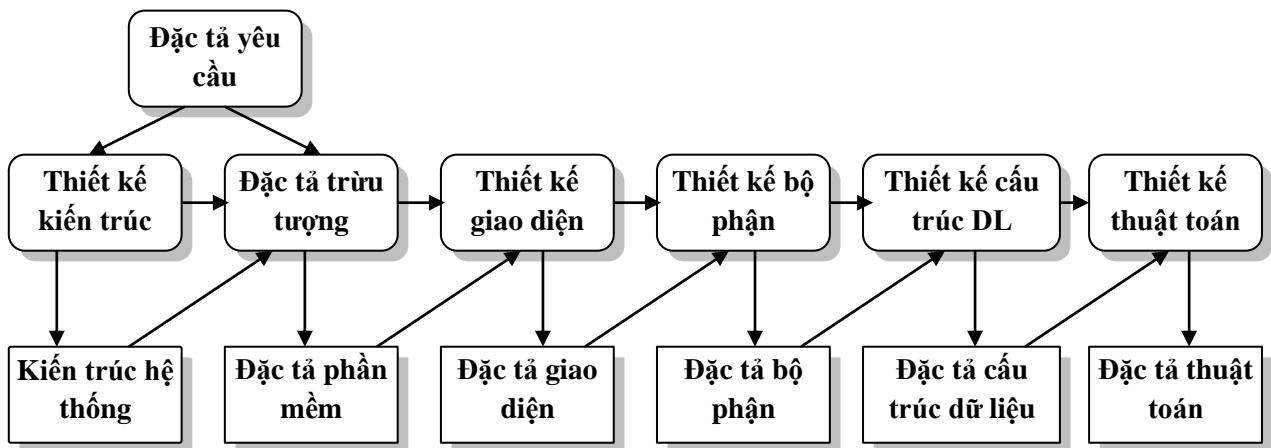
## Chương 7. KỸ THUẬT THIẾT KẾ PHẦN MỀM

### Mục tiêu

Chương 7 này sẽ cung cấp kiến thức về khái niệm thiết kế phần mềm và một số phương pháp thiết kế cũng như giới thiệu một vài công cụ hiện đang được sử dụng để hỗ trợ việc phân tích thiết kế hệ thống

### 7.1. Khái niệm thiết kế phần mềm

Thiết kế phần mềm là một sự mô tả cấu trúc phần mềm được đưa vào sử dụng, dữ liệu - một phần của hệ thống, và đôi khi là giao diện giữa những bộ phận cấu thành hệ thống và những thuật toán được sử dụng. Không chỉ dừng lại ở việc hoàn tất bản thiết kế, những nhà thiết kế còn tiếp tục phát triển thiết kế của mình thông qua hàng loạt phiên bản. Quá trình thiết kế có liên quan đến việc bổ sung thêm những hình thức và chi tiết mới bởi những bản thiết kế được phát triển từ sự phản hồi liên tục trong khi sửa chữa những thiết kế cũ.



**Hình 7.1 Mô hình của một quy trình thiết kế phổ biến**

Quá trình thiết kế cũng có thể liên quan đến việc phát triển một số mẫu hệ thống với sự trừu tượng ở mức độ khác nhau. Những lỗi và những chi tiết bị bỏ sót trong những khâu trước sẽ được phát hiện ra khi thiết kế được chia nhỏ ra. Những sự phản hồi này làm cho những mẫu thiết kế trước trở nên hoàn thiện hơn. Hình 4.7 là một ví dụ về quá trình thiết kế này, nó mô tả thiết kế ở những khâu khác nhau. Sơ đồ này cho thấy các khâu của quá trình thiết kế luôn kế tiếp nhau. Thực tế những hoạt động của quá trình thiết kế luôn xen kẽ nhau. Sự hoàn ngược từ khâu này sang khâu khác và thiết kế hoạt động là hoàn toàn không thể tránh khỏi trong mọi quy trình thiết kế.

Đặc điểm kỹ thuật của khâu tiếp theo là đầu ra của mỗi hoạt động thiết kế. Đặc điểm này có thể là một sự tách biệt ra, có thể là một chi tiết mang tính hình thức để làm sáng tỏ những yêu cầu, hoặc cũng có thể là một đặc điểm kỹ thuật mà thông qua nó, một phần của hệ thống được hiện thực hoá. Những đặc điểm kỹ thuật này trở nên chi tiết hơn khi quá trình thiết kế đang diễn ra. Kết quả cuối cùng của quá trình này là những đặc điểm kỹ thuật chính xác, tỉ mỉ của những thuật toán và cấu trúc dữ liệu được đưa vào sử dụng.

### 7.2. Phương pháp thiết kế phần mềm



Những hoạt động cụ thể của quá trình thiết kế bao gồm:

1. *Thiết kế kiến trúc*: Các hệ thống con tạo nên hệ thống và những mối quan hệ giữa chúng, được xác định và chứng minh. Chủ đề quan trọng này sẽ được nhắc lại ở Chương 11, 12 và 13.
2. *Đặc tả trừu tượng*: Với mỗi hệ thống con, một đặc tả trừu tượng các dịch vụ của nó và các ràng buộc dưới các hoạt động được sản xuất.
3. *Thiết kế giao diện*: Với mỗi hệ thống con, giao diện của nó với những hệ thống con khác được thiết kế và tài liệu hoá. Đặc tả giao diện không được lưỡng nghĩa, mơ hồ bởi nó phải đảm bảo rằng các hệ thống con phải được sử dụng ngay cả khi không biết về hoạt động của các hệ thống con này. Những phương pháp về đặc tả mang tính hình thức, như được thảo luận ở Chương 10, có thể được sử dụng ở khâu này.
4. *Thiết kế thành phần*: Các dịch vụ của hệ thống phải được phân phối tới các thành phần và giao diện của chúng được thiết kế.
5. *Thiết kế cấu trúc dữ liệu*: Cấu trúc dữ liệu được sử dụng trong hệ thống được thiết kế chi tiết và cụ thể.
6. *Thiết kế thuật toán*: Những thuật toán được sử dụng để cung cấp các dịch vụ được thiết kế chi tiết và cụ thể.

Đây là một mô hình phổ biến của quá trình thiết kế và sự thực, những quá trình quan trọng có thể chuyển thể để thích nghi theo những cách khác nhau. Một vài sự thích ứng có thể là:

1. Hai khâu cuối của quá trình thiết kế, thiết kế cấu trúc dữ liệu và thiết kế thuật toán, có thể bị trì hoãn tới khi hệ thống được thi hành.
2. Nếu một sự tiếp cận thiết kế để thăm dò được thực hiện, giao diện của hệ thống có thể được thiết kế sau khi cụ thể hoá cấu trúc dữ liệu.
3. Khâu đặc tả trừu tượng có thể được bỏ qua, mặc dù đây thường là khâu quan trọng nhất của việc thiết kế hệ thống.

cứu.

## **Chương 8. KỸ THUẬT LẬP TRÌNH**

### **Mục tiêu**

Lập trình hay cài đặt là một công đoạn trong việc phát triển phần mềm và nó được xem là một hệ quả tất yếu của thiết kế. Tuy vậy, phong cách lập trình và các đặc trưng của ngôn ngữ lập trình có ảnh hưởng lớn đến chất lượng của phần mềm. Một chương trình được cài đặt tốt đem lại cho ta thuận lợi trong việc bảo trì sau này.

Chương này sẽ cung cấp kiến thức về khái quát các ngôn ngữ lập trình, cấu trúc chương trình liệt kê một số công cụ lập trình và ngôn ngữ lập trình.

### 8.1. Khái quát về ngôn ngữ lập trình

Theo một góc nhìn nào đó có thể nói rằng: lịch sử lý thuyết ngôn ngữ lập trình có trước cả sự phát triển của chính bản thân các ngôn ngữ lập trình đó. Phép tính lambda, do Alonzo Church và Stephen Cole Kleene phát triển vào giai đoạn 1930 - 1939, được một số người coi là ngôn ngữ lập trình đầu tiên trên thế giới, mặc dù nó từng được dự định dùng làm mô hình tính toán hơn là phương tiện để cho các lập trình viên mô tả các giải thuật áp dụng cho một hệ thống máy tính. Nhiều ngôn ngữ lập trình hàm được mô tả như là sự bổ sung một “lớp gỗ dán mỏng” lên trên phép tính lambda, và có thể mô tả nhiều ngôn ngữ trong số chúng bằng các thuật ngữ của phép tính lambda một cách dễ dàng.

Ngôn ngữ lập trình đầu tiên được đề trình là Plankalkul, do Konrad Zuse thiết kế vào giai đoạn 1940-1949, nhưng nó không được công chúng biết đến mãi cho đến năm 1972 (nó không được thực hiện cho đến năm 1998). Ngôn ngữ lập trình đầu tiên được biết đến rộng rãi và thành công là FORTRAN, được phát triển từ năm 1954 đến năm 1957 bởi một nhóm nhà nghiên cứu IBM do John Backus phụ trách. Sự thành công của FORTRAN dẫn đến sự hình thành ủy ban các nhà khoa học nhằm phát triển một ngôn ngữ máy tính mang “tính toàn cầu”; và kết quả cho những cố gắng của họ đó là ALGOL 58. Một cách độc lập, John McCarthy của MIT đã phát triển ngôn ngữ lập trình LISP (dựa trên phép tính lambda), ngôn ngữ đầu tiên thành công với khởi điểm từ giới học viện. Bằng sự thành công từ những cố gắng ban đầu này, các ngôn ngữ lập trình máy tính đã trở thành một chủ đề nghiên cứu sôi nổi trong những năm 196x và về sau.

Một số sự kiện chính trong lịch sử lý thuyết ngôn ngữ lập trình kể từ lúc đó:

- Giai đoạn 1950-1959, Noam Chomsky đã phát triển hệ thống phân cấp Chomsky trong lĩnh vực ngôn ngữ học. Đây là khám phá tác động trực tiếp lên lý thuyết ngôn ngữ lập trình và nhiều nhánh khác của khoa học máy tính.
- Giai đoạn 1960-1969, ngôn ngữ Simula đã được Ole Johan Dahl và Kristen Nygaard phát triển. Simula được coi là hình mẫu đầu tiên của một ngôn ngữ lập trình hướng đối tượng; Simula cũng đã giới thiệu khái niệm đồng chương trình con (tiếng Anh: coroutine).
- Giai đoạn 1970-1979:
  - o Một nhóm các nhà khoa học tại Xerox PARC do Alan Kay dẫn dắt đã phát triển Smalltalk, một ngôn ngữ hướng đối tượng nổi tiếng nhờ môi trường phát triển sáng tạo của nó.
  - o Sussman và Steele đã phát triển ngôn ngữ lập trình Scheme, một phiên bản của Lisp hợp nhất phạm vi từ vựng (tiếng Anh: lexical scoping) với một không gian tên thống nhất và các yếu tố từ mô hình Actor (bao gồm các continuation lớp nhất).

Lập trình logic và Prolog phát triển cho phép các chương trình máy tính được biểu hiện như logic toán học.

- Backus, trong một bài giảng tại ACM Turing Award năm 1977, đã đã kích hiện trạng của các ngôn ngữ công nghiệp thời bấy giờ. Backus đã đề xuất một lớp mới các ngôn ngữ lập trình mà hiện nay được biết đến là các ngôn ngữ lập trình mức hàm.
- Xuất hiện phép tính tiến trình, ví dụ như Phép tính của các Hệ thống Giao tiếp (Calculus of Communicating Systems) của Robin Milner, và mô hình Các tiến trình giao tiếp liên tục (Communicating sequential processes) của C. A. R. Hoare, cũng như các mô hình song song tương tự, ví dụ như mô hình Actor của Carl Hewitt.
- Lí thuyết kiểu bắt đầu được áp dụng như là một ngành học (tiếng Anh: discipline) đối với các ngôn ngữ lập trình, được dẫn dắt bởi Milner; ứng dụng này dẫn đến những tiến bộ to lớn trong lí thuyết kiểu suốt nhiều năm qua.

- Giai đoạn 1980-1989:

Bertrand Meyer đã tạo ra phương pháp học Thiết kế theo hợp đồng (Design by contract) và hợp nhất nó vào trong ngôn ngữ lập trình Eiffel.

- Giai đoạn 1990-1999:

Gregor Kiczales, Jim Des Rivieres và Daniel G. Bobrow đã xuất bản cuốn sách Nghệ thuật của Giao thức Đối tượng meta (tựa tiếng Anh: The Art of the Metaobject Protocol).

Philip Wadler đề xuất dùng các monad cho việc cấu trúc các chương trình viết bằng các ngôn ngữ lập trình hàm...

### 8.1.1 Đặc trưng của ngôn ngữ lập trình

Cách nhìn công nghệ phần mềm về các đặc trưng của ngôn ngữ lập trình tập trung vào nhu cầu xác định dự án phát triển phần mềm riêng. Dù vậy ta vẫn có thể thiết lập một tập hợp tổng quát những đặc trưng công nghệ:

- Dễ dịch thiết kế sang chương trình
- Có trình biên dịch hiệu quả
- Khả năng chuyển chương trình gốc
- Có sẵn công cụ phát triển
- Dễ bảo trì.

Bước lập trình bắt đầu sau khi thiết kế chi tiết đã được xác định, xét duyệt và sửa nếu cần. Về lý thuyết, việc sinh chương trình gốc từ một đặc tả chi tiết nên là trực tiếp. Dễ dịch thiết kế sang chương trình đưa ra một chỉ dẫn về việc ngôn ngữ lập trình phản xạ gần gũi đến mức nào cho một biểu diễn thiết kế. Một ngôn ngữ cài đặt trực tiếp cho các kết cấu có cấu trúc, các cấu trúc dữ liệu phức tạp, vào/ra đặc biệt, khả năng thao tác bit, và các kết cấu hướng sự vật sẽ làm cho việc dịch từ thiết kế sang chương trình gốc dễ hơn nhiều (nếu các thuộc tính này được xác định trong thiết kế).

Mặc dầu những tiến bộ nhanh chóng trong tốc độ xử lý và mật độ nhớ đã bắt đầu làm giảm nhẹ nhu cầu chương trình siêu hiệu quả, nhiều ứng dụng vẫn còn đòi hỏi các chương trình chạy nhanh, gọn (yêu cầu bộ nhớ thấp). Các ngôn ngữ với trình biên dịch tối ưu có thể là hấp dẫn nếu hiệu năng phần mềm là yêu cầu chủ chốt. Tính khả chuyển chương trình gốc là một đặc trưng ngôn ngữ lập trình được hiểu theo ba cách khác nhau:

- Chương trình gốc có thể được chuyển từ bộ nhớ xử lý này sang bộ nhớ xử lý khác và từ trình biên dịch nọ sang trình biên dịch kia với rất ít hoặc không phải sửa đổi gì.
- Chương trình gốc vẫn không thay đổi ngay cả khi môi trường của nó thay đổi (như việc cài đặt bản mới của hệ điều hành).
- Chương trình gốc có thể được tích hợp vào trong các bộ trình phần mềm khác nhau với rất ít hay không cần thay đổi gì vì các đặc trưng của ngôn ngữ lập trình.

Trong số ba cách hiểu về tính khả chuyển này thì cách thứ nhất là thông dụng nhất. Việc đưa ra các chuẩn (do tổ chức tiêu chuẩn quốc gia Mỹ – ANSI) góp phần làm nâng cao tính khả chuyển. Tính sẵn có của công cụ phát triển có thể làm ngắn bớt thời gian cần để sinh ra chương trình gốc và có thể cải thiện chất lượng của chương trình. Nhiều ngôn ngữ lập trình có thể cần tới một loạt công cụ kể cả trình biên dịch gỡ lỗi, trợ giúp định dạng chương trình gốc, thư viện chương trình con mở rộng trong nhiều lĩnh vực ứng dụng, các trình duyệt, trình biên dịch chéo cho phát triển bộ vi xử lý, khả năng bộ xử lý macro, công cụ công nghệ ngược và những công cụ khác. Trong thực tế, khái niệm về môi trường phát triển phần mềm tốt (bao hàm cả các công cụ) đã được thừa nhận như nhân tố đóng góp chính cho công nghệ phần mềm thành công.

Tính dễ bảo trì của chương trình gốc có tầm quan trọng chủ chốt cho tất cả các nỗ lực phát triển phần mềm không tầm thường. Việc bảo trì không thể được tiến hành chừng nào người ta còn chưa hiểu được phần mềm. Các yếu tố của cấu hình phần mềm (như tài liệu thiết kế) đưa ra một nền tảng cho việc hiểu biết, nhưng cuối cùng thì chương trình gốc vẫn phải được đọc và sửa đổi theo những thay đổi trong thiết kế.

Tính dễ dịch thiết kế sang chương trình là một yếu tố quan trọng để bảo trì chương trình gốc. Bên cạnh đó, các đặc trưng tự làm tài liệu của ngôn ngữ (như chiều dài được phép của tên gọi, định dạng nhãn, định nghĩa kiểu, cấu trúc dữ liệu) có ảnh hưởng mạnh đến tính dễ bảo trì.

Các đặc trưng của ngôn ngữ lập trình sẽ quyết định miền ứng dụng của ngôn ngữ. Miền ứng dụng là yếu tố chính để chúng ta lựa chọn ngôn ngữ cho một dự án phần mềm. C thường là một ngôn ngữ hay được chọn cho việc phát triển phần mềm hệ thống.

Trong các ứng dụng thời gian thực chúng ta hay gặp các ngôn ngữ như Ada, C, C++ và cả hợp ngữ cho tính hiệu quả của chúng. Các ngôn ngữ này và Java cũng được dùng cho phát triển phần mềm nhúng.

Trong lĩnh vực khoa học kỹ thuật thì FORTRAN với khả năng tính toán với độ chính xác cao và thư viện toán học phong phú vẫn còn là ngôn ngữ thống trị. Tuy vậy, PASCAL và C cũng được ứng dụng rộng rãi.

COBOL là ngôn ngữ cho ứng dụng kinh doanh và khai thác CSDL lớn nhưng các ngôn ngữ thế hệ thứ tư đã dần dần chiếm ưu thế.

BASIC vẫn đang tiến hóa (VISUAL BASIC) và được đông đảo người dùng máy tính cá nhân ủng hộ mặc dù ngôn ngữ này rất hiếm khi được những người phát triển hệ thống dùng.

Các ứng dụng trí tuệ nhân tạo thường dùng các ngôn ngữ như LISP, PROLOG hay OPS5, tuy vậy nhiều ngôn ngữ lập trình (vạn năng) khác cũng được dùng/

Xu hướng phát triển phần mềm hướng đối tượng xuyên suốt phần lớn các miền ứng dụng đã mở ra nhiều ngôn ngữ mới và các dị bản ngôn ngữ quy ước. Các ngôn ngữ lập trình hướng đối tượng được dùng rộng rãi nhất là Smalltalk, C++, Java. Ngoài ra còn có Eiffel, Object Pascal, Flavors và nhiều ngôn ngữ khác.

Với đặc trưng hướng đối tượng, tính hiệu quả thực hiện cũng như có nhiều công cụ và thư viện, C++ hiện đang được sử dụng rộng rãi trong lĩnh vực phát triển các ứng dụng nghiệp vụ. Java cũng là một ngôn ngữ hướng đối tượng đang được sử dụng rộng rãi cho phát triển dịch vụ Web và phần mềm nhúng vì các lý do độ an toàn cao, tính trong sáng, tính khả chuyển và hướng thành phần. Theo một số thống kê thì tốc độ phát triển một ứng dụng mới bằng Java cao hơn đến 2 lần so với các ngôn ngữ truyền thống như C hay thậm chí C++. Các ngôn ngữ biên dịch (script) với những câu lệnh và thư viện mạnh hiện đang rất được chú ý. ASP, JavaScript, PERL... đang được sử dụng rộng rãi trong lập trình Web.

## **8.2. Ngôn ngữ lập trình và sự ảnh hưởng đến công nghệ phần mềm**

Nói chung, chất lượng của thiết kế phần mềm được thiết kế theo cách độc lập với các đặc trưng ngôn ngữ lập trình. Tuy nhiên thuộc tính ngôn ngữ đóng một vai trò trong chất lượng của thiết kế được cài đặt và ảnh hưởng tới cách thiết kế được xác định. Ví dụ như khả năng xây dựng mô đun và bao gói chương trình. Thiết kế dữ liệu cũng có thể bị ảnh hưởng bởi các đặc trưng ngôn ngữ. Các ngôn ngữ lập trình như Ada, C++, Smalltalk đều hỗ trợ cho khái niệm về kiểu dữ liệu trừu tượng – một công cụ quan trọng trong thiết kế và đặc tả dữ liệu. Các ngôn ngữ thông dụng khác như Pascal, cho phép định nghĩa các kiểu dữ liệu cho người dùng xác định và việc cài đặt trực tiếp danh sách móc nối và những cấu trúc dữ liệu khác. Các tính năng này cung cấp cho người thiết kế phạm vi rộng hơn trong các bước thiết kế sơ bộ và chi tiết.

Một số ngôn ngữ lập trình bậc cao hiện nay như Java hay .Net

Các đặc trưng của ngôn ngữ cũng ảnh hưởng đến kiểm thử phần mềm. các ngôn ngữ trực tiếp hỗ trợ cho các kết cấu có cấu trúc có khuynh hướng giảm bớt độ phức tạp của chương trình, do đó có thể làm cho nó dễ dàng kiểm thử. Các ngôn ngữ hỗ trợ cho việc đặc tả các

chương trình con và thủ tục ngoài (như FORTRAN( thường làm cho việc kiểm thử tích hợp ít sinh lỗi hơn

### 8.3. Cấu trúc chương trình

Phong cách lập trình bao hàm một triết lý về lập trình nhấn mạnh tới tính dễ hiểu của chương trình nguồn. Các yếu tố của phong cách lập trình bao gồm: tài liệu bên trong chương trình, phương pháp khai báo dữ liệu, cách xây dựng câu lệnh và các kỹ thuật vào/ra.

#### 8.3.1 Tài liệu chương trình

Tài liệu bên trong của chương trình gốc bắt đầu với việc lựa chọn các tên gọi định danh (biến và nhãn), tiếp tục với vị trí và thành phần của việc chú thích, và kết luận với cách tổ chức trực quan của chương trình. Việc lựa chọn các tên gọi định danh có nghĩa là điều chủ chốt cho việc hiểu chương trình. Những ngôn ngữ giới hạn độ dài tên biến hay nhãn làm các tên mang nghĩa mơ hồ. Cho dù một chương trình nhỏ thì một tên gọi có ý nghĩa cũng làm tăng tính dễ hiểu. Theo ngôn từ của mô hình cú pháp/ngữ nghĩa tên có ý nghĩa làm “đơn giản hóa việc chuyển đổi từ cú pháp chương trình sang cấu trúc ngữ nghĩa bên trong”.

Một điều rõ ràng là: phần mềm phải chứa tài liệu bên trong. Lời chú thích cung cấp cho người phát triển một ý nghĩa truyền thông với các độc giả khác về chương trình gốc. Lời chú thích có thể cung cấp một hướng dẫn rõ rệt để hiểu pha cuối cùng của công nghệ phần mềm – bảo trì. Có nhiều hướng dân đã được đề nghị cho việc viết lời chú thích. Các chú thích mở đầu và chú thích chức năng là hai phạm trù đòi hỏi cách tiếp cận có hơi khác. Lời chú thích mở đầu xuất hiện ngay ở đầu của mọi module. Định dạng cho lời chú thích như thế là:

1. *Một phát biểu về mục đích chỉ rõ chức năng module.*
2. *Mô tả giao diện bao gồm:*
  - *Một mẫu cách gọi*
  - *Mô tả về dữ liệu*
  - *Danh sách tất cả các module thuộc cấp*
3. *Thảo luận về dữ liệu thích hợp (như các biến quan trọng và những hạn chế, giới hạn về cách dùng chúng) và các thông tin quan trọng khác.*
4. *Lịch sử phát triển bao gồm:*
  - *Tên người thiết kế module (tác giả)*
  - *Tên người xét duyệt và ngày tháng*
  - *Ngày tháng sửa đổi và mô tả sửa đổi*

Các chú thích chứa chức năng được nhúng vào bên trong thân của chương trình gốc và được dùng để mô tả cho các khối chương trình.

Khai báo dữ liệu

Thứ tự khai báo dữ liệu nên được chuẩn hóa cho dù ngôn ngữ lập trình không có yêu cầu bắt buộc về điều đó. Các tên biến ngoài việc có nghĩa mang thông tin về kiểu của chúng. Ví dụ, nên thống nhất các tên biến cho kiểu số nguyên, kiểu số thực...

Cần phải chú giải về mục đích đối với các biến quan trọng, đặc biệt là các biến tổng thể. Các cấu trúc dữ liệu nên được chú giải đầy đủ về cấu trúc và chức năng, và các đặc thù về sử dụng. Đặc biệt là đối với các cấu trúc phức tạp như danh sách móc nối trong C hay Pascal.

#### Xây dựng câu lệnh

Việc xây dựng luồng logic phần mềm được thiết lập trong khi thiết kế. Việc xây dựng từng câu lệnh tuy nhiên lại là một phần của bước lập trình. Việc xây dựng câu lệnh nên tuân theo một quy tắc quan trọng hơn cả: mỗi câu lệnh nên đơn giản và trực tiếp. Nhiều ngôn ngữ lập trình cho phép nhiều câu lệnh trên một dòng. Khía cạnh tiết kiệm không gian của tính năng này khó mà biện minh bởi tính khó đọc nản sinh. Cấu trúc chu trình và các phép toán điều kiện được chứa trong đoạn trên đều bị che lấp bởi cách xây dựng nhiều câu lệnh trên một dòng.

Cách xây dựng câu lệnh đơn và việc tụt lề minh họa cho các đặc trưng logic và chức năng của đoạn này. Các câu lệnh chương trình gốc riêng lẻ có thể được đơn giản hóa bởi:

- Tránh dùng các phép kiểm tra điều kiện phức tạp
- Khử bỏ các phép kiểm tra điều kiện phủ định
- Tránh lồng giữa nhiều các điều kiện hay chu trình
- Dùng dấu ngoặc để làm sáng tỏ các biểu thức logic hay số học
- Dùng dấu cách và/hoặc các ký hiệu dễ đọc để làm sáng tỏ nội dung câu lệnh
- Chỉ dùng các tính năng chuẩn của ngôn ngữ

Để hướng tới chương trình dễ hiểu luôn nên đặt ra câu hỏi: Liệu có thể hiểu được điều này nếu ta không là người lập trình cho nó không?

#### Nhập xuất

Vào ra của các mô đun nên tuân thủ theo một số hướng dẫn sau:

- Làm hợp lệ mọi cái vào.
- Kiểm tra sự tin cậy của các tổ hợp khoản mục vào quan trọng.
- Giữ cho định dạng cái vào đơn giản.
- Dùng các chỉ báo cuối hay dữ liệu thay vì yêu cầu người dùng xác định “số các khoản mục”.
- Giữ cho định dạng cái vào thống nhất khi một ngôn ngữ lập trình có các yêu cầu định dạng nghiêm ngặt.

### 8.3.2. Lập trình tránh lỗi

Tránh lỗi và phát triển phần mềm vô lỗi dựa trên các yếu tố sau:

- Sản phẩm của một đặc tả hệ thống chính xác.

- Chấp nhận một cách tiếp cận thiết kế hệ thống phần mềm dựa trên việc bao gói dữ liệu và che giấu thông tin.
- Tăng cường duyệt lại trong quá trình phát triển và thẩm định hệ thống phần mềm.
- Chấp nhận triết lý chất lượng tổ chức: chất lượng là bánh lái của quá trình phần mềm.
- Việc lập kế hoạch cẩn thận cho việc thử nghiệm hệ thống để tìm ra các lỗi chưa được phát hiện trong quá trình duyệt lại và để định lượng độ tin cậy của hệ thống.

Có hai cách tiếp cận chính hỗ trợ tránh lỗi là:

- Lập trình có cấu trúc: Thuật ngữ này được sinh ra từ cuối những năm 60 và có nghĩa là lập trình mà không dùng đến lệnh goto, lập trình chỉ dùng các vòng lặp why và các phát biểu if để xây dựng điều khiển và trong thiết kế thì dùng cách tiếp cận trên – xuống.

Việc thừa nhận lập trình có cấu trúc là quan trọng bởi vì nó là bước đầu tiên bước từ cách tiếp cận không khuôn phép tới phát triển phần mềm. Lập trình có cấu trúc buộc người lập trình phải nghĩ cẩn thận về chương trình của họ, và vì vậy nó ít tạo ra sai lầm trong khi phát triển. Lập trình có cấu trúc làm cho chương trình có thể được đọc một cách tuần tự và do đó dễ hiểu và dễ kiểm tra. Tuy nhiên nó chỉ là bước đầu tiên trong việc lập trình nhằm đạt độ tin cậy tốt. Có một vài khái niệm khác cũng hay dẫn tới các lỗi phần mềm:

\* Các số thực dấu chấm động: các phép toán số thực được làm tròn khiến cho việc so sánh kết quả phép tính không theo mong muốn. Ví dụ, trong phép tính tích phân chúng ta cần cộng các giá trị nhỏ trước với nhau nếu không chúng sẽ bị làm tròn.

\* Các con trỏ và bộ nhớ động: con trỏ là các cấu trúc bậc thấp khó quản lý và dễ gây ra các lỗi nghiêm trọng đối với hệ thống. Việc cấp phát và thu hồi bộ nhớ động phức tạp và là một trong các nguyên nhân chính gây lỗi phần mềm.

\* Song song: lập trình song song đòi hỏi kỹ thuật cao và hiểu biết sâu sắc về hệ thống. Một trong các vấn đề phức tạp của song song là quản lý tương tranh.

\* Độ quy

\* Cách ngắt.

Các cấu trúc này có ích, nhưng người lập trình nên dùng chúng một cách cẩn thận.

- Phân quyền truy cập dữ liệu: Nguyên lý an ninh trong quân đội là các cá nhân chỉ được biết các thông tin có liên quan trực tiếp đến nhiệm vụ của họ. Khi lập trình người ta cũng tuân theo một nguyên lý tương tự cho việc truy cập dữ liệu hệ thống. Mỗi thành phần chương trình chỉ được phép truy cập đến dữ liệu nào cần thiết để thực hiện chức năng của nó. Ưu điểm của việc che giấu thông tin là các thông tin bị che giấu không thể bị sập đổ (thao tác trái phép) bởi các thành phần chương trình mà được xem là không dùng thông tin đó. Tiến hóa của sự phân quyền truy cập là che giấu thông tin hay nói chính xác hơn là che giấu cấu trúc thông tin. Khi đó chúng ta có thể thay đổi cấu trúc thông tin mà không phải thay đổi các thành phần khác có sử dụng thông tin đó.

### 8.3.3 Lập trình tránh lỗi



Đối với các hệ thống đòi hỏi độ tin cậy rất cao như hệ thống điều khiển bay thì cần phải có khả năng dung thứ lỗi (fault tolerance), tức là khả năng đảm bảo cho hệ thống vẫn hoạt động chính xác ngay cả khi có thành phần sinh lỗi.

Có bốn hoạt động cần phải tiến hành nếu hệ thống là thứ lỗi:

- Phát hiện ra lỗi
- Định ra mức độ thiệt hại
- Hồi phục sau khi gặp lỗi: Hệ thống phải hồi phục về trạng thái mà nó biết là an toàn.

Cũng có thể chỉnh lý trạng thái hủy hoại (hồi phục tiến), cũng có thể là lui về một trạng thái trước mà an toàn (hồi phục lùi).

- Chữa lỗi: Cải tiến hệ thống để cho lỗi đó không xuất hiện nữa.

Tuy nhiên trong nhiều trường hợp phát hiện được đúng nguyên nhân gây lỗi là rất khó khăn vì nó xảy ra bởi một tổ hợp của thông tin vào và trạng thái của hệ thống. Thông thường, thứ lỗi được thực hiện bằng cách song song hóa các chức năng, kết hợp với bộ điều khiển thứ lỗi. Bộ điều khiển sẽ so sánh kết quả của các khối chương trình thực hiện cùng nhiệm vụ và sử dụng nguyên tắc đa số để chọn kết quả.

### **8.3.4 Lập trình phòng thủ**

Lập trình phòng thủ là cách phát triển chương trình mà người lập trình giả định rằng các mâu thuẫn hoặc các lỗi chưa được phát hiện có thể tồn tại trong chương trình. Phải có phần mềm kiểm tra trạng thái hệ thống sau khi biến đổi và phải đảm bảo rằng sự biến đổi trạng thái là kiên định. Nếu phát hiện một mâu thuẫn thì việc biến đổi trạng thái là phải rút lại và trạng thái phải trở về trạng thái đúng đắn trước đó.

Nói chung một lỗi gây ra một sự sụp đổ trạng thái: các biến trạng thái được gán các trị không hợp luật. Ngôn ngữ lập trình như Ada cho phép phát hiện các lỗi đó ngay trong thời gian biên dịch. Tuy nhiên việc kiểm tra biên dịch chỉ hạn chế cho các giá trị tĩnh và trong một vài phép kiểm tra thời gian thực là không thể tránh được. Một cách để phát hiện lỗi trong chương trình Ada là dùng cơ chế xử lý bất thường kết hợp với đặc tả miền trị.

Hồi phục là một quá trình cải biên không gian trạng thái của hệ thống sao cho hiệu ứng của lỗi là nhỏ nhất và hệ thống có thể tiếp tục vận hành, có lẽ là trong một mức suy giảm. Hồi phục tiến liên quan đến việc cố gắng chỉnh lại trạng thái hệ thống.

Hồi phục lùi liên quan đến việc lưu trạng thái của hệ thống ở một trạng thái đúng đã biết.

Hồi phục tiến thường là một chuyên biệt ứng dụng. Có hai tình thế chung mà khi đó phục hồi tiến có thể thành công:

- Khi dữ liệu bị sụp đổ: Việc sử dụng kỹ thuật mã hóa thích hợp bằng cách thêm các dữ liệu dư thừa vào dữ liệu cho phép sửa sai khi phát hiện lỗi.
- Khi cấu trúc nối bị sụp đổ: Nếu các con trỏ tiến và lùi đã có trong cấu trúc dữ liệu thì cấu trúc đó có thể tái tạo nếu như còn đủ các con trỏ chưa bị sụp. Kỹ thuật này thường được dùng cho việc sửa chữa hệ thống tệp và cơ sở dữ liệu.

Hồi phục lùi là một kỹ thuật đơn giản liên quan đến việc duy trì các chi tiết của trạng thái an toàn và cất giữ trạng thái đó khi mà sai lầm đã bị phát hiện. Hầu hết các quan hệ quản trị cơ sở dữ liệu (CSDL) đều có bộ hồi phục lỗi. CSDL chỉ cập nhật dữ liệu khi một giao dịch đã hoàn tất và không phát hiện được vấn đề gì. Nếu giao dịch thất bại thì CSDL không được cập nhật.

Một kỹ thuật khác là thiết lập các điểm kiểm tra thường kỳ mà chúng là các bản sao của trạng thái hệ thống. Khi mà một lỗi được phát hiện thì trạng thái an toàn đó được tái lưu kho từ điểm kiểm tra gần nhất. Trường hợp hệ thống dính líu tới nhiều quá trình hợp tác thì đây các giao tiếp có thể là các điểm kiểm tra của các quá trình đó không đồng bộ và để hồi phục thì mỗi quá trình phải trở lại trạng thái ban đầu của nó.

## **8.4. Lập trình hướng hiệu quả thực hiện**

### **8.4.1 Tính hiệu quả chương trình**

Tính hiệu quả của chương trình gốc có liên hệ trực tiếp với tính hiệu quả của thuật toán được xác định trong thiết kế chi tiết. Tuy nhiên, phong cách lập trình có thể có một tác động đến tốc độ thực hiện và yêu cầu bộ nhớ. Tập hợp các hướng dẫn sau đây bao giờ cũng có thể áp dụng được khi thiết kế chi tiết được dịch thành chương trình:

- Đơn giản hóa các biểu thức số học và logic trước khi đi vào lập trình.
- Tính cẩn thận từng chu kỳ lồng nhau để xác định liệu các câu lệnh hay biểu thức có thể được chuyển ra ngoài hay không
  - Khi có thể hãy tránh việc dùng con trỏ và danh sách phức tạp
  - Dùng các phép toán số học “nhanh”
  - Không trộn lẫn các kiểu dữ liệu, cho dù ngôn ngữ có cho phép điều đó
  - Dùng các biểu thức số học và logic bất kì khi nào có thể được

Nhiều trình biên dịch có tính năng tối ưu tự động sinh ra chương trình hiệu quả bằng cách dồn nén các biểu thức lặp, thực hiện tính chu trình, dùng số học nhanh và áp dụng các thuật toán có hiệu quả liên quan khác. Với những ứng dụng trong đó tính hiệu quả có ý nghĩa quan trọng, những trình biên dịch như thế là công cụ lập trình không thể thiếu được.

### **8.4.2. Hiệu quả bộ nhớ**

Tính hiệu quả bộ nhớ phải được tính vào đặc trưng phân trang của hệ điều hành. Nói chung, tính cục bộ của chương trình hay việc bảo trì lĩnh vực chức năng qua các kết cấu có cấu trúc là một phương pháp tuyệt vời làm giảm việc phân trang và do đó làm tăng tính hiệu quả. Hạn chế bộ nhớ trong phát triển phần mềm nhúng là mối quan tâm rất thực tế, mặc dù bộ nhớ giá thấp, mật độ cao vẫn đang tiến hóa nhanh chóng. Nếu yêu cầu hệ thống cần tới bộ nhớ tối thiểu (như sản phẩm giá thấp, khối lượng lớn) thì trình biên dịch ngôn ngữ cấp cao phải được trù tính cẩn thận với những tính năng nén bộ nhớ, hay như một phương kế cuối cùng, có thể phải dùng tới hợp ngữ.

### **8.4.3. Hiệu quả nhập/xuất**

Các thiết bị vào ra thường có tốc độ chậm hơn rất nhiều so với khả năng tính toán của máy tính và tốc độ truy cập bộ nhớ trong. Việc tối ưu vào ra có thể làm tăng đáng kể tốc độ thực hiện. Một số hướng dẫn đơn giản để tăng cường hiệu quả vào/ra:

- Số các yêu cầu vào/ra nên giữ mức độ tối thiểu
- Mọi sự việc vào/ra nên qua bộ nhớ đệm để làm giảm phí tổn liên lạc
- Với bộ nhớ phụ (như đĩa) nên lựa chọn và dùng phương pháp thâm nhập đơn giản nhất chấp nhận được.
- Nên xếp khối vào/ra với các thiết bị bộ nhớ phụ.
- Việc vào/ra với thiết bị cuối hay máy in nên nhận diện các tính năng của thiết bị có thể cải tiến chất lượng và tốc độ.

### **8.5. Cấu trúc chương trình**

Cấu trúc chương trình gốc bắt đầu với việc chọn lựa các tên gọi định danh (biến và nhãn), tiếp tục với vị trí và thành phần của việc chú thích, và kết luận với cách tổ chức trực quan của chương trình. Việc lựa chọn các tên gọi định danh có nghĩa là điều chủ chốt cho việc hiểu chương trình. Những ngôn ngữ giới hạn độ dài tên biến hay nhãn làm các tên mang nghĩa mơ hồ. Cho dù một chương trình nhỏ thì một tên gọi có nghĩa cũng làm tăng tính dễ hiểu. Theo ngôn từ của mô hình cú pháp/ngữ nghĩa tên có ý nghĩa làm “đơn giản hóa việc chuyển đổi từ cú pháp chương trình sang cấu trúc ngữ nghĩa bên trong”.

Một điều rõ ràng là phần mềm phải chứa tài liệu bên trong. Lời chú thích cung cấp cho người phát triển một ý nghĩa truyền thông với các độc giả khác về chương trình gốc. Lời chú thích có thể cung cấp một hướng dẫn rõ rệt để hiểu trong pha cuối cùng của kỹ nghệ phần mềm - bảo trì. Có nhiều hướng dẫn đã được đề nghị cho việc viết lời chú thích. Các chú thích mở đầu và chú thích chức năng là hai phạm trù đòi hỏi cách tiếp cận có hơi khác. Lời chú thích mở đầu nên xuất hiện ở ngay đầu của mọi modul. Định dạng cho lời chú thích như thế là:

1. Một phát biểu về mục đích chỉ rõ chức năng mô đun.
2. Mô tả giao diện bao gồm:
  - Một mẫu cách gọi
  - Mô tả về dữ liệu
  - Danh sách tất cả các mô đun thuộc cấp.
3. Thảo luận về dữ liệu thích hợp (như các biến quan trọng và những hạn chế, giới hạn về cách dùng chúng) và các thông tin quan trọng khác.
4. Lịch sử phát triển bao gồm:
  - Tên người thiết kế modul (tác giả).
  - Tên người xét duyệt và ngày tháng.
  - Ngày tháng sửa đổi và mô tả sửa đổi.

Các chú thích chức năng được nhúng vào bên trong thân của chương trình gốc và được dùng để mô tả cho các khối chương trình.

Bước lập trình là một tiến trình dịch (chuyển hóa) thiết kế chi tiết thành chương trình mà cuối cùng được biến đổi thành các lệnh mã máy thực hiện được. Các đặc trưng của ngôn ngữ lập trình có ảnh hưởng lớn đến quá trình xây dựng, kiểm thử cũng như bảo trì phần mềm. Phong cách lập trình quyết định tính dễ hiểu của chương trình gốc. Các yếu tố của phong cách bao gồm việc làm tài liệu bên trong, phương pháp khai báo dữ liệu, thủ tục xây dựng câu lệnh, và kỹ thuật lập trình vào/ra. Lập trình cần hướng tới hiệu quả thực hiện, tức là tích kiệm tài nguyên phần cứng (mức độ sử dụng CPU, bộ nhớ...). Mặc dầu tính hiệu quả có thể là yêu cầu cực kỳ quan trọng, chúng ta nên nhớ rằng một chương trình hoạt động hiệu quả mà lại không dễ hiểu dẫn đến khó bảo trì thì giá trị của nó cũng bị hạn chế.

### **Câu hỏi**

1. Thế nào là một phong cách cài đặt chương trình tốt.
2. Tài liệu trong của chương trình đem lại những lợi ích gì?
3. Các nền tảng của một ngôn ngữ có thể lập trình được?
4. Lựa chọn ngôn ngữ và nền tảng ứng dụng cho phần mềm các anh chị đang nghiên cứu.
5. Lập bảng luật định và các phương pháp lập trình sử dụng cho nhóm phát triển của các anh chị.

## Chương 9. KIỂM THỬ PHẦN MỀM

### Mục tiêu

Như đã nói ở trước, sản phẩm phần mềm được gọi là đúng nếu nó thực hiện được chính xác những tiêu chuẩn mà người thiết kế đã đặt ra. Để có một đánh giá chính xác về cấp độ đúng của phần mềm, ta phải kiểm tra chất lượng phần mềm. Như thế, kiểm tra là quá trình tìm lỗi và nó là một đánh giá cuối cùng về các đặc tả, thiết kế và mã hoá. Mục đích của kiểm tra là đảm bảo rằng tất cả các thành phần của ứng dụng ăn khớp, vận hành như mong đợi và phù hợp các tiêu chuẩn thiết kế.

Trong chương này, chúng ta thảo luận các chiến lược kiểm tra phần mềm và các kỹ thuật, phương pháp kiểm thử, thiết kế các trường hợp kiểm thử và chiến thuật kiểm thử.

Cuối cùng, các công cụ hỗ trợ kiểm tra tự động và các công cụ hỗ trợ kiểm tra độc lập được trình bày để hỗ trợ cho quá trình kiểm tra.

### 9.1. Khái niệm kiểm thử

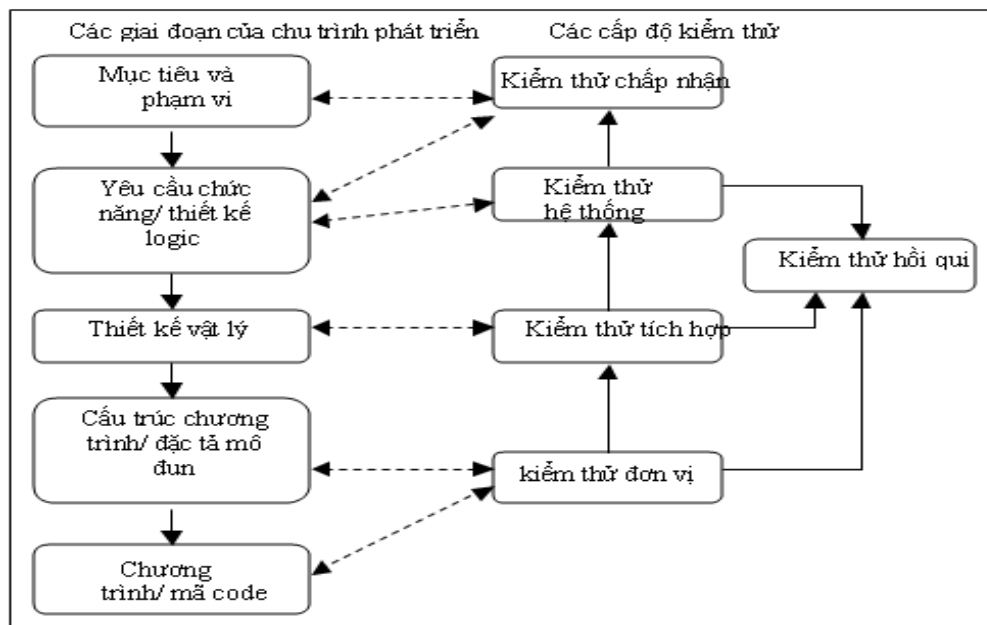
Kiểm thử phần mềm là quá trình phân tích một hạng mục phần mềm để phát hiện sự khác nhau giữa các điều kiện tồn tại và điều kiện yêu cầu để đánh giá các đặc trưng của các hạng mục phần mềm (IEEE, 1986; IEEE, 1990). Kiểm thử phần mềm là một hoạt động được thực hiện xuyên suốt toàn bộ quá trình phát triển phần mềm.

Kiểm thử phần mềm là một trong số các hoạt động kiểm tra và thẩm định phần mềm. Kiểm tra là quá trình đánh giá hệ thống hay thành phần để xác định kết quả của một pha phát triển có thỏa mãn điều kiện đưa ra tại bước khởi đầu pha. Hoạt động kiểm tra bao gồm kiểm thử và kiểm duyệt. Thẩm định là quá trình đánh giá hệ thống hay thành phần trong suốt hay ở giai đoạn cuối của quá trình phát triển để xác định việc thỏa mãn các yêu cầu xác định. Trong bước cuối của sự phát triển hoạt động thẩm định được sử dụng để đánh giá các đặc trưng đã được xây dựng có thỏa mãn các yêu cầu khách hàng và có thể mô tả rõ được các yêu cầu của khách hàng.

Boehm đã xác định hoạt động kiểm tra và thẩm định như sau:

- Kiểm tra: thông qua việc kiểm tra, chúng ta đảm bảo sản phẩm hoạt động đúng như cách mà chúng ta mong đợi.
- Thẩm định: thông qua việc thẩm định, chúng ta kiểm tra để đảm bảo rằng ở đâu đó trong quá trình, việc thẩm định một lỗi luôn liên quan tới việc đối chiếu lại với yêu cầu.

## 9.2. Phương pháp kiểm thử



**Hình 9.1. Các mức kiểm thử tương ứng với các giai đoạn phát triển phần mềm.**

Cấu trúc của một quy trình kiểm thử trong các tổ chức phần mềm có sự khác nhau một cách đáng kể. Các hệ thống khác nhau yêu cầu việc kiểm thử cũng khác nhau, và rất nhiều công ty có các ký hiệu, các định nghĩa khác nhau cho từng giai đoạn kiểm thử mà họ sử dụng.

### **Kiểm thử đơn vị**

Kiểm thử đơn vị là kiểm thử các chức năng của những đơn vị phần mềm cơ sở. Kiểm thử đơn vị được thực hiện bởi người lập trình, mục đích là tìm ra các lỗi trong từng đơn vị phần mềm riêng lẻ, như việc kiểm tra các lớp, các chức năng bị cô lập.

Kiểm thử đơn vị là mức thấp nhất trong tiến trình kiểm thử, thường là áp dụng phương pháp kiểm thử hộp trắng. Người kiểm thử thiết kế đơn vị kiểm thử từ cấu trúc mã lệnh với các yêu cầu chức năng được trình bày trong bảng đặc tả yêu cầu. Một khi việc kiểm thử được thực hiện trên các mô đun mã lệnh nhỏ hơn, điều này dễ dàng để cô lập các lỗi bởi vì thông thường nguồn gốc của chúng từ kiểm thử unit.

Kết quả của kiểm thử Unit thường tìm ra khoảng 20% lỗi trong tất cả cá lỗi của dự án.

### **Kiểm thử tích hợp**

Khi hai hay nhiều đơn vị phần mềm đã được kiểm thử được kết hợp vào một cấu trúc lớn hơn thì việc kiểm thử tích hợp sẽ tìm ra các khiếm khuyết trong các giao diện giữa các đơn vị và trong các hàm mà chưa được kiểm thử.

Việc kiểm thử tích hợp là một tiến trình lặp cho đến khi tất cả các đơn vị được hợp nhất thành một hệ thống và được kiểm thử như một sản phẩm đầy đủ ở giai đoạn tiếp theo.

Quy trình của việc tích hợp các thành phần được thực hiện theo một trong hai phương pháp: tích hợp theo phương pháp top-down (từ trên xuống), nghĩa là các thành phần đơn vị

được thêm vào phần kiểm soát trên cùng của các thành phần hoặc phương pháp bottom-up (từ dưới lên), nghĩa là người thực hiện kiểm thử thêm các thành phần từ mức thấp nhất của hệ thống phân cấp. Kiểm thử tích hợp thường bao gồm kiểm thử giao diện giao tiếp giữa các thành phần. Mục tiêu chính của kiểm thử tích hợp là kiểm thử tất cả các giao tiếp bên trong của các thành phần hay các đơn vị.

### ***Kiểm thử hệ thống***

Sau khi kiểm thử tích hợp đã hoàn tất, kiểm thử hệ thống được thực hiện để kiểm tra tổng thể hệ thống. Trong giai đoạn này sẽ tìm ra khiếm khuyết trong tất cả các yêu cầu chức năng và yêu cầu phi chức năng. Do đó, chức năng kiểm tra thường xuyên là hoạt động chính trong giai đoạn này. Toàn bộ miền yêu cầu phải được xem xét để đáp ứng các tiêu chuẩn cho hệ thống kiểm tra. Để đảm bảo tính chính xác trong các kết quả kiểm tra cần thực hiện các thử nghiệm hệ thống trong một môi trường như môi trường đích của hệ thống khi cài đặt sử dụng. Kiểm thử hệ thống đôi khi bao gồm cả việc kiểm tra cả quy trình như kiểm thử tích hợp hệ thống và kiểm thử cài đặt hệ thống. Kiểm thử tích hợp hệ thống là quá trình kiểm tra khi các sản phẩm cần phải hợp tác, tương tác với các hệ thống phần mềm khác. Trong giai đoạn này, đặc biệt là các yêu cầu chức năng cần phải giao tiếp với các hệ thống khác, yêu cầu phi chức năng cần được kiểm tra. Cài đặt thử nghiệm cũng có thể được gọi là kiểm thử cấu hình nhằm kiểm tra xem phần mềm và phần cứng đã được cài đặt đúng.

Mục đích của kiểm thử hệ thống là để đảm bảo toàn bộ hệ thống hoạt động như ý mà khách hàng mong muốn.

### ***Kiểm thử chấp nhận***

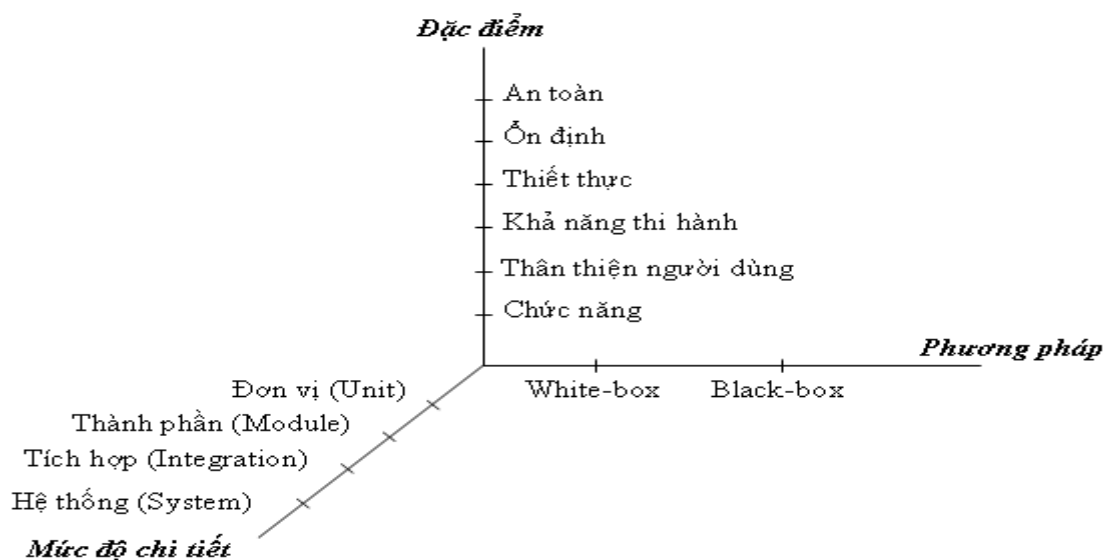
Khi kiểm thử hệ thống được hoàn tất, hệ thống được đưa vào vận hành, khách hàng cần phải chấp nhận hệ thống. Để đạt được sự chấp nhận từ phía khách hàng, bộ phận kiểm thử cần thiết lập một buổi kiểm thử chấp nhận cùng với khách hàng. Mục đích của kiểm thử chấp nhận là để xác nhận một cách chắc chắn rằng hệ thống đang làm việc tốt, đáp ứng yêu cầu của khách hàng và kiểm tra tìm ra những khiếm khuyết lần cuối cùng. Kiểm thử chấp nhận chủ yếu được thực hiện trong hợp đồng phát triển để xác minh rằng hệ thống đáp ứng được các yêu cầu đã thỏa thuận trước đây. Kiểm thử chấp nhận đôi khi được đưa vào trong giai đoạn kiểm thử hệ thống.

### ***Kiểm thử hồi quy***

Quá trình kiểm thử hồi quy được thực hiện sau khi một mô-đun được sửa đổi hoặc một mô-đun mới được thêm vào hệ thống. Vì vậy, kiểm thử hồi quy không phải là một pha độc lập mà được thực hiện lặp lại trong các giai đoạn kiểm thử khác. Mục đích của kiểm thử hồi quy là để kiểm tra các chương trình bị thay đổi với các trường hợp kiểm thử để thiết lập lại sự đúng đắn, tin cậy mà chương trình sẽ phải thực hiện theo đúng đặt tả đã đưa ra. Kiểm thử hồi quy thường được áp dụng ở mức cao của thử nghiệm, và nhằm đạt được hiệu quả, hiệu suất của việc kiểm thử. Kiểm thử hồi quy phụ thuộc rất nhiều vào việc sử dụng lại các trường

hợp kiểm thử đã tạo ra trước đó và cả các kịch bản kiểm thử (test script). Do tính chất thực hiện kiểm thử lặp đi lặp lại nhiều lần nên hoạt động kiểm thử hồi quy được coi là hoạt động tốn kém nhất trong chu trình phát triển phần mềm. Theo Harrold, một số nghiên cứu gần đây cho rằng hoạt động kiểm thử hồi quy tốn gần 1/3 chi phí toàn hệ thống phần mềm.

Hình biểu diễn sự tương quan của “các tiêu chí chất lượng phần mềm”, “mức độ chi tiết đơn vị” và “phương pháp kiểm nghiệm”.



**Hình 9.2. Sự tương quan của tiêu chí chất lượng, các mức và phương pháp**

### 9.3. Thiết kế các trường hợp kiểm thử

Thiết kế kiểm thử cho phần mềm và các sản phẩm kỹ nghệ khác có thể có tính thách thức như việc thiết kế ban đầu cho chính bản thân sản phẩm. Chúng ta cần thiết kế các trường hợp kiểm thử có khả năng cao nhất để tìm ra nhiều lỗi với thời gian và chi phí tối thiểu.

Trong suốt thập kỷ qua, có rất nhiều phương pháp thiết kế trường hợp kiểm thử phần mềm đã tiến hoá. Những phương pháp này cung cấp cho người phát triển một cách tiếp cận hệ thống tới việc kiểm thử. Quan trọng hơn, những phương pháp này đưa ra một cơ chế có thể giúp đảm bảo tính đầy đủ của kiểm thử và đưa ra khả năng cao nhất để phát hiện lỗi trong phần mềm.

Bất cứ sản phẩm kỹ nghệ nào đều có thể kiểm thử theo 2 cách:

- Khi biết chức năng xác định mà một sản phẩm đã được thiết kế để thực hiện thì có thể tiến hành kiểm thử xem từng chức năng có vận hành hoàn toàn không
- Biết cách làm việc bên trong của một sản phẩm, tiến hành kiểm thử để đảm bảo rằng tất cả “các bánh răng ăn khớp vào nhau” tức là đảm bảo rằng sự vận hành bên trong thực hiện tương ứng với đặc tả và tất cả các thành phần bên trong đã được thử một cách thích hợp.

Cách tiếp cận thứ nhất được gọi là kiểm thử hộp đen, cách tiếp cận thứ hai gọi là kiểm thử hộp trắng.



Khi phần mềm máy tính được xem xét thì kiểm thử hộp đen ám chỉ việc kiểm thử được tiến hành tại giao diện phần mềm. Mặc dầu chúng được thiết kế để phát hiện ra lỗi, kiểm thử hộp đen được dùng để thể hiện rằng các chức năng phần mềm là vận hành; rằng cái vào được chấp nhận là đúng, cái ra được tạo ra đúng; rằng tính toàn vẹn của thông tin ngoài (như các tệp dữ liệu) là được duy trì. Phép kiểm thử hộp đen xem xét một số khía cạnh của hệ thống mà ít để ý đến cấu trúc logic bên trong của phần mềm.

Việc thiết kế kiểm thử hộp trắng cho phần mềm được xác định trên xem xét kỹ về chi tiết thủ tục. Các đường logic đi qua phần mềm được kiểm thử bằng cách đưa ra các trường hợp kiểm thử, vốn thực hiện trên một tập các điều kiện và/hoặc chu trình. Trạng thái của chương trình có thể được xem xét tại nhiều điểm khác nhau để xác định liệu trạng thái được trông đợi hay khẳng định có tương ứng với trạng thái hiện tại hay không.

Giá trị đầu vào	Kết quả (Số lần lặp)
11	0 (bỏ qua vòng lặp)
10	1 (chạy 1 lần lặp)
9	2 (chạy 2 lần lặp)
5	6 (trường hợp chạy m lần lặp khi $m < n$ )
2	9 (chạy $N-1$ lần lặp)
1	10 (chạy $N$ lần lặp)
0	0 (bỏ qua vòng lặp)

### Câu hỏi

1. Độ tin cậy của phần mềm? Vì sao phải đảm bảo chất lượng phần mềm.
2. Đặc điểm của việc kiểm tra phần mềm? Sự khác nhau của kiểm tra hộp trắng và hộp đen.
3. Thế nào là kiểm thử phần mềm? Các đặc điểm của việc kiểm thử phần mềm.
4. Tìm hiểu các công cụ hỗ trợ kiểm thử phần mềm và ứng dụng vào phần mềm các anh chị đang nghiên cứu.
5. Có bao nhiêu chiến thuật kiểm thử ? Đánh giá hiệu quả của các chiến thuật kiểm thử phần mềm.

## Chương 10. BẢO TRÌ PHẦN MỀM

### Mục tiêu

Bảo trì là giai đoạn cuối cùng của một chu trình phát triển phần mềm. Các chương trình máy tính luôn thay đổi- phải mở rộng, sửa lỗi, tối ưu hoá,...và theo thống kê thì bảo trì chiếm đến 70% toàn bộ công sức bỏ ra cho một dự án phần mềm. Do vậy, bảo trì là một hoạt động phức tạp nhưng nó lại là vô cùng cần thiết trong chu trình sống của sản phẩm phần mềm để đảm bảo cho phần mềm phù hợp với người sử dụng.

Do nhu cầu phát triển của các hệ thống thông tin, rất hiếm hay không muốn nói là không thể có một hệ thống thông tin không có sự thay đổi trong suốt chu trình sống của nó. Để duy trì tính đúng đắn, trật tự trong giai đoạn bảo trì thì quản lý sự thay đổi phần mềm là một hoạt động cần thiết song song.

Chương 10 sẽ cung cấp kiến thức về khái niệm bảo trì phần mềm, qui trình nghiệp vụ bảo trì phần mềm, các vấn đề về bảo trì phần mềm hiện nay

### 10.1. Khái niệm bảo trì phần mềm

Không thể tạo ra một hệ thống nào mà hoàn toàn không cần thay đổi. Trong suốt thời gian tồn tại của một hệ thống, các yêu cầu ban đầu của nó sẽ thay đổi do những thay đổi trong nhu cầu của khách hàng và người sử dụng. Môi trường hệ thống cũng thay đổi khi một phần cứng mới được giới thiệu. Lỗi, không được phát hiện trong quá trình kiểm thử, cũng có thể nguy hiểm và cần thiết phải sửa chữa.

Quá trình thay đổi hệ thống sau khi chuyển giao cho người sử dụng và vẫn đang được sử dụng gọi là bảo trì phần mềm. Sự thay đổi có thể bao gồm một số thay đổi như sửa các lỗi lập trình, sửa các lỗi thiết kế hoặc các tăng cường đáng kể để sửa chữa các lỗi đặc tả hoặc thích ứng với các yêu cầu mới. Do đó bảo trì trong ngữ cảnh này gần như có nghĩa là tiến hoá.

Người ta chia ra thành các loại bảo trì sau mặc dù sự khác biệt của chúng là không rõ ràng:

- Bảo trì để tu sửa
- Bảo trì để thích hợp
- Bảo trì để cải tiến
- Bảo trì để phòng ngừa.

#### 10.1.1 Bảo trì để tu sửa

Hoạt động bảo trì này để khắc phục những khiếm khuyết trong phần mềm. Hoạt động này xuất hiện vì giả thiết rằng việc kiểm thử phần mềm sẽ phát hiện ra tất cả các lỗi tiềm tàng trong một hệ thống phần mềm nhất là các hệ thống phần mềm lớn là vô lý. Trong khi dùng bất cứ chương trình lớn nào, lỗi sẽ xuất hiện và được báo về cho người phát triển.

Bảo trì này xuất hiện do một số nguyên nhân chủ yếu sau:

- Kỹ sư phần mềm và khách hàng hiểu nhầm nhau
- Lỗi tiềm ẩn của phần mềm do sơ ý của lập trình hoặc khi kiểm thử chưa bao quát hết
- Vấn đề tính năng của phần mềm: không đáp ứng được yêu cầu về bộ nhớ, tệp..., thiết kế sai, biên tập sai.

- Thiếu chuẩn hoá trong phát triển phần mềm

Khi áp dụng hoạt động bảo trì này cần lưu ý đến:

- Kỹ nghệ ngược: dò lại thiết kế để tu sửa
- Mức trừu tượng
- Tính tương tác
- Tính đầy đủ

### **10.1.2 Bảo trì để thích hợp**

Hoạt động bảo trì thích hợp tu chỉnh phần mềm theo những thay đổi của môi trường bên ngoài nhằm duy trì và quản lý phần mềm theo vòng đời của nó. Các thể hệ phần cứng mới, các hệ điều hành mới hay các phiên bản mới của hệ điều hành cũ xuất hiện đều đặt, thiết bị ngoại vi và các phần tử hệ thống khác thường xuyên được nâng cấp và thay đổi sẽ yêu cầu phần mềm phải thay đổi để khớp với môi trường thay đổi.

### **10.1.3 Bảo trì để cải tiến**

Khi phần mềm được đưa vào sử dụng, người ta sẽ nhận được từ người sử dụng những khuyến cáo về khả năng mới, những sửa đổi về chức năng hiện tại, và những nâng cao chung. Để thoả mãn những yêu cầu trong phạm vi này, người ta tiến hành bảo trì để cải tiến.

Các bước thực hiện bảo trì để cải tiến:

- Xây dựng lưu đồ phần mềm
- Suy dẫn ra biểu thức Boolean cho từng dãy xử lý
- Biên dịch bảng chân lý
- Tái cấu trúc phần mềm

### **10.1.4 Bảo trì để phòng ngừa**

Đây là hoạt động bảo trì tu chỉnh chương trình có tính đến tương lai phần mềm đó sẽ mở rộng và thay đổi như thế nào. Thực ra trong khi thiết kế phần mềm đã tính đến tính mở rộng của nó nên thực tế ít khi gặp bảo trì phòng ngừa nếu như phần mềm được thiết kế tốt.

Bảo trì này được thực hiện trên các thiết kế không tương minh. Sử dụng các công cụ CASE cho kỹ nghệ ngược và tái kỹ nghệ sẽ tự động hoá một phần công việc.

## **10.2 Các đặc trưng của bảo trì**

Để hiểu các đặc trưng của bảo trì phần mềm, ta xem xét dựa trên 3 quan điểm sau:

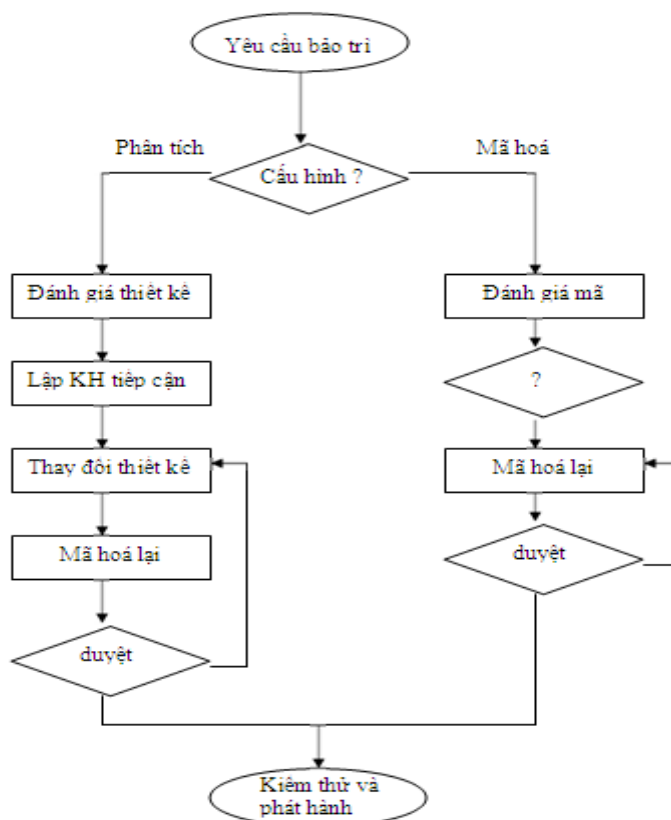
- Các hoạt động đòi hỏi hoàn thành giai đoạn bảo trì và tác động của cách tiếp cận kỹ nghệ phần mềm (hay thiếu cách tiếp cận) lên tính hiệu quả của các hoạt động như vậy.
- Chi phí liên kết với giai đoạn bảo trì
- Các vấn đề thường gặp khi tiến hành bảo trì phần mềm

### 10.2.1. Bảo trì có cấu trúc so với bảo trì phi cấu trúc

Luồng các sự kiện có thể xuất hiện như kết quả của yêu cầu bảo trì được minh hoạ như trong hình 8.1. Nếu phần tử sẵn có của cấu hình phần mềm là chương trình gốc, thì hoạt động bảo trì bắt đầu với một đánh giá cẩn thận về mã, thường bị phức tạp bởi tài liệu nội bộ nghèo nàn. Những đặc trưng tinh vi như cấu trúc chương trình, cấu trúc dữ liệu toàn cục, giao diện hệ thống, và các ràng buộc hiệu năng và/hoặc thiết kế thì khó chắc.

chẩn được và thường bị hiểu sai. Việc phân nhánh những thay đổi thường được tiến hành sau cùng cho chương trình thì thật khó thẩm định. Các phép kiểm thử hồi quy (lặp lại phép thử quá khứ để đảm bảo rằng những sửa đổi không đưa ra lỗi vào phần mềm vận hành trước đây) không thể nào tiến hành được vì không có bản ghi lại việc kiểm thử.

Chúng ta đang tiến hành việc bảo trì phi cấu trúc và phải trả giá trong việc lãng phí công sức và gây chán nản cho con người) thường đi kèm với những phần mềm chưa được phát triển có dùng phương pháp luận xác định rõ.



**Hình 10.1: Bảo trì có cấu trúc so với bảo trì phi cấu trúc**

Nếu tồn tại một cấu hình phần mềm đầy đủ, bảo trì bắt đầu với việc đánh giá tài liệu thiết kế. Cấu trúc quan trọng, hiệu năng và các đặc trưng giao diện của phần mềm được xác định. Tác động của các sửa đổi hay sửa chữa được yêu cầu sẽ được thẩm định và một cách tiếp cận được lên kế hoạch. Bản thiết kế sẽ được sửa đổi và xét duyệt lại. Chương trình gốc mới được xây dựng, các phép thử hồi quy được tiến hành bằng cách dùng các thông tin có trong bản Đặc tả kiểm thử, và phần mềm lại được xuất xưởng như một kết quả của việc áp

dụng trước đây các phương pháp luận về kỹ nghệ phần mềm. Mặc dầu sự tồn tại của cấu hình phần mềm không đảm bảo việc bảo trì không có vấn đề, khối lượng công sức cũng đã được rút gọn đáng kể và chất lượng tổng thể của thay đổi hay sửa đổi được nâng cao.

### 10.2.2 Chi phí bảo trì

Chi phí cho việc phần mềm đã tăng dần trong 20 năm qua. Chi phí về tiền là mối quan tâm của chúng ta. Tuy nhiên những chi phí ít thấy khác cũng lại có thể là nguyên nhân cho các mối quan tâm lớn hơn. Một chi phí vô hình của việc bảo trì phần mềm là cơ hội phát triển bị trì hoãn hay bị mất tài nguyên sẵn có phải chuyển cho các nhiệm vụ bảo trì. Các chi phí vô hình khác bao gồm:

Sự không thoả mãn của khách hàng khi các yêu cầu sửa chữa hay thay đổi có vẻ hợp lý lại không được đề cập đến một cách kịp thời. Sự suy giảm chất lượng phần mềm tổng thể xem như kết quả của những thay đổi tạo thêm lỗi trong phần mềm đã được bảo trì. Biến động đột ngột xảy ra trong nỗ lực phát triển khi nhân viên bị “kéo” sang làm việc bảo trì.

Chi phí cuối cùng cho việc bảo trì phần mềm giảm rất nhiều theo hiệu suất (được đo theo LOC trên người – tháng hay điểm chức năng trên người – tháng), điều thường gặp phải khi bảo trì chương trình cũ được khởi đầu.

Nỗ lực dành cho việc bảo trì có thể bị phân chia vào các hoạt động sản xuất (như phân tích và đánh giá, sửa đổi thiết kế, mã hoá) và các hoạt động như hiểu mã chương trình làm gì, thử diễn giải cấu trúc dữ liệu, các đặc trưng giao diện, các biên hiệu năng.

Biểu thức sau đây đưa ra mô hình về nỗ lực bảo trì:

$$M = p + Ke(c-d)$$

Với  $M$  = tổng nỗ lực dành cho bảo trì

$K$  = hằng số kinh nghiệm

$c$  = việc đo độ phức tạp có thể coi là đóng góp vào việc thiếu thiết kế và tư liệu tốt

$d$  = việc đo mức độ quen thuộc với phần mềm

Mô hình được mô tả trên chỉ ra rằng nỗ lực (và chi phí) có thể tăng lên theo hàm mũ nếu cách tiếp cận phát triển phần mềm nghèo nàn (tức là thiếu kỹ nghệ phần mềm) được sử dụng và người hay nhóm người dùng cách tiếp cận này còn chưa sẵn có để thực hiện việc bảo trì.

Lưu ý:

Chi phí cho việc thêm vào một chức năng mới của hệ thống sau khi nó được triển khai thường lớn hơn chi phí xây dựng các chức năng tương tự khi phần mềm được phát triển ban đầu bởi vì:

Các nhân viên bảo trì thường không có kinh nghiệm và không quen thuộc với miền ứng dụng. Bảo trì là một hình ảnh nghèo nàn giữa các kỹ sư phần mềm. Đây thường được

xem là công việc đòi hỏi ít kỹ năng hơn phát triển hệ thống mà nó thường được giao cho các nhân viên cấp thấp.

Chương trình được bảo trì đã được phát triển nhiều năm trước đây nên không có các kỹ thuật kỹ nghệ phần mềm hiện đại. Chúng cần được xây dựng và tối ưu hoá để nâng cao tính hiệu quả. Thay đổi chương trình có thể làm phát sinh ra các lỗi mới. Các lỗi này lại yêu cầu các thay đổi.

Khi hệ thống thay đổi cấu trúc của nó dường như bị suy biến. Điều đó làm cho hệ thống khó hiểu hơn và các thay đổi sẽ khó khăn hơn khi tính cố kết của hệ thống bị giảm. Mối liên quan giữa chương trình và các tài liệu liên quan giảm đi. Do đó các tài liệu này không còn đủ tin cậy để trợ giúp việc hiểu chương trình

### **10.2.3 Các vấn đề**

Phần lớn các vấn đề liên quan đến bảo trì phần mềm đều có thể quy về những khiếm khuyết trong cách vạch kế hoạch và xây dựng phần mềm. Trong số các vấn đề cổ điển có liên quan đến việc bảo trì, có các vấn đề sau:

Thường khó hay không thể thay đổi được sự tiến hoá của phần mềm qua nhiều phiên bản hay lần phát hành. Những thay đổi đã không được chứng minh bằng tư liệu một cách đầy đủ. Khó hay không thể nào theo dõi được tiến trình qua đó tạo nên phần mềm. Thường đặc biệt khó để hiểu chương trình là của ai đó. Cái khó tăng lên khi số phần tử trong cấu hình phần mềm giảm đi. Nếu chỉ có mã chương trình không có tư liệu thì chắc chắn sẽ có nhiều vấn đề nghiêm trọng.

Thường không có ai đó bên cạnh để giải thích. Tính cơ động giữa các nhân viên phần mềm khá cao. Chúng ta không thể dựa vào giải thích cá nhân của người phát triển phần mềm khi phải bảo trì Tài liệu không có hoặc có nhưng tồi. Việc thừa nhận rằng phần mềm phải được tư liệu hoá là bước đầu tiên, nhưng việc tư liệu hoá phải làm sao để có thể hiểu được và nhất quán với chương trình gốc bằng bất cứ giá nào.

Phần lớn các phần mềm đều không được thiết kế cho sự thay đổi. Chừng nào mà phương pháp thiết kế còn chưa phù hợp với sự thay đổi thông qua các khái niệm như phụ thuộc hàm hay lớp sự vật thì việc thay đổi đối với phần mềm vẫn còn khó khăn và sinh lỗi.

Việc bảo trì phần mềm còn chưa được coi là công việc rất quyến rũ. Phần lớn sự nhận thức này đến từ mức độ chán nản cao liên quan đến công việc bảo trì phần mềm.

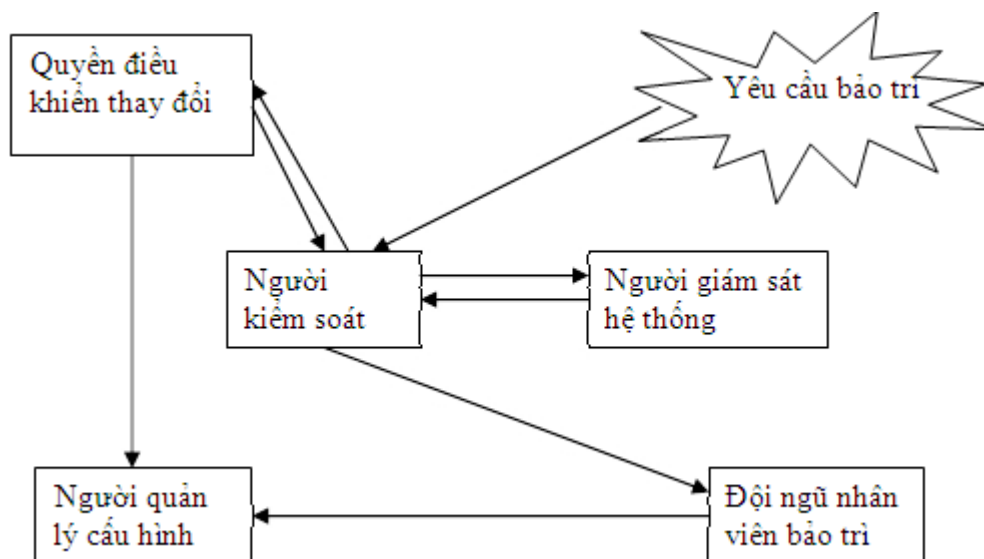
Tất cả những vấn đề được mô tả ở trên, một phần, đều có thể là do một phần số chương trình hiện nay đang tồn tại mà không có sự cân nhắc nào về kỹ nghệ phần mềm. Kỹ nghệ phần mềm cung cấp ít ra là các giải pháp bộ phận cho từng vấn đề có liên quan đến việc bảo trì.

## **10.3 Nhiệm vụ bảo trì**

Nhiệm vụ liên quan đến bảo trì phần mềm bắt đầu từ lâu trước khi một yêu cầu về bảo trì được nêu ra. Ban đầu, phải thành lập tổ chức bảo trì chính thức hay ngầm định, phải mô tả

các thủ tục báo cáo và đánh giá, và cũng phải xác định trình tự đã chuẩn hóa cho các sự kiện đối với từng yêu cầu bảo trì. Bên cạnh đó, cũng phải thiết lập cả thủ tục cất giữ việc ghi lại đối với các hoạt động bảo trì cùng việc xác định ra các tiêu chuẩn xét duyệt và đánh giá.

### 10.3.1 Tổ chức bảo trì



**Hình 10.2: Mô hình tổ chức bảo trì**

Các yêu cầu bảo trì được chuyển qua kênh người kiểm soát bảo trì, người chuyển tiếp từng yêu cầu đánh giá cho người giám sát hệ thống. Người giám sát hệ thống là thành viên của bộ phận kỹ thuật, những người được trao trách nhiệm phải trở nên am hiểu từng tập con nhỏ nhất của chương trình sản phẩm. Một khi việc đánh giá đã được tiến hành thì người có thẩm quyền điều khiển thay đổi (đôi khi còn được gọi là ban điều khiển thay đổi) phải xác định hoạt động cần tiến hành.

Tổ chức trên đã làm giảm bớt những lẫn lộn và cải tiến luồng hoạt động bảo trì. Vì tất cả các yêu cầu bảo trì đều trút về một cá nhân hay một nhóm nhưng sự sắp xếp bí mật (ví dụ như những thay đổi không được thừa nhận và do đó có thể gây ra lẫn lộn) thì khó có thể xảy ra. Vì ít nhất một cá nhân bao giờ cũng có một sự am hiểu nào đó với chương trình sản phẩm, nên yêu cầu thay đổi (bảo trì) có thể được thẩm định nhanh chóng hơn.

Vì việc chấp nhận điều khiển thay đổi đặc biệt được thực hiện nên tổ chức có thể tránh việc gây ra thay đổi có lợi cho một người yêu cầu này mà lại có tác dụng tiêu cực với người khác.

Mỗi một trong các tiêu đề công việc trên đều dùng để thiết lập một lĩnh vực trách nhiệm cho bảo trì. Người kiểm soát và quyền điều khiển thay đổi có thể chỉ là một người hay là một nhóm các nhà quản lý và nhân viên kỹ thuật cấp cao (với hệ thống lớn). Người giám sát hệ thống có thể có nghĩa vụ khác, nhưng cũng đưa ra một “tiếp xúc” với bộ trình phần mềm đặc biệt.

Khi các trách nhiệm được trao trước hết để bắt đầu hoạt động bảo trì thì sự lẫn lộn được giảm đi nhiều. Nhưng điều quan trọng hơn, việc xác định trách nhiệm từ sớm có thể kiềm chế bất kỳ cảm giác khó chịu nào vẫn thường phát triển khi một người bị “kéo ra” khỏi nỗ lực phát triển để tiến hành bảo trì.

### **10.3.2 Báo cáo**

Tất cả các yêu cầu về bảo trì phần mềm được trình bày theo cách thức chuẩn hoá. Người phát triển phần mềm thông thường tạo ra một mẫu yêu cầu bảo trì (MRF), đôi khi còn được gọi là báo cáo vấn đề phần mềm, do người muốn có hoạt động bảo trì điền vào.

Nếu gặp phải lỗi, cần phải đưa vào cả một mô tả đầy đủ về hoàn cảnh dẫn tới lỗi (dữ liệu vào, bản in và các tài liệu hỗ trợ khác). Với các yêu cầu bảo trì thích nghi hay hoàn thiện, một đặc tả thay đổi ngắn gọn (đặc tả yêu cầu tóm tắt) cần được đệ trình.

MRF là tài liệu được sinh ra bên ngoài, vốn được dùng làm cơ sở cho việc lập kế hoạch nhiệm vụ bảo trì. Về nội bộ, tổ chức phần mềm xây dựng ra một báo cáo thay đổi phần mềm (SCR) chỉ ra:

1. Độ lớn của nỗ lực cần cho việc thoả mãn MRF
2. Bản chất của những thay đổi cần có
3. Số ưu tiên của yêu cầu
4. Dữ liệu sau sự kiện về việc thay đổi

SCR được đệ trình để thay đổi quyền kiểm soát trước khi việc lập kế hoạch thêm được bắt đầu.

### **10.3.3 Luồng sự kiện**

Yêu cầu đầu tiên là xác định khiếu bảo trì cần được tiến hành. Trong nhiều trường hợp, người dùng có thể xét yêu cầu như một chỉ dẫn về lỗi phần mềm (bảo trì sửa chữa) trong khi người phát triển có thể coi cũng yêu cầu đó là thích nghi hay nâng cao. Nếu tồn tại những ý kiến khác nhau thì phải thương lượng phương pháp.

Một yêu cầu cho bảo trì sửa lỗi (đường lỗi) bắt đầu với một ước lượng về mức nghiêm trọng của lỗi. Nếu lỗi nghiêm trọng xuất hiện (như hệ thống tới hạn không thể vận hành) thì nhân sự sẽ được phân bổ theo chỉ thị của người giám sát hệ thống và việc phân tích vấn đề được bắt đầu ngay lập tức. Với những lỗi ít nghiêm trọng hơn, yêu cầu về bảo trì sửa chữa được ước lượng và phân loại rồi lập lịch đi kèm với các nhiệm vụ khác cần tới tài nguyên phát triển phần mềm.

Trong một số trường hợp, một lỗi có thể nghiêm trọng đến mức việc kiểm soát thông thường về bảo trì bị tạm thời ngưng lại. Chương trình phải được sửa đổi ngay lập tức, không có sự ước lượng tương ứng về hiệu quả phụ tiềm năng và việc cập nhật thích hợp cho tài liệu. Cách “chữa cháy” này cho bảo trì sửa chữa chỉ được dành riêng cho các tình huống “khủng hoảng” và nên biểu thị cho một số phần trăm rất nhỏ trong tất cả các hoạt động bảo trì. Cũng nên lưu ý rằng việc chữa cháy trì hoãn nhưng không khừ bỏ được nhu cầu kiểm soát và ước



lượng. Sau khi giải quyết xong khủng hoảng thì những hoạt động này phải được tiến hành để đảm bảo rằng những lỗi hiện tại sẽ không lan truyền các vấn đề nghiêm trọng hơn.

Các yêu cầu về bảo trì thích nghi và hoàn thiện đi theo một con đường khác. Việc thích nghi được ước lượng và phân loại (sắp ưu tiên) trước khi được đặt vào hàng đợi bảo trì. Việc nâng cao cũng trải qua cùng ước lượng này. Tuy nhiên không phải tất cả các yêu cầu nâng cao đều được tiến hành. Chiến lược kinh doanh, tài nguyên có sẵn, khuynh hướng sản phẩm phần mềm hiện tại và tương lai, và nhiều vấn đề khác có thể làm cho yêu cầu nâng cao bị bác bỏ. Những sự nâng cao cần được tiến hành cũng được đặt vào hàng đợi. Độ ưu tiên cho mỗi yêu cầu đều được thiết lập và công việc cần thiết sẽ được lên lịch đường như nó là nỗ lực phát triển khác (với mọi ý định và mục đích thì nó quả là như vậy). Nếu một số ưu tiên thật cao được nêu ra thì công việc có thể bắt đầu ngay lập tức.

Bất kể đến kiểu bảo trì, đều phati tiến hành cùng những nhiệm vụ kỹ thuật. Những nhiệm vụ bảo trì này bao gồm việc sửa đổi thiết kế phần mềm, xét duyệt, sửa chương trình, kiểm thử đơn vị và tích hợp (kể cả kiểm thử hồi quy dùng các trường hợp kiểm thử trước đây), kiểm thử hợp lệ và xét duyệt. Trong thực tế, việc bảo trì phần mềm chính là kỹ nghệ phần mềm được áp dụng đệ quy. Sự nhấn mạnh sẽ dịch chuyển theo từng kiểu bảo trì, nhưng cách tiếp cận toàn bộ vẫn không thay đổi. Sự kiện cuối cùng trong luồng bảo trì là việc xét duyệt làm hợp lệ tất cả các phần tử của cấu hình phần mềm và đảm bảo rằng MRF trong thực tế đã được hoàn thành.

Sau khi nhiệm vụ bảo trì được hoàn tất, một ý tốt là tiến hành xét duyệt tình huống.

Nói chung, việc xét duyệt cố gắng trả lời cho các câu hỏi sau:

- Với các tình huống được cho, những khía cạnh nào của thiết kế, mã hoá hay kiểm thử có thể được thực hiện khác đi?

- Tài nguyên bảo trì nào đáng phải có mà lại không có?

- Các hành cảnh khó khăn chính (phụ) cho nỗ lực này là gì?

- Liệu việc bảo trì phòng ngừa được chỉ ra bởi yêu cầu có được báo cáo lại hay không?

Việc xét duyệt tình huống này có thể có ảnh hưởng tới việc tiến hành các nỗ lực bảo trì tương lai và đưa ra phản hồi, điều quan trọng cho việc quản lý của tổ chức phần mềm.

#### **10.3.4 Lưu trữ bản ghi**

Vấn đề đầu tiên gặp phải khi gìn giữ bản ghi bảo trì là phải hiểu dữ liệu nào đáng ghi lại. Swanson nêu ra một danh sách có cân nhắc sau:

1. Xác định chương trình
2. Số các câu lệnh chương trình gốc
3. Số các lệnh mã máy
4. Ngôn ngữ lập trình được dùng
5. Ngày thiết đặt chương trình

6. Số chương trình chạy từ lần thiết đặt đó
7. Số lần hỏng hóc xử lý liên quan đến mục 6
8. Mức độ và việc định danh sự thay đổi chương trình
9. Số câu lệnh gốc được thêm vào bởi việc thay đổi chương trình
10. Số câu lệnh gốc bị xóa
11. Số người - giờ dành cho việc thay đổi
12. Ngày thay đổi chương trình
13. Định danh kỹ sư phần mềm
14. Định danh MRF
15. Kiểu bảo trì
16. Ngày bắt đầu và kết thúc bảo trì
17. Số tích lũy về người -giờ dành cho bảo trì
18. Ích lợi rõ ràng liên kết với việc bảo trì được thực hiện

Các dữ liệu trên được thu thập cho từng nỗ lực bảo trì. Swanson đề nghị những mục này nên lấy làm nền tảng cho cơ sở dữ liệu bảo trì có thể được ước lượng.

Một công trình khác về độ đo bảo trì phần mềm thì tập trung vào các đặc trưng phần mềm gần như có thể ảnh hưởng tới tần số bảo trì và các mô hình kinh nghiệm để tiên đoán khối lượng công việc bảo trì dựa trên các đặc trưng chương trình khác. Bằng cách dùng COCOMO làm cơ sở, mô hình sau đây đã gợi ý như một bộ dự báo về số người – tháng, E.maint, cần chi phí cho việc bảo trì phần mềm hàng năm

$$E.maint = ACT * 2,4 * KLOC^{1.05}$$

Với ACT là lưu lượng thay đổi hàng năm được định nghĩa là:

$$ACT = KLOC \text{ cho hệ thống đang được bảo hành/ CI}$$

Với CI là số lệnh chương trình gốc bị thay đổi hay thêm vào trong một năm bảo trì.

Mặc dầu kết quả của nghiên cứu lâm sàng phải được áp dụng một cách thận trọng, các mô hình định lượng đưa ra một phần tử của việc kiểm soát quản lý hay bị bỏ qua đối với bảo trì phần mềm.

### 10.3.5 Ước lượng

Một ước lượng về các hoạt động bảo trì thường phức tạp do thiếu dữ liệu khó. Nếu việc gìn giữ bản ghi được khởi đầu thì có thể xây dựng được một số cách đo hiệu năng bảo trì. Swanson đưa ra một danh sách tóm tắt các độ đo hiệu năng:

1. Số trung bình những sai hỏng xử lý trên việc chạy chương trình
2. Toàn bộ số người - giờ dành cho từng phạm trù bảo trì
3. Số trung bình những thay đổi chương trình được thực hiện theo chương trình, theo ngôn ngữ, theo kiểu bảo hành

4. Số trung bình người - giờ trên câu lệnh chương trình gốc cần thêm vào hay xóa đi do việc bảo trì

5. Số người -giờ trung bình dành ra cho ngôn ngữ

6. Thời gian quay vòng trung bình cho một MRF

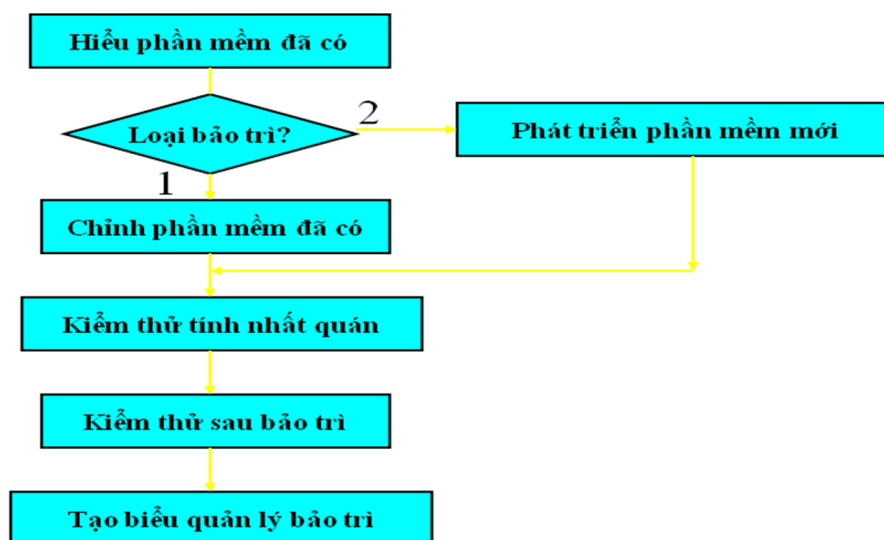
7. Số phần trăm các yêu cầu bảo trì phần mềm

Bằng cách đo trên có thể cung cấp một khuôn khổ định lượng để đưa ra những quyết định về kỹ thuật phát triển, chọn lựa ngôn ngữ, dự phòng nỗ lực bảo trì, cấp phát tài nguyên và nhiều vấn đề khác. Rõ ràng, những dữ liệu như vậy có thể được áp dụng để ước lượng nhiệm vụ bảo trì.

## 10.2. Quy trình nghiệp vụ bảo trì phần mềm

Quy trình bảo trì là quá trình trong vòng đời của phần mềm, cũng tuân theo các pha phân tích, thiết kế, phát triển và kiểm thử từ khi phát sinh vấn đề cho đến khi giải quyết xong.

- Thao tác bảo trì: gồm 2 loại.
  - Tu chỉnh cái đã có (loại 1)
  - Thêm cái mới (loại 2)



**Hình 10.3. Sơ đồ bảo trì**

Hiểu phần mềm đã có

- Theo tài liệu nắm chắc các chức năng.
- Theo tài liệu chi tiết hãy nắm vững đặc tả chi tiết, điều kiện kiểm thử...
- Đọc chương trình nguồn, hiểu trình tự xử lý chi tiết của hệ thống.

Ba việc trên đều là công việc thực thi trên bàn

- Tu sửa phần mềm đã có:
  - Bảo trì chương trình nguồn, tạo các modul mới và dịch lại.
  - Thực hiện kiểm thử unit và tu chỉnh những mục liên quan có trong tư liệu đặc tả.
  - Chú ý theo sát tác động của modul được sửa đến các thành phần khác trong hệ thống.

- Phát triển phần mềm mới:
  - Khi thêm chức năng mới phải phát triển chương trình cho phù hợp với yêu cầu.
  - Cần tiến hành từ thiết kế, lập trình, gỡ lỗi và kiểm thử unit.
  - Phản ánh vào giao diện của phần mềm (thông báo, phiên bản...)
- Kiểm chứng tính nhất quán bằng kiểm thử kết hợp.
  - Đưa đơn vị ( unit) đã được kiểm thử vào hoạt động trong hệ thống.
  - Điều chỉnh sự tương thích giữa các modun.
  - Dùng các dữ liệu trước đây khi kiểm thử để kiểm thử lại tính nhất quán.

### 10.3. Các vấn đề về bảo trì phần mềm hiện nay

Phương pháp cải tiến thao tác bảo trì :

- Sáng kiến trong quy trình phát triển phần mềm.
- Sáng kiến trong quy trình bảo trì phần mềm.
- Phát triển những kỹ thuật mới cho bảo trì.

Sáng kiến trong quy trình phát triển phần mềm.

- (1) Chuẩn hóa mọi khâu trong phát triển phần mềm.
- (2) Người bảo trì chủ chốt tham gia vào giai đoạn phân tích và thiết kế.
- (3) Thiết kế để dễ bảo trì.
- (4) Sử dụng các công cụ hỗ trợ phát triển phần mềm.
- (5) Chuẩn hóa thao tác bảo trì và thiết bị môi trường bảo trì.
- (6) Lưu lại những thông tin sử bảo trì.
- (7) Dự án nên cử một người chủ chốt của mình làm công việc bảo trì sau khi dự án kết thúc giai đoạn phát triển.

### Câu hỏi

1. Chuyển giao phần mềm? Các hoạt động chính của chuyển giao phần mềm?
2. Bảo trì phần mềm là gì?
3. Các hoạt động nào trong bảo trì phần mềm là chính yếu?
4. Các đặc trưng của hoạt động bảo trì?
4. Các khó khăn của hoạt động bảo trì?
5. Một số hình thức bảo trì phần mềm?
6. Nêu quy trình bảo trì phần mềm . Áp dụng đưa ra quy trình bảo trì phần mềm các anh chị đang nghiên cứu.

## Chương 11. CÁC CHỦ ĐỀ KHÁC TRONG CNPM

### Mục tiêu

Ngoài những kiến thức đã cung cấp xuyên suốt 10 chương, chương này sẽ cung cấp một số thông tin bổ sung về các mảng ước lượng chi phí phần mềm, quản lý chất lượng, cải tiến quy trình và các lĩnh vực trong ngành công nghiệp phần mềm.

#### 11.1. Ước lượng chi phí phần mềm

Trong chương 4, chúng ta đã nghiên cứu về lập kế hoạch cho dự án, những công việc của dự án được chia thành các hoạt động riêng biệt. Sự thảo luận sớm về việc lập kế hoạch cho dự án tập trung vào việc miêu tả các hoạt động, sự phụ thuộc và sự phân phối của con người để thực hiện các nhiệm vụ. Trong chương này, tôi muốn nói đến vấn đề kết hợp ước lượng của sự cố gắng và thời gian với các hoạt động của dự án. Sự ước lượng bao gồm trả lời các câu hỏi sau:

1. Bao nhiêu cố gắng được yêu cầu để hoàn thành mỗi hoạt động.
2. Bao nhiêu thời gian cần thiết để hoàn thành mỗi hoạt động.
3. Tổng giá trị của mỗi hoạt động.

Ước lượng chi phí dự án và lập kế hoạch cho dự án thường được tiến hành cùng nhau. Chi phí của sự phát triển là chi phí chính trong các thứ bao gồm, do đó sự tính toán được sử dụng là trong cả ước đoán chi phí và ước đoán lịch trình. Tuy nhiên, bạn có thể phải làm một số sự ước lượng chi phí trước khi lịch trình được đi vào cụ thể. Sự ước lượng khởi đầu có thể sử dụng là ngân sách cho dự án hoặc tập giá trị phần mềm cho khách hàng. Có ba tham số trong sự tính toán tổng chi phí của dự án phát triển phần mềm:

1. Chi phí phần cứng và phần mềm kể cả bảo dưỡng.
2. Chi phí đi lại và đào tạo.
3. Chi phí cho kỹ sư (Chi phí chi trả chi kỹ sư phần mềm).

Hầu hết các dự án, chi phí chủ yếu là chi phí cho kỹ sư. Máy tính mà nó đủ mạnh cho phát triển phần mềm là phải tương đối rẻ. Mặc dù chi phí cho sự đi lại có thể cần thiết khi dự án được phát triển ở các địa điểm khác nhau, chi phí đi lại thường chỉ là một phần nhỏ so với chi phí cho kỹ sư. Hơn nữa, sử dụng các hệ thống giao tiếp điện tử như thư điện tử, các trang web và hội nghị trực tuyến có thể làm giảm đáng kể yêu cầu cho sự đi lại. Hội thảo điện tử cũng có nghĩa là thời gian đi lại giảm và có thể có năng suất cao hơn cho việc phát triển phần mềm. Trong một dự án, nơi tôi đã làm việc, tạo ra tất cả các hội thảo trực tuyến thay vì các hội thảo gặp mặt trực tiếp, đã giảm được chi phí đi lại và thời gian đến 50%. Chi phí cho kỹ sư không hẳn là lương của các kỹ sư phần mềm mà họ có trong dự án. Tổ chức tính toán chi phí cho kỹ sư trong nhóm dựa trên tất cả chi phí mà họ có được để thực hiện dự án và chia nó cho số lượng năng suất nhân viên. Do đó, các chi phí sau đây là tất cả các thành phần của chi phí cho kỹ sư.

- Chi phí cho điều kiện nhiệt độ, ánh sáng của công sở.
- Chi phí hỗ trợ nhân viên như nhân viên kế toán, nhà quản trị, người quản lý hệ thống, người dọn dẹp, và các kỹ thuật viên.
- Chi phí kết nối mạng và giao tiếp
- Chi phí cho những trung tâm tạo điều kiện tốt cho làm việc như thư viện hay nơi giải trí.
- Chi phí cho sự bảo vệ chung và lợi ích cho người làm việc như tiền trợ cấp và bảo hiểm sức khỏe.

Chi phí chính thường ít nhất bằng hai lần lương của kỹ sư phần mềm, phụ thuộc và kích thước tổ chức và sự kết hợp của nó. Vì thế, nếu một công ty trả chi kỹ sư phần mềm 90.000 bảng/Năm thì tổng chi phí ít nhất là 180.000 Bảng/Năm hay 15.000

bảng/Tháng. Khi một dự án đang được thực hiện, người quản lý dự án nên thường xuyên và đều đặn cập nhật ước lượng chi phí và lịch trình. Điều này giúp cho việc lập kế hoạch cho tiến trình và việc sử dụng hiệu quả tài nguyên. Nếu thực sự có sự tiêu dùng vượt đáng kể so với dự tính, thì người quản lý dự án phải có một số hành động. Nó có thể bao gồm việc tăng ngân sách cho dự án hoặc sửa đổi lại công việc cho nó trở nên đúng đắn.

Dự toán phần mềm nên được thực hiện khách quan với mục đích dự đoán chính xác chi phí phát triển phần mềm. Nếu chi phí dự án được tính toán như giá khách hàng phải trả cho dự án, thì việc quyết định định giá là việc đặt giá cho khách hàng. Giá cả đơn giản là chi phí cộng với lợi nhuận. Tuy nhiên, mối quan hệ giữa chi phí dự án và giá chi khách hàng thường không đơn giản.

Việc định giá cho phần mềm phải bao gồm cả tổ chức, kinh tế, chính trị, thương mại, mà nó được thể hiện trong hình 26.1. Bởi vậy, nó có thể không chỉ là sự quan hệ đơn giản giữa giá cả khách hàng phải trả cho dự án và chi phí phát triển dự án. Bởi vì bao gồm nhiều rủi ro trong tổ chức, việc định giá dự án nên có sự góp mặt của những nhà quản lý kinh nghiệm (để có những quyết định chiến lược).

Ví dụ, một công ty phần mềm nhỏ phục vụ đầu thuê 10 kỹ sư hồi đầu năm, nhưng chỉ có những hợp đồng trong lĩnh vực mà nó yêu cầu 5 thành viên trong việc mở rộng nhân viên. Tuy nhiên, đó là sự trả giá cho những hợp đồng rất lớn với công ty đầu lớn mà nó yêu cầu 30 người làm việc cố gắng trong 2 năm. Dự án sẽ không thể bắt đầu sau 12 tháng, và như thế điều đó sẽ thay đổi tài chính của công ty nhỏ. Công ty cung cấp đầu có cơ hội trả giá cho dự án mà nó yêu cầu 6 người và phải hoàn thành trong 10 tháng. Chi phí (bao gồm tất cả trong dự án) được ước tính là 1.2 triệu Đô. Tuy nhiên, để tăng khả năng cạnh tranh, công ty cung cấp đầu định giá cho khách hàng là 0.8 triệu Đô. Điều đó có nghĩa là, mặc dù mất tiền trong hợp đồng này, nhưng nó đã được những chuyên viên hữu ích cho những dự án trong tương lai.

**Bảng 11.1: Nhân tố ảnh hưởng giá phần mềm**

Nhân tố	Diễn tả
Cơ hội trong thị trường	Một tổ chức phát triển có thể đặt một mức giá thấp bởi vì nó mong muốn chuyển tới một giai đoạn mới trong thị trường phần mềm. Chấp nhận một mức lợi thấp trong một dự án có thể cho tổ chức cơ hội có lợi nhuận lớn hơn trong tương lai, và vốn kinh nghiệm thu được có thể sẽ rất hữu ích trong tương lai.
Sự không chắc chắn của ước lượng chi phí	Nếu một tổ chức không dám chắc và sự ước tính chi phí, điều đó có thể làm gia tăng giá thành bởi một số sự cố bất ngờ.
Những yêu cầu dễ thay đổi	Nếu các yêu cầu có khả năng thay đổi, một tổ chức có thể giảm giá thành để có được hợp đồng. Sau đó, lại tăng giá thành để thay đổi các yêu cầu.
Tình trạng tài chính	Nhà phát triển trong sự khó khăn tài chính có thể giảm giá thành để đạt được hợp đồng. Điều đó tốt hơn để tạo ra các lợi nhuận nhỏ hơn là bình thường hoặc sẽ không thể tiếp tục kinh doanh.

## 11.2. Năng suất phần mềm

Bạn có thể tính toán năng suất trong hệ thống bằng cách đếm số lượng các đơn vị được sản xuất và chia chúng cho lượng thời gian yêu cầu làm ra chúng. Tuy nhiên, với bất kỳ vấn đề phần mềm nào, sẽ có nhiều giải pháp khác nhau, mỗi trong số chúng lại có những tính chất khác nhau. Một giải pháp có thể thực hiện một cách hiệu quả trong khi những cái khác lại có thể dễ đọc và dễ bảo trì. Khi những giải pháp với những thuộc tính khác nhau được đưa ra, việc so sánh chúng thường không có nhiều ý nghĩa. Tuy nhiên, là một nhà quản lý dự án, bạn có thể phải đối mặt với vấn đề về ước lượng năng suất của kỹ sư phần mềm. Bạn có thể cần sự ước lượng này để giúp đưa ra chi phí cho dự án và lập lịch trình, để quyết định sự đầu tư hoặc để đánh giá phát triển quá trình và kỹ thuật thêm hiệu quả.

Ước lượng năng suất thường có cơ sở dựa trên việc tính toán các thuộc tính của phần mềm và chia chúng cho tổng sự cố gắng yêu cầu cho việc phát triển. Có hai loại thang đo được sử dụng:

- Thước đo liên quan đến kích thước: Đó là sự liên quan đến kích thước một số đầu ra từ một số hoạt động. Cái chung nhất trong thước đo này là các đường phân chia mã nguồn. Các thước đo khác có thể sử dụng số lượng của phân chia lệnh mã mục tiêu hoặc số lượng của các trang của tài liệu hệ thống.

- Thước đo liên quan đến hàm: Đó là sự liên quan đến tổng thể các chức năng trong sự phân chia phần mềm. Năng suất giới hạn bởi số lượng chức năng hữu dụng được sử dụng trong thời gian hạn định. Điểm hàm và điểm mục tiêu là điều quan trọng nhất trong loại thước đo này.

Các đường mã nguồn / lập trình viên-tháng ( Lines of source code per programmer-month – LOC/pm) được sử dụng rất nhiều trong việc đo năng suất phần mềm. Bạn có thể tính toán LOC/pm bằng cách đếm tổng số đường mã nguồn mà nó được phân chia, sau đó chia nó cho tổng thời gian làm việc của lập trình viên trong một tháng yêu cầu để hoàn thành dự án. Thời gian này bao gồm tất cả các hoạt động yêu cầu (Nhu cầu, thiết kế, viết mã, kiểm tra và dẫn chứng tư liệu) cần thiết trong việc phát triển phần mềm.

Điều đó tiến gần đến sự phát triển đầu tiên khi hầu hết các chương trình được lập trình bằng FORTRAN, hợp ngữ hoặc COBOL. Sau đó, các chương trình như một loại thẻ, với mỗi câu lệnh trên một thẻ. Số lượng của mã dễ dàng đếm được: Nó tương ứng với số lượng của các thẻ trong chương trình. Tuy nhiên, các chương trình trong các ngôn ngữ như JAVA hoặc C++ bao gồm các khai báo, các câu lệnh và các chú thích. Nó có thể cũng bao gồm các macro lệnh mà nó mở rộng thêm vài dòng trong đoạn mã. Nó cũng có thể có nhiều hơn một câu lệnh trên một dòng. Vì thế, quan hệ đơn giản giữa các lệnh chương trình và các dòng ở trong danh sách. So sánh năng suất thông qua ngôn ngữ lập trình có thể cũng mắc các sai lầm do cảm giác về năng suất chương trình. Các ngôn ngữ lập trình càng cao thì càng khó xác định năng suất. Điều không bình thường này nảy sinh bởi vì tất cả các hoạt động phát triển phần mềm đều được tính toán đến trong khi tính toán thời gian phát triển, nhưng thang đo LOC chỉ áp dụng cho quá trình lập trình. Vì thế, nếu một ngôn ngữ yêu cầu nhiều dòng hơn các cái khác để bổ sung cùng một chức năng, việc ước lượng năng suất sẽ trở nên bất thường.

Ví dụ, xem xét một hệ thống ghi thời gian thực nó có thể có mã trong 5000 dòng trong hợp ngữ hoặc 1500 dòng trong C. Thời gian phát triển cho các mặt khác nhau được diễn tả trong bảng 11.2. Người lập trình hợp ngữ có năng suất là 714 dòng/tháng và người lập trình ngôn ngữ bậc cao thì ít hơn một nửa là 300 dòng/tháng. Vào lúc này nói chi phí phát triển cho phát triển hệ thống trong C là thấp hơn thì nó được phát biểu quá sớm.

**Bảng 11.2: Thời gian phát triển hệ thống.**

	Phân tích	Thiết kế	Viết mã	Kiểm tra	Báo cáo
Hợp ngữ	3 tuần	5 tuần	8 tuần	10 tuần	2 tuần
Ngôn ngữ cấp cao	3 tuần	4 tuần	4 tuần	6 tuần	2 tuần



	Kích thước	Thời gian làm	Năng suất
Hợp ngữ	5000 dòng	28 tuần	714 dòng/tháng
Ngôn ngữ cấp cao	1500 dòng	20 tuần	300 dòng/tháng

Cách khác sử dụng kích cỡ mã như việc ước lượng thuộc tính sản phẩm là sử dụng các tính toán chức năng của mã. Điều này tránh các bất thường nêu trên, như các chức năng là độc lập với ngôn ngữ bổ sung. MacDonell (MacDonell, 1994) mô tả ngắn gọn và so sánh vài tính toán về các hàm cơ bản. Hiểu biết tốt nhất về tính toán này là việc đếm điểm-hàm. Nó được trình bày bởi Albrecht (Albrecht, 1979) và được sửa lại bởi Albrecht và Gaffney (Albrecht và Gaffney, 1983). Garmus và Herron (Garmus và Herron, 2000) mô tả thực tiễn sử dụng điểm-hàm trong các dự án phần mềm.

Năng suất thông qua điểm-hàm là sự bổ sung của người làm việc trên tháng. Điểm hàm không chỉ là đặc điểm mà là sự tính toán một vài sự đo đạc là ước lượng. Bạn tính tổng số điểm-hàm trong chương trình bằng cách đo hoặc ước lượng các đặc điểm sau của chương trình:

- Giao tiếp vào và ra
- Sử dụng tương tác
- Giao diện giao tiếp
- Văn bản sử dụng trong hệ thống

Hiển nhiên, một số đầu vào và đầu ra, các tương tác, là những thứ phức tạp và tốn nhiều thời gian hơn để bổ sung. Thước đo điểm-hàm đưa vào báo cáo bằng cách nhân điểm-hàm khởi tạo với hệ số-phức tạp-tác dụng. Bạn nên ước định mỗi trong các đặc điểm cho sự phức tạp và sau đó gán hệ số tác dụng từ 3 (cho giao tiếp đầu vào đơn giản) tới 15 cho các văn bản nội bộ phức tạp. Có thể lựa chọn giá trị tác dụng đề xuất bởi Albrecht hoặc các giá trị cơ sở trong kinh nghiệm để sử dụng.

Sau đó bạn có thể tính toán cái gọi là đếm điểm hàm không thích ứng (unadjusted function-point count-UFC) bằng cách nhân mỗi giá trị khởi tạo với ước lượng tác dụng và cộng tổng tất cả các giá trị.

$$UFC = \Sigma(\text{số lượng của các thành phần có kiểu nhất định}) * (\text{hệ số tác dụng})$$

Bạn có thể thay đổi UFC bằng cách thêm vào hệ số phức tạp mà nó gần với độ phức tạp của toàn bộ hệ thống. Điều này đưa vào báo cáo mức độ phân bố tiến trình, tổng số dùng lại, hiệu suất,... Đếm điểm hàm không thích ứng (UFC) được nhân với hệ số phức tạp dự án để đưa ra điểm hàm cho toàn bộ hệ thống.

Symons (Symons, 1988) chú ý rằng ước lượng độ phức tạp bằng cách đếm điểm-hàm trong chương trình phụ thuộc vào người đưa ra ước lượng. Con người khác nhau sẽ có những ý niệm khác nhau về độ phức tạp. Vì thế sự đa dạng trong việc đếm điểm-hàm phụ thuộc vào quyết định của người ước lượng và kiểu của hệ thống được phát triển. Hơn thế

nữa, điểm hàm thường có xu hướng về hệ thống xử lý dữ liệu mà nó bị chi phối bởi các toán tử vào và ra. Và thật khó để ước lượng điểm hàm cho các hệ thống hướng sự kiện. Vì lý do đó, một số người nghĩ điểm hàm là không thực sự thích hợp để tính toán năng suất phần mềm (Furey và Kitchenham, 1997; Armour, 2002). Tuy nhiên, người sử dụng điểm hàm tranh luận gây gắt rằng đó là một cách hiệu quả trong thực tiễn (Banker,... 1993, Garmus và Herron, 2000)

Điểm mục tiêu (Banker,...,1994) là một khả năng khác của điểm hàm. Chúng có thể được sử dụng với các ngôn ngữ như các ngôn ngữ lập trình cơ sở dữ liệu, ngôn ngữ kịch bản. Điểm mục tiêu không phải là lớp mục tiêu mà nó có thể sản xuất khi tiến gần định hướng mục tiêu được đưa lại trong sự phát triển phần mềm. Số lượng của điểm hàm trong chương trình là việc ước lượng tác dụng của:

1. *Số lượng các màn ảnh riêng biệt được hiển thị* một màn hình đơn được coi như một điểm mục tiêu, các màn ảnh phức tạp vừa phải được đếm là 2, và các màn ảnh rất phức tạp được đếm là 3.
2. *Số lượng các biên bản được đưa ra* cho một biên bản đơn lẻ đếm thêm 2 điểm mục tiêu, cho các biên bản phức tạp vừa phải là 5 và các biên bản rất khó để đưa ra là 8.
3. *Số lượng các modul trong các ngôn ngữ lập trình bắt buộc như Java hoặc C++ mà nó phải được phát triển để bổ xung cho mã lập trình cơ sở dữ liệu* mỗi trong các modul được đếm 10 điểm mục tiêu.

Điểm mục tiêu được sử dụng trong mô hình ước lượng COCOMO II (được gọi là điểm ứng dụng) được nói đến trong các chương sau. Lợi thế của điểm mục tiêu hơn điểm hàm là nó dễ ràng hơn để ước lượng cho các chi tiết kỹ thuật phần mềm cấp cao. Điểm mục tiêu thường chỉ liên quan đến các màn hình, biên bản và các modul trong các ngôn ngữ lập trình. Nó thường không liên quan đến các chi tiết bổ xung, và các ước lượng hệ số phức tạp là đơn giản.

Nếu điểm hàm hoặc điểm mục tiêu được sử dụng, chúng có thể được ước lượng trong giai đoạn sớm của quá trình phát triển trước khi quyết định ảnh hưởng đến kích thước chương trình được tạo ra. Ước lượng các thông số có thể làm cùng giao tiếp tương tác của hệ thống khi được thiết kế. Trong giai đoạn này, là rất khó để đưa ra kích thước của chương trình và số dòng của mã nguồn.

Điểm hàm và điểm mục tiêu có thể được sử dụng kết hợp với mô hình ước lượng số dòng của mã. Kích thước mã cuối cùng được tính toán từ số lượng điểm hàm. Sử dụng phân tích dữ liệu lịch sử, số lượng dòng trung bình của mã, AVC, trong các ngôn ngữ riêng biệt để bổ xung các điểm hàm có thể ước lượng được. Giá trị của AVC thay đổi từ 200 đến 300 LOC/FP trong hợp ngữ, từ 2 đến 40 LOC/FP trong ngôn ngữ lập trình cơ sở dữ liệu như SQL. Ước lượng kích thước mã cho ứng dụng mới được tính toán như sau:

Kích thước mã = AVC\*số lượng điểm hàm.

Năng suất lập trình trong làm việc cá nhân trong tổ chức là phụ thuộc vào nhiều yếu tố. Một số trong các yếu tố quan trọng được đưa ra trong hình 26.3. Tuy nhiên, cá nhân khác nhau trong khả năng thường có nhiều ý nghĩa hơn bất kỳ nhân tố nào. Trong sự ước định sớm của năng suất, Sackman (Sackman, 1968) tìm thấy một số lập trình viên mà họ có năng suất gấp hơn 10 lần so với những người khác. Kinh nghiệm của tôi cho thấy điều này vẫn đúng. Những nhóm lớn thường có khả năng hoà hợp khả năng và kinh nghiệm và sẽ có được năng suất trung bình. Trong những nhóm nhỏ, tuy nhiên, năng suất chung thường phụ thuộc rất nhiều vào các khả năng và năng khiếu cá nhân.

**Bảng 11.3 Các nhân tố ảnh hưởng đến năng suất thiết kế phần mềm.**

Nhân tố	Mô tả
Kinh nghiệm miền ứng dụng	Hiểu biết về miền ứng dụng là cần thiết để phát triển phần mềm hiệu quả. Những kỹ sư đã hiểu về miền thường làm việc có hiệu quả cao.
Chất lượng quá trình	Sử dụng quá trình phát triển có thể có hiệu quả đặc biệt trong năng suất. Điều này được nói đến trong Chương 28
Kích thước dự án	Một dự án lớn sẽ cần nhiều thời gian hơn yêu cầu cho việc giao tiếp trong nhóm. Ít thời gian có thể sử dụng cho việc phát triển dẫn đến suy giảm năng suất cá nhân.
Hỗ trợ kỹ thuật	Hỗ trợ kỹ thuật tốt như công cụ CASE và hệ thống tổ chức quản lý cấu hình có thể tăng năng suất.
Môi trường làm việc	Như đã thảo luận trong Chương 25, một môi trường làm việc yên tĩnh với vùng làm việc riêng góp phần nâng cao năng suất.

Năng suất phát triển phần mềm thay đổi nhanh chóng giữa miền ứng dụng và các tổ chức. Với hệ thống lớn, phức tạp, năng suất được ước lượng dưới 30 LOC/pm. Với những hệ thống ứng dụng dễ hiểu, viết bằng ngôn ngữ như Java, nó có thể cao hơn 900 LOC/pm. Khi tính toán giới hạn của điểm hàm Boehm (Boehm, 1995) đề xuất năng suất thay đổi từ 4 điểm hàm trên tháng đến 50 trên tháng, phụ thuộc vào kiểu ứng dụng, công cụ hỗ trợ và khả năng phát triển.

Vấn đề với phép đo mà nó phụ thuộc vào số lượng sản phẩm trong thời gian hạn định là nó đưa vào báo cáo về đặc điểm chất lượng có thể duy trì và đáng tin cậy. Beck (Beck, 2000) trong thảo luận của ông về lập trình, đã tạo điểm xuất sắc về sự ước lượng. Nếu cách tiếp cận của bạn là cơ sở cho việc đơn giản mã và phát triển, do đó việc đến dòng trong mã không còn nhiều ý nghĩa.

Phép đo cũng không đưa vào báo cáo khả năng sử dụng lại các sản phẩm phần mềm, sử dụng những mã có sẵn và các công cụ khác để tạo ra phần mềm.

Là một nhà quản lý, bạn không nên sử dụng việc tính toán năng suất để có những quyết định vội vàng về khả năng của kỹ sư trong nhóm của bạn. Nếu bạn làm, kỹ sư có thể thỏa hiệp về chất lượng trong kế hoạch trở nên hiệu quả hơn. Nó có thể trở thành trường hợp mà người lập trình kém hiệu quả đưa ra mã đáng tin cậy hơn – mã mà dễ dàng để hiểu và có thể duy trì không tốn kém. Vì thế, bạn nên luôn nghĩ về đo năng suất như cung cấp các thông tin không hoàn chỉnh về năng suất của lập trình viên. Bạn cũng nên quan tâm đến các thông tin khác về chất lượng của chương trình họ đã làm ra.

### **11.3. Kỹ thuật ước lượng**

Không có cách đơn giản nào tạo ra ước lượng một cách chính xác cho hệ thống phát triển phần mềm. Bạn có thể phải tạo ra ước lượng khởi tạo trên nền tảng của định nghĩa yêu cầu người dùng cấp cao. Phần mềm có thể phải thực thi trong các máy tính lạ hoặc sử dụng kỹ thuật phát triển mới. Những người trong dự án và kỹ năng của họ có thể sẽ không được biết đến. Tất cả những ý trên là không thể ước lượng chi phí phát triển hệ thống một cách chính xác trong giai đoạn đầu của dự án.

Hơn thế nữa, đó là cái khó cơ bản trong việc đánh giá đúng đắn của các cách tiếp cận khác nhau đến kỹ thuật ước lượng chi phí. Sự ước lượng được dùng để xác định ngân sách dự án, và sản phẩm sẽ được điều chỉnh hợp lý. Tôi không biết gì về các cuộc thí nghiệm điều khiển với việc định giá dự án nơi mà ước lượng chi phí không được sử dụng trong thí nghiệm. Cuộc thí nghiệm điều khiển sẽ không bộc lộ được ước lượng chi phí cho người quản lý dự án. Chi phí trên thực tế sẽ được so sánh với chi phí ước lượng. Tuy nhiên, như một cuộc thí nghiệm là có thể không khả thi bởi vì bao gồm chi phí cao và số lượng có thể thay đổi là không thể điều khiển.

Tuy nhiên, tổ chức cần phải tạo ra sự cố gắng trong phần mềm và ước lượng chi phí. Để làm việc đó, các kỹ thuật được đưa ra trong hình 8.4 có thể được sử dụng (Boehm, 1981). Tất cả các kỹ thuật đều dựa trên nền tảng kinh nghiệm quyết định bởi những người quản lý dự án mà họ sử dụng hiểu biết của họ để xem xét dự án và ước lượng tài nguyên yêu cầu cho dự án. Tuy nhiên, có thể có những khác biệt quan trọng giữa dự án quá khứ và tương lai. Rất nhiều phương thức và kỹ thuật phát triển mới được giới thiệu sau 10 năm. Một số thí dụ ảnh hưởng đến cơ sở ước lượng trong nhiệm vụ bao gồm:

1. Phân phối các hệ thống mục tiêu hơn là hệ thống chung.
2. Sử dụng dịch vụ web.
3. Sử dụng ERP hoặc các hệ thống trung tâm cơ sở dữ liệu
4. Sử dụng các phần mềm không bảo vệ hơn là phát triển hệ thống nguồn.
5. Phát triển cùng với việc sử dụng lại hơn là phát triển mới tất cả các thành phần của hệ thống.
6. Phát triển sử dụng các ngôn ngữ kịch bản như TCL hoặc Perl (Ousterhout, 1998)

7. Sử dụng công cụ CASE và người viết chương trình hơn là phát triển hệ thống không có hỗ trợ.

**Bảng 11.4 Kỹ thuật ước lượng chi phí**

Kỹ thuật	Mô tả
Mô hình hoá chi phí giải thuật	Mô hình được phát triển sử dụng thông tin chi phí trong quá khứ mà nó gần với một số thước đo phần mềm (thường là kích cỡ của nó) để tính chi phí dự án. Sự ước lượng được tạo bởi thước đo và mô hình này cho biết trước các yêu cầu cần thiết.
Sự phán quyết của chuyên gia	Một số chuyên gia đề xuất kỹ thuật phát triển phần mềm và miền ứng dụng để tham khảo. Mỗi người có các ước lượng chi phí dự án. Chúng được đưa ra so sánh và thảo luận. Quá trình ước lượng được nhắc lại đến khi được tán thành và trở nên hợp lý.
Ước lượng bằng cách loại suy	Kỹ thuật này được áp dụng khi các dự án khác có cùng miền ứng dụng đã hoàn tất. Chi phí của dự án mới được ước lượng bằng cách loại suy với các dự án đã hoàn thành. Myers (Myers, 1989) đưa ra sự mô tả rất chi tiết về phương hướng này
Sự dừng đỉnh	Trạng thái dừng đỉnh được mở rộng để lấp đầy thời gian có thể sử dụng. Chi phí được ấn định bởi tài nguyên có thể sử dụng hơn là đánh giá mục tiêu. Nếu phần mềm được phân phối trong 12 tháng và 5 người có thể sử dụng, thì yêu cầu được ước lượng.
Định giá để chiến thắng	Chi phí phần mềm được ước lượng cho bất kỳ khách hàng nào có thể trả tiền cho dự án. Chi phí cho sự cố gắng phụ thuộc vào ngân sách của khách hàng chứ không phụ thuộc vào chức năng phần mềm.

Nếu những nhà quản lý không làm việc về kỹ thuật, sự xem xét kinh nghiệm có thể không giúp gì được họ trong việc ước lượng chi phí dự án phần mềm. Điều này tạo ra khó khăn cho họ trong việc đưa ra ước lượng chi phí và lịch trình chính xác.

Bạn có thể khác phục các phương hướng ước lượng trong hình 8.4 sử dụng hoặc từ phương pháp từ trên xuống hoặc từ dưới lên. Phương pháp từ trên xuống bắt đầu từ cấp hệ thống. Bạn bắt đầu với việc khảo sát các chức năng của sản phẩm và làm thể nào cung cấp các chức năng này bằng các hàm con. Chi phí của các hoạt động cấp hệ thống coi như sự quản lý tích hợp, cấu hình và hướng dẫn trong báo cáo.

Ngược lại, phương pháp từ dưới lên, bắt đầu từ cấp các thành phần. Hệ thống được phân tách thành các thành phần, và bạn ước lượng các yêu cầu để phát triển cho mỗi thành phần. Sau đó bạn cộng các chi phí thành phần để tính toán yêu cầu cho phát triển toàn bộ hệ thống.

Sự bất lợi của phương pháp từ trên xuống lại là sự thuận lợi của phương pháp từ dưới lên và ngược lại. Ước lượng từ trên xuống có thể đánh giá thấp chi phí của việc giải quyết liên kết các vấn đề kỹ thuật khó với từng phần riêng biệt như là giao diện của phần cứng không chuẩn. Không có sự chứng minh cụ thể nào của phép ước lượng được đưa ra. Ngược lại, ước lượng từ dưới lên có sự chứng minh và quan tâm tới mỗi thành phần. Tuy nhiên, phương pháp này lại có thể đánh giá thấp chi phí các hoạt động của hệ thống như sự tích hợp. Ước lượng từ dưới lên cũng đắt hơn, nó phải được khởi tạo thiết kế hệ thống để nhận biết chi phí của các thành phần.

Mỗi phương pháp ước lượng đều có những điểm mạnh và điểm yếu. Sử dụng mỗi thông tin khác nhau về các dự án và các nhóm phát triển, nếu bạn sử dụng đơn lẻ một phương pháp và các thông tin này thì sẽ không chính xác, ước lượng cuối cùng sẽ bị sai. Vì thế, cho những dự án lớn, bạn nên sử dụng một vài kỹ thuật ước lượng và so sánh kết quả của chúng. Nếu kết quả là khác nhau về cơ bản, bạn có thể không có đủ thông tin về sản phẩm hoặc quá trình phát triển, bạn nên tiếp tục quá trình ước lượng cho đến khi các kết quả là đồng nhất.

Các kỹ thuật ước lượng được áp dụng nơi văn bản yêu cầu cho hệ thống được đưa ra. Nó nên xác định tất cả các người dùng và yêu cầu hệ thống. Do đó bạn có thể có những ước lượng hợp lý về chức năng của hệ thống sẽ được phát triển. Nhìn chung, các dự án thiết kế hệ thống lớn sẽ có những văn bản yêu cầu.

Tuy nhiên, trong rất nhiều trường hợp, chi phí của nhiều dự án phải được ước lượng sử dụng các yêu cầu người dùng không thoả đáng cho hệ thống. Điều đó có nghĩa là sự ước lượng có rất ít thông tin về công việc. Phân tích yêu cầu và cụ thể hoá là đắt, và người quản lý trong công ty có thể cần có các ước lượng chi phí khởi tạo cho hệ thống trước khi họ có thể chấp nhận ngân sách để phát triển các yêu cầu cụ thể hoặc các phiên bản đầu tiên của hệ thống.

Trường hợp dưới cùng, “trả giá để dành chiến thắng” là một chiến lược chung thường sử dụng. Chú ý của việc trả giá để dành chiến thắng là nó dường như không hợp lý và không có tính thương mại. Tuy nhiên, nó có một số thuận lợi. Chi phí của dự án được chấp nhận trên cơ sở mục đích trên nguyên tắc chung. Sự điều chỉnh sau đó do khách hàng thiết lập các đặc trưng cơ bản của dự án. Các chi tiết phụ thuộc vào sự chi phí chấp nhận. Người mua và người bán phải đồng ý cái gì là chức năng hệ thống chấp nhận được. Sự yêu cầu là có thể thay đổi nhưng chi phí là không được vượt quá.

Ví dụ, một công ty trả cho một hợp đồng phát triển hệ thống phân phối nhiên liệu cho công ty dầu mà lịch phân phối nhiên liệu là tới các trạm phục vụ. Không có văn bản yêu cầu cụ thể nào cho hệ thống và ước lượng chi phí là \$900.000 như là sự cạnh tranh ngân sách các công ty dầu. Sau sự công nhận như vậy trong hợp đồng, họ đàm phán các yêu cầu cụ thể cho hệ thống và các chức năng cơ bản được phân phối, sau đó họ ước lượng

thêm các chi phí cho các yêu cầu khác. Công ty dầu không cần thiết mất cái trên bởi vì nó có hợp đồng. Các yêu cầu thêm có thể dung các ngân sách trong tương lai, vì thế ngân sách công ty dầu có thể bị phá vỡ bởi cái giá phần mềm khởi điểm quá cao.

#### **11.4. Phương pháp ước lượng trong scrum**

Các bước tính thời gian và chi phí chi phí

**Chi phí = REP /PM/FF**

**Thời gian phát hành = REP /EV (số Sprint)**

Trong đó:

- REP: Release Estimated Points = Số point ước tính của release
- PM: Point – Man = quy đổi 1 point tương ứng man-day
- EV: Estimated Velocity = Tốc độ ước tính
- FF: Focus Factor = Hệ số tập trung

Một quy trình ước tính chi phí cơ bản sẽ trải qua các bước sau đây:

**Xác định focus factor > Xác định estimated velocity > Xác định độ quan trọng và cam kết release > Ước tính chi phí**

Các chi tiết của từng bước được thảo luận kĩ hơn ở bên dưới.

##### **Xác định focus factor**

Dựa vào dữ liệu thực tế (nếu nhóm đã có sự cộng tác trước đó), tính chất của dự án, năng lực hiện có của nhóm và các tham số khác để xác định focus factor. Nếu có ít thông tin, có thể lựa chọn con số an toàn là 50%, sau đó làm mịn lại ở Sprint tiếp theo.

Số liệu FF sẽ ảnh hưởng đến capacity như thế nào?

Giả sử FF = 50%. Nhóm bạn có tổng cộng 9 developer, làm việc 5 ngày/1 tuần, Sprint 2 tuần. Vậy là bạn có  $9 \times 5 \times 2 = 90$  man-day. Nhưng FF=50% nên chỉ dùng có 45 man-day cho sản xuất, còn lại là các việc hành chính, học tập, giải trí v.v. Capacity thực sự để tính tốc độ là 45 man-day.

##### **Xác định estimated velocity (EV)**

Có một số tình huống cho việc ước tính velocity như sau:

Tình huống 1: Dự án đã chạy được một số Sprint (qua quá trình pilot, hoặc chạy thật):

Chỉ cần đếm và đo tốc độ trung bình. Các dự án inhouse, RnD có thể rơi vào tình huống này. Dễ.

Tình huống 2: Dự án mới, cần ước tính velocity (để tính được chi phí)

Cách 1: chạy pilot (hoặc calibration – tùy cách bạn gọi) một Sprint hoặc mini-Sprint (độ dài rút ngắn xuống 1 tuần hoặc ít hơn) để có dữ liệu. Cách này luôn luôn thực hiện được. Dữ liệu empirical luôn là dữ liệu thật nhất. Dĩ nhiên là bạn phải phân tích kĩ các dữ liệu đo đếm được trước khi ra quyết định cuối cùng (Count>Calculate>Judge).

Cách 2: phân tích dữ liệu lịch sử. Nếu dự án mới không quá khác so với các dự án trước đó, bạn có thể lấy dữ liệu cũ để dùng cho dữ liệu mới. Nếu dự án mới tinh, nhóm mới tinh, bạn không thể dùng được cách này.

Giả sử trước đó bạn không dùng story point để ước lượng, bạn sẽ phải quy đổi từ đơn vị cũ sang đơn vị mới. Ví dụ, trước đó chức năng “Login” được thực hiện với 5 man-day, giờ đây bạn xác định story “Login” là 1 point thì có quy đổi 1 point = 5 man-day. Nếu bạn chuyển từ waterfall sang agile, bạn có thể thực hiện quá trình calibration để biết được giá trị quy đổi thực sự. Cách làm là: chạy một mini-Sprint để pilot, đo và quy đổi (cách 1).

Còn nếu trước đó bạn đã dùng point để đo thì không có gì để bạn, biết rồi!

**Ví dụ:** Nhóm 9 người với FF là 50%, tốc độ ước tính là 50 point/Sprint\_2\_tuần, quy đổi PM=5 (tức 1 point tương ứng 5 man-day), release 1.0 tới cần 20 story với tổng cộng 200 point (REP = 500) thì:

Chi phí =  $500/5/0.5 = 200$  man-day.

Thời gian =  $500/50 = 10$  Sprint = 5 tháng.

Vậy là theo ước tính này, dự án sẽ hoàn thành release 1.0 sau 5 tháng với chi phí là 200 man-day. Nếu bạn chi cho mỗi 1 man-day là 50\$/ngày công (bao gồm mọi chi phí tiền lương, máy móc, phụ cấp v.v.) thì chi phí ước tính cho dự án là  $200*25 = 5000\$$ .



## TÀI LIỆU THAM KHẢO

- [1]. Bài giảng môn học Introduction to software Engineering, John Vu-CMU , 2008
- [2]. Kỹ nghệ phần mềm cách tiếp cận của người thực hành, Roger S.Pressman, tập 1, người dịch Ngô Trung Việt, NXB Giáo dục ,1997
- [3]. Kỹ nghệ phần mềm cách tiếp cận của người thực hành, Roger S.Pressman, tập 2, người dịch Ngô Trung Việt, NXB Giáo dục ,1997
- [4]. Software Engineering – A Practitioner’s Approach, Roger S.Pressman, 5<sup>th</sup> Edition, McGraw-Hill.Inc, 2010.
- [5]. The Unified Software Development Process, Ivar Jacobson, Grandy Booch, James Rumbaugh, 2009
- [6]. Software Engineering, Ian Sommerville, 9<sup>th</sup> Edition, Addison Wasley, 2010
- [7]. Software Engineering: A Hands-On Approach, Roger Y. Lee 2013