

MỤC LỤC

QUẢN LÝ DỰ ÁN	5
I.1 Quản lý dự án	5
I.2 Tiến trình quản lý dự án	5
I.2.1 Thiết lập một dự án phần mềm	6
I.2.2 Cách đo và độ đo	6
I.2.3 Ước lượng	7
I.2.4 Phân tích rủi ro	7
I.2.5 Lập lịch	8
I.2.6 Theo dõi và kiểm soát	8
ĐỘ ĐO PHẦN MỀM	10
II.1 Đo phần mềm	10
I.2 Phân loại các độ đo phần mềm	11
II.2.1 Độ đo theo kích cỡ	12
II.2.2 Độ đo theo chức năng	13
II.3 Đo chất lượng phần mềm	17
II.3.1 Các nhân tố ảnh hưởng	17
II.3.1.1 Các nhân tố ảnh hưởng tới chất lượng	17
II.3.2 Đo chất lượng	18
III.4 Tích hợp độ đo trong tiến trình kỹ nghệ phần mềm	20
III.4.1 Luận cứ về độ đo phần mềm	20
III.4.2 Thiết lập vạch ranh giới	21
III.4.3 Thu thập độ đo, tính toán và đánh giá	22
ƯỚC LƯỢNG DỰ ÁN PHẦN MỀM	28
III.1 Ước lượng	28
III.2 Mục tiêu lập kế hoạch dự án	29
III.3 Xác định phạm vi phần mềm	30

III.4 Ước lượng về tài nguyên.....	32
III.4.1 Tài nguyên con người.....	32
III.4.2 Tài nguyên phần cứng.....	33
III.4.3 Tài nguyên phần mềm.....	33
III.4.4 Tính sử dụng lại.....	36
III.5 Ước lượng dự án phần mềm.....	37
III.6 Kỹ thuật phân rã.....	39
III.6.1 Ước lượng LOC và FP.....	39
III.6.2 Ước lượng công sức.....	42
III.6.3 Thí dụ về các ước lượng.....	44
III.6.3.1 Ước lượng LOC và FP.....	45
III.6.3.2 Ước lượng chi phí và công sức.....	47
III.7 Mô hình ước lượng kinh nghiệm.....	48
III.8 Mô hình ước lượng Putnam.....	54
III.9 Công cụ ước lượng tự động.....	56
LẬP KẾ HOẠCH DỰ ÁN PHẦN MỀM.....	59
IV.1 Phân tích rủi ro.....	59
IV.1.1 Xác định rủi ro.....	59
IV.1.2 Dự phòng rủi ro.....	61
IV.1.3 Quản lý rủi ro.....	63
IV.2 Lập lịch dự án phần mềm.....	64
IV.2.1 Mối quan hệ con người – công việc.....	65
IV.2.2 Xác định nhiệm vụ.....	66
IV.2.3 Phân bổ nỗ lực.....	67
IV.2.4 Phương pháp lập lịch.....	68
IV.2.5 Thí dụ về lập lịch.....	70
IV.2.6 Theo dõi và kiểm soát dự án.....	70
IV.3 Tái kỹ nghệ phần mềm.....	76

IV.4 Lập kế hoạch tổ chức nhân sự.....	78
IV.5 Kế hoạch dự án phần mềm.....	80
HỆ THỐNG THỜI GIAN THỰC.....	84
V.1 Hệ thống thời gian thực.....	84
V.1.1 Vấn đề tích hợp và hiệu năng.....	85
V.1.2 Xử lý ngắt	86
V.1.3 Cơ sở dữ liệu thời gian thực	88
V.1.4 Hệ điều hành thời gian thực.....	90
V.1.5 Ngôn ngữ thời gian thực.....	91
V.1.6 Đồng bộ hoá và truyền thông nhiệm vụ	92
V.2 Phân tích và mô phỏng hệ thống thời gian thực.....	93
V.2.1 Công cụ toán học cho phân tích hệ thống thời gian thực	94
V.2.2 Các kỹ thuật mô hình hoá cho hệ thống thời gian thực	98
V.2.2.1 Cách nhìn quan niệm.....	98
V.2.2.2 Cách nhìn vật lý.....	100
V.2.2.3 Phân tích và mô phỏng	100
V.2.2.4 Kịch bản chạy.....	101
V.2.2.5 Mô phỏng lập trình	102
V.2.2.6 Dịch tự động thành mã	103
V.4 Phương pháp thiết kế hướng luồng dữ liệu.....	104
V.4.1 Các yêu cầu về phương pháp thiết kế hệ thống thời gian thực.....	104
V.4.2 DARTS.....	106
V.4.3 Thiết kế nhiệm vụ.....	108
KỸ NGHỆ WEB	110
VI.1 Các thuộc tính của ứng dụng dựa trên Web	110
VI.2 Một khuôn khổ cho kỹ nghệ Web	111
VI.3 Phát biểu/phân tích hệ thống dựa trên Web.....	113
VI.4.2 Thiết kế dẫn lái	121

<i>VI.4.3 Thiết kế giao diện.....</i>	<i>122</i>
<i>VI.5 Kiểm thử ứng dụng dựa trên Web</i>	<i>124</i>
<i>VI.6 Vấn đề quản lí.....</i>	<i>126</i>
<i>VI. 6. 1 Tổ kĩ nghệ Web</i>	<i>126</i>
<i>VI. 6. 2 Quản lý dự án</i>	<i>128</i>
<i>SƠ LƯỢC VỀ UML (Unified Modeling Language).....</i>	<i>132</i>
<i>I. Giới thiệu</i>	<i>132</i>
<i>II. Bảng từ vựng của UML.....</i>	<i>132</i>
<i>II.1 Các sự vật.....</i>	<i>133</i>
<i>II.2 Các quan hệ</i>	<i>136</i>
<i>II.3 Các biểu đồ.....</i>	<i>137</i>
<i>III. Các quy tắc của UML.....</i>	<i>138</i>
<i>III.1 Các quy tắc ngữ nghĩa</i>	<i>138</i>
<i>III.2 Các quy tắc xây dựng mô hình</i>	<i>139</i>
<i>IV. Các cơ chế chung.....</i>	<i>139</i>
<i>Phụ lục.....</i>	<i>142</i>

CHƯƠNG I

QUẢN LÝ DỰ ÁN

I.1 Quản lý dự án

Trong việc quản lý dự án phần mềm, một phát biểu có lẽ đã được nhiều cố vấn kỹ nghệ phần mềm nhắc tới, đó là những mô tả của Page-Jones về những triệu chứng nảy sinh từ một loạt các vấn đề quản lý và kỹ thuật:

"Tôi đã đi thăm hàng chục công ty, cả tốt lẫn xấu, tôi đã quan sát rất nhiều nhà quản lý xử lý dữ liệu. Thường tôi khiếp hãi khi thấy các nhà quản lý này vật lộn vô ích với các dự án ác mộng, luồn lách qua những hạn chót không thể tưởng tượng nổi, hoặc các hệ thống đã được bàn giao, coi thường người dùng và tốn hết thời gian bảo hành".

Quản lý dự án phần mềm là tảng đầu trong tiến trình kỹ nghệ phần mềm, ta gọi nó là tảng vì thay vì là một bước bởi vì nó phủ lên toàn bộ tiến trình phát triển từ đầu tới cuối. Để tiến hành một dự án phần mềm thành công ta phải hiểu phạm vi công việc cần làm, những rủi ro phải chịu, nguồn nhân lực cần tới, nhiệm vụ cần hoàn thành, những cột mốc cần theo dõi, công sức (chi phí) phải chi tiêu, và lịch biểu phải tuân thủ. Việc quản lý dự án phần mềm cung cấp những hiểu biết thuộc loại đó. Nó bắt đầu trước khi công việc kỹ thuật bắt đầu, tiếp tục khi phần mềm tiến hoá từ khái niệm thành hiện thực và tới khi phần mềm đã được "nghỉ" hẳn.

Trong chương này, chúng ta sẽ xem xét các khái niệm dẫn tới việc quản lý dự án phần mềm hiệu quả.

I.2 Tiến trình quản lý dự án

Quản lý phần mềm rất quan trọng cho sự thành công của dự án nên những người phụ trách dự án đều phải hiểu cách tiến hành nó, mọi người hành nghề đều

phải hiểu cách làm việc trong khuôn khổ do nó lập ra. Nhưng không may là nhiều người lại không hiểu như vậy. Các mục sau trình bày một tổng quan về các yếu tố chủ chốt trong việc quản lý dự án phần mềm.

I.2.1 Thiết lập một dự án phần mềm

Trước khi lập kế hoạch dự án, cần phải thiết lập các mục tiêu và phạm vi của dự án, xem xét các giải pháp khác nhau, xác định các ràng buộc kỹ thuật và quản lý. Không có những thông tin này thì không thể xác định được những ước lượng hợp lý, chính xác về chi phí, không thể chia nhỏ các nhiệm vụ dự án, không thể xác định được một lịch biểu dự án nhằm cung cấp những chỉ dẫn có ý nghĩa về tiến độ.

Người phát triển phần mềm phải gặp gỡ nhau để xác định các mục tiêu và phạm vi dự án. Các mục tiêu sẽ xác định được mục đích toàn bộ của dự án mà chưa xét tới cách đạt được các mục đích này. Phạm vi sẽ xác định ra các chức năng chủ yếu mà phần mềm phải thực hiện, và điều quan trọng hơn, *cố gắng hạn chế các chức năng này theo cách định lượng*. Một khi các mục tiêu và phạm vi đã được hiểu rõ thì có thể xem xét tới các giải pháp khác.

I.2.2 Cách đo và độ đo

Trong phần lớn những cố gắng kỹ thuật, cách đo và độ đo giúp cho ta hiểu được tiến trình kỹ thuật được dùng để phát triển một sản phẩm và bản thân sản phẩm. Tiến trình này được đo theo nỗ lực *cải thiện* nó. Sản phẩm được đo theo nỗ lực tăng *chất lượng* của nó.

Đầu tiên, dường như là việc đo là điều hiển nhiên. Sau cùng, việc đo làm cho ta định lượng được và do đó quản lý hiệu quả hơn. Nhưng thực tế có thể hơi khác. Việc đo thường dẫn tới tranh cãi và biện minh. Đây là độ đo thích hợp cho tiến trình và sản phẩm? Nên dùng dữ liệu thu thập được như thế nào? Liệu việc dùng cách đo để so sánh con người, tiến trình hay sản phẩm thì có tốt không? ...Trong những mục tới chúng ta sẽ xem xét các độ đo phần mềm. Các độ đo này đã được phát triển để cung cấp cho các nhà quản lý và người hành nghề kỹ thuật cái nhìn thông tỏ vào tiến trình kỹ nghệ phần mềm và sản phẩm do nó tạo ra.

1.2.3 Ước lượng

Một trong những hoạt động thử nghiệm trong tiến trình quản lý dự án phần mềm là việc *lập kế hoạch*. Khi một dự án phần mềm được lập kế hoạch, cần phải đưa ra những ước lượng về công sức con người cần có, thời hạn dự án theo ngày tháng và chi phí của dự án. Những việc này được tiến hành như thế nào? Trong nhiều trường hợp ước lượng được thực hiện bằng cách dùng kinh nghiệm quá khứ xem như hướng dẫn duy nhất. Nếu một dự án mới rất giống về kích cỡ và chức năng với một dự án qua khứ thì rất có thể là dự án mới thì rất có thể dự án mới sẽ đòi hỏi xấp xỉ khối lượng công sức, mất một khoảng thời gian, chi phí như dự án cũ. Nhưng điều gì sẽ xảy ra nếu dự án mở ra một miền đất mới. Khi đó chỉ riêng kinh nghiệm quá khứ có thể không đủ.

Người ta đã phát triển một số kỹ thuật ước lượng để phát triển phần mềm. Mặc dầu mọi kỹ thuật đều có điểm mạnh điểm yếu của nó, nhưng đều có thuộc tính chung như sau:

- Phải thiết lập phạm vi dự án trước tiên
- Độ đo phần mềm (cách đo quá khứ) được dùng làm cơ sở tiến hành ước lượng
- Dự án cần phải chia thành từng phần nhỏ để ước lượng riêng biệt
- Nhiều nhà quản lý áp dụng một số kỹ thuật ước lượng khác nhau, sử dụng kỹ thuật này để kiểm tra chéo cho kỹ thuật kia.

1.2.4 Phân tích rủi ro

Bất kỳ khi nào một chương trình máy tính được xây dựng, luôn có những miền không chắc chắn. Liệu nhu cầu khách hàng đã được hiểu rõ chưa? Liệu những chức năng cần phải thực hiện có được hoàn thiện trước hạn chót của dự án không? Liệu có những vấn đề kỹ thuật hiện bị che khuất bởi cách nhìn hiện tại không? Liệu những thay đổi có làm cho lịch biểu bị trượt quá đáng không?

Phân tích rủi ro là điều chủ chốt cho việc quản lý dự án phần mềm tốt, trong thực tế có rất nhiều dự án được tiến hành mà không tính đến rủi ro (rủi ro ở đây có thể là rủi ro về dự án, hoặc về kỹ thuật). Như vậy: "Nếu bạn không tích cực tấn công vào các rủi ro thì chúng sẽ tích cực tấn công bạn"

Phân tích rủi ro thực tế là một loạt các bước quản lý giúp chúng ta tấn công được rủi ro, như việc: xác định rủi ro, đánh giá rủi ro, phân loại rủi ro chiến lược quản lý rủi ro, giải quyết rủi ro, điều khiển rủi ro. Các bước này được áp dụng trong suốt tiến trình kỹ nghệ phần mềm (mô hình xoắn ốc).

I.2.5 Lập lịch

Mọi dự án phần mềm đều có lịch biểu tiến hành, nhưng không phải tất cả các lịch biểu đều được tạo ra như nhau. Liệu lịch biểu tiến hoá theo cách riêng của nó hay theo kế hoạch lập từ trước. Liệu công việc được thực hiện theo mong muốn hay theo một tập các nhiệm vụ đã được xác định rõ. Người quản lý chỉ tập trung vào hạn chót hay đã xác định được đường găng (ranh giới) và điều phối đường găng này để đảm bảo đạt được hạn chót (deadline). Tiến độ có được đo theo các tập mốc đã được định sẵn.

Lập lịch cho dự án phần mềm thực sự không khác việc lập lịch cho bất kỳ dự án kỹ nghệ nào. Cần xác định một tập các nhiệm vụ dự án. Cần thiết lập các mối quan hệ tương hỗ giữa các nhiệm vụ này. Thiết lập cả những nỗ lực gắn với từng nhiệm vụ đó. Phân bổ nhân lực và các nguồn tài nguyên khác. Sau đó phải tạo ra một mạng lưới nhiệm vụ và xây dựng lịch biểu.

I.2.6 Theo dõi và kiểm soát

Một khi lịch biểu phát triển đã được thiết lập thì hoạt động theo dõi và kiểm soát bắt đầu. Mỗi nhiệm vụ được ghi trong lịch đều được người quản lý dự án theo dõi. Nếu nhiệm vụ bị lệch khỏi lịch biểu thì người quản lý có thể dùng một công cụ lập lịch dự án tự động hoá để xác định tác động của việc trượt lịch lên các mốc trung gian và ngày bàn giao toàn bộ. Có thể bố trí lại tài nguyên, có thể đảo lại các nhiệm vụ hay sửa lại các cam kết bàn giao cho phù hợp với vấn đề còn chưa bao quát hết. Theo cách này việc phát triển phần mềm có thể được kiểm soát tốt hơn.

Tóm tắt

Việc quản lý dự án phần mềm biểu thị cho tầng đầu tiên của tiến trình kỹ nghệ phần mềm. Việc quản lý dự án bao gồm các hoạt động kể cả đo, ước lượng, phân tích rủi ro, lập lịch, theo dõi tiến độ và kiểm soát.

CHƯƠNG II

ĐỘ ĐO PHẦN MỀM

Phần này xem xét vai trò của việc đo, độ đo phần mềm và những ảnh hưởng của chúng lên việc quản lý dự án.

Độ đo phần mềm nói tới một phạm vi rộng các cách đo phần mềm máy tính. Trong khung cảnh quản lý dự án chúng ta quan tâm chủ yếu đến độ đo hiệu suất (các cách đo cái ra) và chất lượng (cách đo "sự phù hợp sử dụng" của cái ra). Trong các mục sau đây, trước hết chúng ta xem xét một phạm vi rộng các độ đo phần mềm để hiểu cách đo hiệu suất và chất lượng thích hợp vào đâu. Sau đó hai cách nhìn quan trọng, đối lập nhau về cách đo tính hiệu suất và chất lượng phần mềm sẽ được trình bày. Cuối cùng chúng ta xem xét các vấn đề thực tế của việc thu thập các độ đo và dùng chúng để ước lượng dự án phần mềm.

II.1 Đo phần mềm

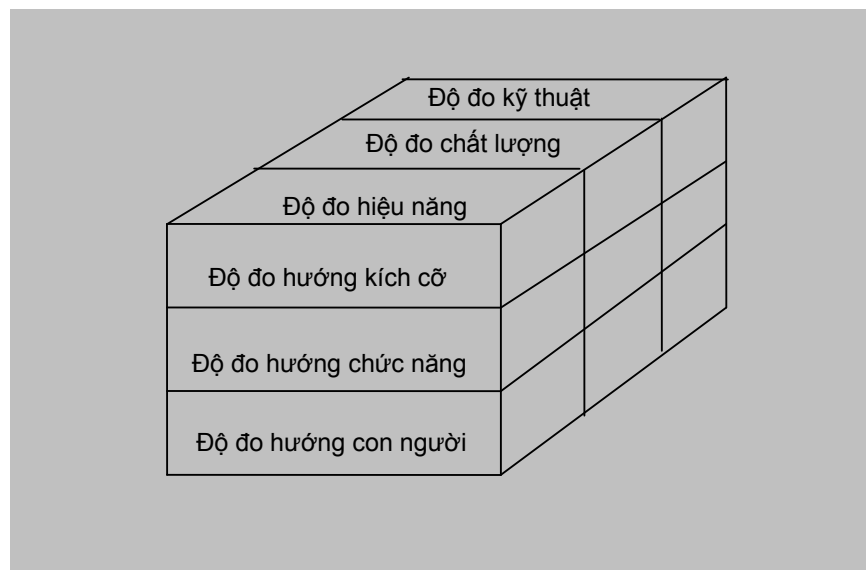
Những lý do cần phải đo phần mềm:

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

Có hai cách đo: đo trực tiếp và đo gián tiếp. **Đo trực tiếp:** Cách đo trực tiếp áp dụng đối với tiến trình kỹ nghệ phần mềm, bao gồm việc đo về công sức và chi phí bỏ ra, đối với 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 trong một thời kỳ nào đó. **Đo**

gián tiếp: Việc đo gián tiếp áp dụng đối với sản phẩm bao gồm việc đo chức năng, chất lượng, độ phức tạp, tính hiệu quả, độ tin cậy, tính bảo trì và nhiều khả năng khác.

I.2 Phân loại các độ đo phần mềm



H2.1 Phân loại độ đo

Như đã lưu ý, độ đo hiệu năng tập trung vào cái ra của tiến trình kỹ nghệ phần mềm, độ đo chất lượng đưa ra một chỉ dẫn về việc phần mềm tuân thủ đến đâu đối với các yêu cầu của người dùng. Còn độ đo kỹ thuật tập trung vào đặc trưng phần mềm (độ phức tạp, mức modul...)

Theo như Hình 2.1, chúng ta có thể phát triển cách phân loại thứ hai. Độ đo theo kích cỡ được dùng để thu thập cái ra và chất lượng kỹ nghệ phần mềm. Còn độ đo theo chức năng đưa ra các cách đo gián tiếp, độ đo theo con người thì thu thập tổng

tin về cách thức để con người phát triển phần mềm máy tính và cảm nhận của con người về tính hiệu quả của công cụ và phương pháp.

II.2.1 Độ đo theo kích cỡ

Dự án	Công sức	1000\$	KLOC	Trang tư liệu	Lỗi	Người
aaa-01	24	168	12.1	365	29	3
ccc-04	62	440	27.2	1224	86	5
fff-03	43	314	20.7	1050	64	6

H2.2 Độ đo theo kích cỡ

Độ đo phần mềm theo kích cỡ là các cách đo trực tiếp cho phần mềm và tiến trình phát triển nó. Một tổ chức làm phần mềm có thể tạo ra một bảng dữ liệu theo kích cỡ, bảng này liệt kê từng dự án phát triển phần mềm đã được hoàn thành trong vài năm qua và dữ liệu kích cỡ tương ứng cho dự án đó.

Ta hãy xem xét các ô trong bảng dành cho dự án aaa-01: 12.1 KLOC (K-ngàn) câu lệnh chương trình đã được xây dựng với công sức của 24 người - tháng và chi phí 168000 \$, đã viết được 365 trang tư liệu, gặp phải 29 lỗi sau khi bàn giao cho khách hàng trong năm vận hành đầu tiên, có 3 người làm cho việc phát triển phần mềm cho dự án.

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 từng dự án, tính được giá trị trung bình cho mọi dự án.

Hiệu năng = KLOC / người-tháng

Chất lượng = khiếm khuyết / KLOC

Ngoài ra còn tính được các độ đo 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 để đo tiến trình phát triển phần mềm.

II.2.2 Độ đo theo chức năng

Độ đo phần mềm theo chức năng là các cách đo gián tiếp cho phần mềm và tiến trình phát triển nó. Thay vì đếm LOC, độ đo theo chức năng tập trung vào các chức năng hay tiện ích của chương trình. Cách tiếp cận đo hiệu năng tên là phương pháp *điểm chức năng* (FP). Điểm chức năng được tính bằng cách hoàn thành bảng được vẽ dưới đây

Tham biến độ đo	Số đếm	Nhân tố trọng số			
		Đơn giản	Trung bình	Phức tạp	
Số cái vào theo người dùng	<input type="text"/>	*	3	4	6 = <input type="text"/>
Số cái ra theo người dùng	<input type="text"/>	*	4	5	7 = <input type="text"/>
Số yêu cầu người dùng	<input type="text"/>	*	3	4	6 = <input type="text"/>
Số các tệp	<input type="text"/>	*	7	10	15 = <input type="text"/>
Số các giao diện ngoài	<input type="text"/>	*	5	7	10 = <input type="text"/>
Số đếm tổng cộng					→ <input type="text"/>

H2.3 Tính độ đo điểm chức năng

Năm đặc trưng miền thông tin sẽ được xác định và số đếm được nêu ra tại các vị trí bảng thích hợp.

- Số cái vào theo người dùng: Phải tính cái vào mà từng người dùng cung cấp dữ liệu theo ứng dụng phân biệt cho phần mềm.
- Số cái ra theo người dùng: Phải tính tới cái ra mà từng người dùng cung cấp thông tin theo ứng dụng cho phần mềm (các báo cáo, thông báo lỗi...).
- Số các câu hỏi của người dùng: Câu hỏi được định nghĩa như một cái vào trực tuyến trong việc sinh ra một đáp ứng ngay lập tức của phần mềm dưới dạng một cái ra trực tuyến.
- Số các giao diện ngoài: Mọi giao diện máy đọc được (như các tệp dữ liệu trên đĩa) được dùng để truyền thông tin sang hệ thống khác đều phải được tính tới.

Một khi những dữ liệu trên đã được thu nhận thì có thể gán cho từng số đếm một giá trị độ 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 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:

$$FP = \text{tổng số đếm} * [0.65 + 0.01 * \text{SUM}(Fi)] \quad (2.1)$$

với tổng số đếm là tổng của tất cả các mục FP có được trong bảng trong hình Fi (i=1 tới 14) là “giá trị điều chỉnh độ phức tạp” dựa trên việc đáp ứng với các câu hỏi trong bảng dưới. Các giá trị hằng trong phương trình trên và nhân tố trọng số được áp dụng cho các số đếm lĩnh vực thông tin 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

Tỷ lệ lỗi nhân tố theo từng 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

Bảng 2.1 Tính điểm chức năng

Cách đo điểm chức năng ban đầu được thiết kế để áp dụng cho các ứng dụng hệ thống thông tin nghiệp vụ. FP phụ thuộc vào ngôn ngữ lập trình, làm cho nó trở thành lý tưởng cho các ứng dụng có dùng ngôn ngữ quy ước và phi thủ tục. Nó dựa trên dữ liệu và rất có thể dùng từ rất sớm trong tiến hoá của một dự án, làm cho nó trở

nên hấp dẫn hơn như một cách tiếp cận ước lượng. Nhược điểm của phương pháp này là đòi hỏi một số khả năng thủ công trong đó tính toán dựa nhiều vào chủ quan chứ không khách quan, thông tin về lĩnh vực thông tin có thể là khó thu nhập...

Ngoài ra có cách đo các *điểm tính năng* thích hợp cho các ứng dụng trong đó độ phức tạp thuật toán là cao. Các ứng dụng thời gian thực, điều khiển tiến trình và phần mềm nhúng có khuynh hướng có độ phức tạp thuật toán cao và do đó dùng được điểm tính năng.

Để tính điểm tính năng, lại phải tính 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 tới cả tới một đặc trưng phần mềm mới, thuật toán. Thuật toán được định nghĩa là một vấn đề tính toán có giới hạn được bao hàm bên trong một chương trình máy tính đặc biệt.

Để tính các điểm chức năng, ta dùng bảng sau vẽ trong Hình 2.4. Một giá trị trong số đơn được dùng cho mỗi tham biến đo và giá trị điểm chức năng toàn thể được tính bằng cách dùng phương trình (2.1).

Tham biến độ đo	Số đếm	Trọng số	
Số cái vào theo người dùng	<input type="text"/>	* 4 =	<input type="text"/>
Số cái ra theo người dùng	<input type="text"/>	* 5 =	<input type="text"/>
Số yêu cầu người dùng	<input type="text"/>	* 4 =	<input type="text"/>
Số các tệp	<input type="text"/>	* 7 =	<input type="text"/>
Số các giao diện ngoài	<input type="text"/>	* 7 =	<input type="text"/>
Thuật toán	<input type="text"/>	* 3 =	<input type="text"/>
Số đếm tổng cộng			<input type="text"/>

H2.4 Tính độ đo điểm tính năng

Cần lưu ý rằng các điểm tính năng và các điểm chức năng biểu thị cho cùng một điều – “chức năng” hay “tiện ích” do phần mềm cung cấp. Trong thực tế, cả hai cách đo đều cho cùng giá trị của FP đối với các tính toán công nghệ qui ước hay hệ thống

thông tin. Đối với các hệ thống thời gian thực phức tạp hơn, số điểm tính năng thường giữa 20 và 35 phần trăm cao hơn số điểm được xác định bằng cách dùng một mình điểm chức năng.

II.3 Đo chất lượng phần mềm

Có thể đo được chất lượng thông qua tiến trình kỹ nghệ phần mềm và sau khi phần mềm đã được giao cho khách hàng. Các độ đo trước khi bàn giao phần mềm đưa ra một cơ sở định lượng cho việc ra quyết định thiết kế và kiểm thử. Độ đo chất lượng trong phân loại này bao gồm độ đo phức tạp chương trình, tính modul hiệu quả, kích cỡ chương trình tổng thể. Các độ đo được dùng sau khi bàn giao tập trung vào số các khiếm khuyết chưa được phát hiện trong lĩnh vực và tính bảo trì được của hệ thống. Điều cần nhấn mạnh là các cách đo sau khi bàn giao về chất lượng phần mềm để trình bày với các nhà quản lý và nhân viên kỹ thuật một chỉ báo về tính hiệu quả của tiến trình kỹ nghệ phần mềm.

II.3.1 Các nhân tố ảnh hưởng

II.3.1.1 Các nhân tố ảnh hưởng tới chất lượng

Các nhân tố ảnh hưởng tới chất lượng phần mềm định giá phần mềm theo ba quan điểm phân biệt:

- (1) thao tác sản phẩm
- (2) xem xét sản phẩm (thay đổi)
- (3) chuyển đổi sản phẩm

II.3.1.2 Các nhân tố ảnh hưởng tới hiệu năng phần mềm

Nhân tố con người: Kích cỡ và mức độ chuyên gia của tổ chức phát triển.

Nhân tố vấn đề: Độ phức tạp của vấn đề cần giải quyết và số những thay đổi trong ràng buộc và yêu cầu thiết kế.

Nhân tố sản phẩm: Độ tin cậy và hiệu suất của hệ thống dựa trên máy tính.

Nhân tố tài nguyên: Có sẵn các công cụ CASE, các tài nguyên phần cứng và phần mềm.

Nhân tố	Biến thiên % xấp xỉ
Nhân tố con người	90
Nhân tố vấn đề	40
Nhân tố tiến trình	50
Nhân tố sản phẩm	140
Nhân tố tài nguyên	40

Giả sử rằng hai nhóm kỹ nghệ phần mềm có số người với kỹ năng như nhau, dùng cùng tài nguyên và tiến trình. Một trong hai nhóm làm việc với vấn đề tương đối đơn giản với độ tin cậy trung bình và đòi hỏi độ hoàn thiện trung bình. Nhóm kia làm việc trên vấn đề phức tạp với mục tiêu độ tin cậy và hiệu suất cực kỳ cao. Dựa trên các con số danh sách ở trên, nhóm thứ nhất có thể trình bày hiệu năng phát triển phần mềm giữa 40 và 140 phần trăm tốt hơn hiệu năng của nhóm thứ hai. Nhân tố vấn đề và sản phẩm làm cho việc so sánh hiệu năng giữa hai nhóm trở thành vô nghĩa.

II.3.2 Đo chất lượng

Mặc dầu có nhiều cách đo chất lượng phần mềm, các cách đo thảo luận trước đây là những cách đã được dùng rộng rãi nhất, chúng bao gồm: tính đúng đắn, tính bảo trì, tính toàn vẹn và tính dùng được. Sau đây là định nghĩa và độ đo cho từng cách đo:

Tính đúng đắn: Một chương trình phải vận hành đúng đắn nếu không nó sẽ có ít giá trị với người dùng. Tính đúng đắn là mức độ mà phần mềm thực hiện được các chức năng mong đợi của nó. Cách đo thông dụng nhất cho tính đúng đắn là *số khiếm khuyết theo KLOC*, với số khiếm khuyết được định nghĩa như việc thiếu tuân thủ theo yêu cầu đã được kiểm chứng. Khiếm khuyết do người dùng báo cáo lại về chương trình sau khi chương trình sau khi chương trình đã được xuất xưởng để dùng đại trà được và được tính theo từng thời kỳ chuẩn, điển hình là một năm.

Tính bảo trì: Việc bảo trì phần mềm tốn nhiều công sức hơn bất kỳ hoạt động kỹ nghệ phần mềm nào khác. Tính bảo trì là dễ dàng đối với chương trình có thể sửa được nếu gặp lỗi, có thể thích nghi nếu môi trường thay đổi, hay có thể được nâng cao thêm nếu khách hàng mong muốn thay đổi. Không có cách nào đo được trực tiếp tính bảo trì: do đó chúng ta phải dùng cách đo gián tiếp. Một độ đo theo thời gian đơn giản là thời gian thay đổi trung bình (MTTC), khoảng thời gian cần cho việc phân tích sự thay đổi yêu cầu, thiết kế sửa đổi thích hợp, cài đặt sự thay đổi, kiểm thử nó, và phân phát sự thay đổi đó cho mọi người dùng. Về trung bình, các chương trình bảo trì được sẽ có MTTC thấp (với kiểu thay đổi tương đương) hơn là các chương trình không bảo trì được.

Chúng ta dùng một độ đo theo chi phí cho tính bảo trì gọi là “hư hỏng” - chi phí để sửa lại những khiếm khuyết gặp phải sau khi phần mềm được bàn giao cho người dùng cuối cùng. Khi tỉ lệ hư hỏng trên tổng chi phí dự án (đối với nhiều dự án) được vẽ như một hàm theo thời gian thì nhà quản lý có thể xác định liệu tính bảo trì toàn bộ cho phần mềm do cơ quan phát triển phần mềm tạo ra có được cải thiện hay không. Có thể tiến hành những hoạt động để đáp ứng với các hiểu thấu thu được từ thông tin này.

Tính toàn vẹn: Tính toàn vẹn phần mềm đã trở nên ngày càng quan trọng trong thời đại của những kẻ mày mò và virus. Thuộc tính này do khả năng của một hệ thống trụ được với sự tấn công (cả ngẫu nhiên lẫn chủ định) vào sự an toàn của nó. Những công kích có thể được tiến hành trên ba thành phần: chương trình, dữ liệu và tư liệu.

Để đo tính toàn vẹn, cần phải xác định hai thuộc tính phụ: sự đe dọa và độ an toàn. Đe dọa là xác suất (có thể được ước lượng hay suy dẫn từ những bằng chứng kinh nghiệm) để cho việc công kích thuộc một kiểu đặc biệt sẽ xuất hiện trong thời gian đã nêu. An toàn là xác suất (có thể được ước lượng hay suy dẫn từ những bằng chứng kinh nghiệm) rằng sự công kích thuộc một đặc biệt nào đó bị đẩy lùi.

Tính toàn vẹn của một hệ thống có thể được định nghĩa là:

$$\text{Tính toàn vẹn} = \sum [1 - \text{đe dọa} * (1 - \text{an toàn})]$$

với đe dọa và an toàn được lấy theo từng kiểu công kích.

Tính sử dụng được: Câu cửa miệng thân thiện người dùng đã trở thành thông dụng trong các thảo luận về sản phẩm phần mềm. Nếu một chương trình không thân thiện người dùng thì nó thường thất bại, cho dù các chức năng mà nó thực hiện là có giá trị. Tính sử dụng là một nỗ lực để định lượng sự thân thiện người dùng và có thể được đo dưới dạng bốn đặc trưng:

- kỹ năng vật lý và hoặc trí tuệ để đọc hệ thống
- thời gian cần để trở thành tương đối hiệu quả trong việc dùng hệ thống
- sự tăng lên rõ rệt về hiệu năng được đo khi hệ thống được dùng bởi người dùng trung bình
- xác nhận chủ quan (đôi khi thu được qua một bảng hỏi) của thái độ người dùng hướng về hệ thống

Bốn nhân tố được mô tả ở trên là một cách lấy mẫu cho những điều đã được đề nghị cho cách đo chất lượng phần mềm (đã trình bày kỹ trong học phần một). Các điểm chức năng và LOC tỏ ra là các công cụ tương đối chính xác về công sức và chi phí phát triển phần mềm. Tuy nhiên để dùng được LOC và FP trong các kỹ thuật được mô tả thì còn cần phải thiết lập được một nền thông tin lịch sử. Trong mục tiếp theo sẽ trình bày về những hướng dẫn thu thập dữ liệu độ đo.

III.4 Tích hợp độ đo trong tiến trình kỹ nghệ phần mềm

Trong mục này ta xem xét một số *luận cứ* về độ đo phần mềm và tiếp cận cách thiết lập một chương trình thu thập các độ đo bên trong tổ chức kỹ nghệ phần mềm.

III.4.1 Luận cứ về độ đo phần mềm

Tại sao việc đo tiến trình của kỹ nghệ phần mềm và sản phẩm và sản phẩm mà nó tạo ra lại quan trọng đến như vậy?

Bằng cách yêu cầu và tính toán cách đo tính hiệu năng và chất lượng, một nhà quản lý cấp cao có thể thiết lập các mục tiêu có nghĩa để cải tiến tiến trình kỹ nghệ phần mềm. Nhưng để thiết lập các mục tiêu cải tiến, cần hiểu rõ hiện trạng của tiến

trình phát triển phần mềm. Do đó việc đo được dùng để thiết lập một vạch ranh giới tiến hành mà việc cải tiến có thể được xác nhận.

Tính nghiêm ngặt từng ngày của công việc dự án phần mềm gần như không để lại thời gian cho việc suy nghĩ chiến lược. Hàng ngày, các nhà quản lý dự án phần mềm bận tâm với nhiều việc: phát triển cách ước lượng dự án có nghĩa; tạo ra hệ thống chất lượng cao hơn; nhận sản phẩm ra lò đúng hạn. Bằng cách dùng việc đo để thiết lập một vạch ranh giới, các vấn đề trên dễ quản lý hơn. Chúng ta đã lưu ý rằng vạch ranh giới dùng như một cơ sở cho việc ước lượng. Bên cạnh đó, tập hợp các độ đo chất lượng làm cho một tổ chức “điều chỉnh” được tiến trình kỹ nghệ phần mềm để khắc phục một số nguyên nhân gây khiếm khuyết ảnh hưởng đến việc phát triển phần mềm.

Tại mức kỹ thuật, độ đo phần mềm khi được áp dụng cho sản phẩm sẽ đưa ra lợi ích ngay lập tức. Khi thiết kế phần mềm được hoàn tất thì phần lớn các nhà phát triển phần mềm đều lo lắng:

- Yêu cầu nào của người dùng là hay thay đổi nhất?
- Đơn thể nào trong hệ thống này là hay sinh lỗi nhất?
- Phải lập kế hoạch kiểm thử như thế nào cho từng modul?
- Sẽ có bao nhiêu lỗi khi việc kiểm thử bắt đầu?

Câu trả lời cho những câu hỏi này có thể được xác định nếu ta đã thu thập được các độ đo và dùng như một bản hướng dẫn kỹ thuật. Trong chương sau ra sẽ xem xét cách thực hiện điều này.

III.4.2 Thiết lập vạch ranh giới

Bằng cách thiết lập một vạch ranh giới độ đo, ta có thể thu được lợi ích ở mức chiến lược, dự án và kỹ thuật. Thông tin được thu nhập không nhất thiết phải khác biệt cơ bản để thoả mãn cho từng yếu tố khác nhau đã được thảo luận ở trên. Các thức để trình bày thông tin sẽ khác nhau, nhưng bản thân các độ đo có thể phục vụ cho nhiều mục đích.

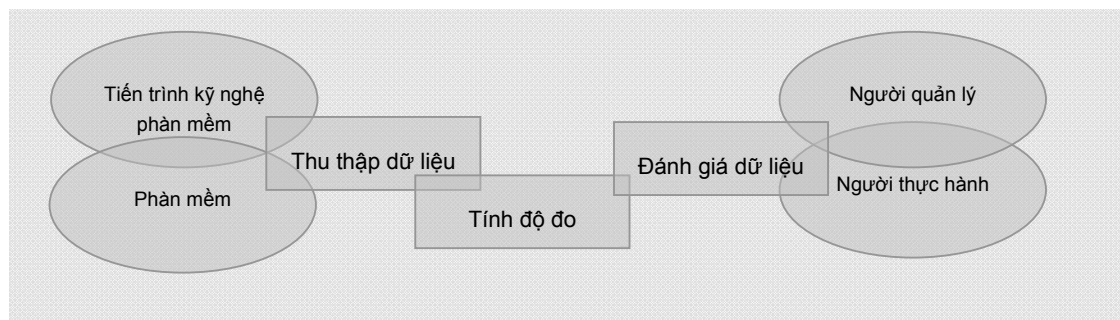
Vạch ranh giới bao gồm dữ liệu được thu thập từ các dự án phát triển phần mềm quá khứ và có thể đơn giản như được trình bày trong Hình 2.2 hay phức tạp như trong minh hoạ Hình 2.6. Bổ sung vào các cách đo hướng chức năng hay kích cỡ đơn giản, vạch ranh giới có thể được bổ sung thêm với độ đo chất lượng như được mô tả trong mục đo chất lượng.

Các thuộc tính của vạch ranh giới:

- dữ liệu phải chính xác hợp lý – nên tránh dự đoán về các dự án quá khứ
- dữ liệu nên được thu thập cho thật nhiều các dự án
- cách đo phải nhất quán, tức là một dòng mã phải được hiểu nhất quán qua tất cả các dự án có dữ liệu được thu thập
- ứng dụng phải tương tự với công việc được ước lượng (vì không có ý nghĩa khi dùng vạch ranh giới cho các công việc của hệ thống thông tin theo lô để ước lượng ứng dụng vì xử lý thời gian thực)

III.4.3 Thu thập độ đo, tính toán và đánh giá

Tiến trình thiết lập vạch ranh giới được minh hoạ trong Hình 2.5. Một cách lý tưởng cần thu thập dần dữ liệu cho việc thiết lập vạch ranh giới. Do đó việc thu thập dữ liệu đòi hỏi phải có điều tra lịch sử về các dự án quá khứ để xây dựng lại dữ liệu cần thiết. Một khi dữ liệu đã được thu thập thì việc tính toán độ đo mới có thể tiến hành được. Tuỳ theo mức độ chính xác của dữ liệu được thu thập các độ đo có thể trải trên miền rộng các cách đo LOC hay FP. Cuối cùng, dữ liệu được tính toán phải đánh giá và áp dụng trong việc ước lượng.



H2.5 Tiến trình thu thập độ đo phần mềm

Bảng dưới trình bày một mô hình trang tính để thu thập và tính toán dữ liệu vạch ranh giới phần mềm lịch sử. Lưu ý rằng mô hình này bao gồm dữ liệu chi phí, dữ liệu hướng kích cỡ, và dữ liệu hướng chức năng, cho phép tính toán cả các độ đo theo LOC và FP. Tuy nhiên không phải bao giờ cũng có thể thu nhập được tất cả các dữ liệu cần tới trong mô hình này. Nếu chúng ta áp dụng mô hình này cho một số dự án quá khứ thì có thể thiết lập một vạch ranh giới độ đo phần mềm.

CÁI VÀO DỮ LIỆU CHI PHÍ		
Mô tả	Đơn vị	Dữ liệu mẫu
Giá lao động	\$/người - tháng	\$7744
Lao động năm	giờ/năm	1460
DỮ LIỆU CHO TÍNH TOÁN ĐỘ ĐO		
Mô tả	Đơn vị	Dữ liệu mẫu
Tên hay mã hiệu dự án	alphanumeric	Proj#1
Kiểu đưa ra (phát triển/bảo trì)	alphanumeric	bảo trì
Số thành viên dự án	người	3
Công sức	người - giờ	4800
Công sức (đã tính)	người - tháng	36.9
Thời gian cần để hoàn thành	tháng	13.0
Dòng lệnh mới phát triển	KLOC	11.5
Dòng lệnh sửa đổi (mã cũ)	KLOC	0.4
Dòng lệnh đã bàn giao (kể cả mã cũ)	KLOC	33.4
Dòng lệnh dừng lại (từ CT/TV khác)	KLOC	0.8
Số chương trình tách biệt	chương trình	1
Tư liệu kỹ thuật	trang	265
Tư liệu người dùng	trang	122
Số lỗi sau 1 năm đưa ra	lỗi	26

Bảo trì công sức sửa đổi (1 năm)		người - giờ	3550
Bảo trì lỗi công sức (1 năm)		người - giờ	1970
DỮ LIỆU DỰ ÁN			
Mô tả		Đơn vị	Dữ liệu mẫu
Phần trăm toàn bộ thời gian dự án trên			
	Phân tích và đặc tả vấn đề	%	18%
	Thiết kế	%	20%
	Mã hóa	%	23%
	Kiểm thử	%	25%
	Mô tả khác	%	14%
CÁI VÀO DỮ LIỆU HƯỚNG CHỨC NĂNG			
Mô tả		Đơn vị	Dữ liệu mẫu
Miền thông tin			
	1. số cái vào người sử dụng	cái vào	24
	2. số cái ra người sử dụng	cái ra	46
	3. số câu hỏi sử dụng	câu hỏi	8
	4. số tệp	tệp	4
	5. số giao diện ngoài	giao diện	2
Giá trị trọng số			
	Dùng số đầu cho <i>đơn giản</i> , số thứ hai cho <i>trung bình</i> , số cuối cho <i>phức tạp</i>		
	1. trọng số vào	3,4,5	4
	2. cái ra	4,5,7	4
	3. câu hỏi	3,4,6	6
	4. tệp	7,10,15	10
	5. giao diện	5,7,10	5

Xử lý nhân tố phức tạp			
	Nhân tố được đánh tỉ lệ từ 0 đến 5 với: không ảnh hưởng (0); tình cờ (1); vừa phải (2); trung bình (3); có ý nghĩa (4); bản chất (5).		
	1. sao lưu và khôi phục cần có	0,1,2,3,4,5	4
	2. truyền thông dữ liệu cần có	0,1,2,3,4,5	1
	3. chức năng xử lý phân bố	0,1,2,3,4,5	0
	4. độ căng hoàn thiện	0,1,2,3,4,5	3
	5. môi trường vận hành dùng nhiều	0,1,2,3,4,5	3
	6. vào dữ liệu trực tuyến	0,1,2,3,4,5	5
	7. giao tác với nhiều màn hình	0,1,2,3,4,5	4
	8. tệp chính được cập nhật trực tuyến	0,1,2,3,4,5	4
	9. vào, ra, tệp, hỏi phức tạp	0,1,2,3,4,5	3
	10. xử lý bên trong phức tạp	0,1,2,3,4,5	3
	11. mã được thiết kế để dùng lại	0,1,2,3,4,5	2
	12. thiết kế có cả chuyển đổi/cài đặt	0,1,2,3,4,5	2
	13. thiết kế hệ thống cho nhiều cài đặt	0,1,2,3,4,5	4
	14. bảo trì/dễ dùng	0,1,2,3,4,5	5
ĐỘ ĐO HƯỚNG KÍCH CỠ			
Mô tả		Đơn vị	Dữ liệu mẫu
Độ đo hiệu năng và chi phí			
	Tên dự án	Alphanumeric	Proj#1
	Cái ra	KLOC/ng-th	0.905
	Tổng chi phí mã được bảo trì	\$/KLOC	\$22514
	Chi phí không tính mã dùng lại	\$/KLOC	\$24028
	Thời gian cần thực hiện	Tháng/KLOC	1.0

	Tư liệu	Trang/KLOC	30
	Tư liệu	Trang/ng-th	10
	Tư liệu	\$/Trang	\$739
Độ đo chất lượng			
	Khiếm khuyết	lỗi/KLOC	2.0
	Chi phí lỗi	\$/lỗi	\$376
	Bảo trì lỗi/Tổng bảo trì	tỉ lệ	0.36
	Bảo trì sửa đổi/Tổng bảo trì	tỉ lệ	0.64
	Bảo trì công sức/Tổng bảo trì	tỉ lệ	1015
	Lưu ý: độ đo hướng kích cỡ được tính như trên dùng “KLOC được bảo trì”, tức là chỉ tính số dòng chương trình mới phát triển, sửa đổi và làm mới.		
ĐỘ ĐO HƯỚNG CHỨC NĂNG			
Mô tả		Đơn vị	Dữ liệu mẫu
Tính toán điểm chức năng			
	điểm chức năng không điều chỉnh		378
	mức ảnh hưởng toàn bộ		43
	điều chỉnh độ phức tạp		1.08
	điểm chức năng (FP)		408
Độ đo hiệu năng và chi phí FP			
	tên dự án	alphanumeric	Proj#1
	cái ra	FP/ng-th	11.1
	chi phí	\$/FP	\$700
	thời gian tồn	FP/tháng	31.4
	tư liệu	trang/FP	0.9
Chức năng			

	kích cỡ chương trình	FP/chương trình	408
	chức năng theo kích cỡ	FP/KLOC-bảo trì	32
Độ đo chất lượng			
	khiếm khuyết	lỗi/FP	0.064
	bảo trì lỗi – công sức	người – ngày/FP	0.817
	bảo trì lỗi - sửa đổi	người – ngày/FP	1.427

Tóm tắt

Việc đo làm cho các nhà quản lý và người phát triển hiểu rõ hơn về tiến trình kỹ nghệ phần mềm và sản phẩm phần mềm mà nó tạo ra. Có thể xác định việc dùng các cách đo trực tiếp và gián tiếp, các cách đo hiệu năng và chất lượng.

Cả hai độ đo theo kích cỡ và chức năng đều được dùng. Độ đo theo kích cỡ dùng số dòng mã làm nhân tố chuẩn cho các cách đo khác như người-tháng hay khiếm khuyết. Độ đo điểm chức năng được suy dẫn từ các cách đo lĩnh vực thông tin và một xác nhân chủ quan về độ phức tạp vấn đề.

Độ đo chất lượng phần mềm, giống như độ đo hiệu năng, tập trung và cả tiến trình và sản phẩm. Bằng cách phát triển và phân tích vạch ranh giới độ đo chất lượng, một tổ chức có thể hành động để sửa chữa lại các yếu tố của tiến trình kỹ nghệ phần mềm vốn là nguyên nhân cho các khiếm khuyết phần mềm. Chúng ta đã thảo luận về độ đo chất lượng – tính đúng đắn, tính bảo trì, tính toàn vẹn và tính dùng lại.

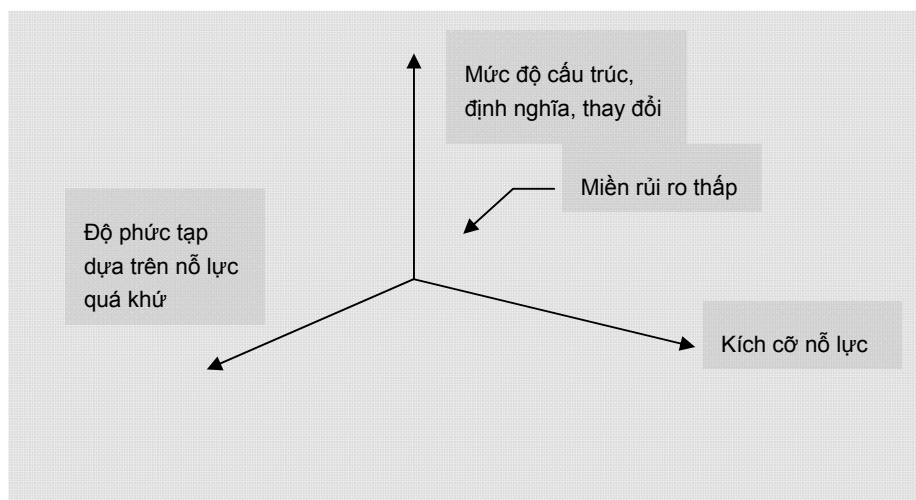
Việc đo tạo ra sự thay đổi về văn hoá. Việc thu thập dữ liệu, tính toán độ đo và đánh giá dữ liệu là ba bước cần phải thực hiện để bắt đầu một chương trình độ đo. Bằng cách tạo ra vạch ranh giới độ đo - một cơ sở dữ liệu có chứa cách đo tiến trình và sản phẩm – các kỹ sư phần mềm và người quản lý của họ có thể có được cái nhìn sâu hơn vào công việc của họ đang làm và sản phẩm của họ tạo ra.

CHƯƠNG III**ƯỚC LƯỢNG DỰ ÁN PHẦN MỀM**

Tiến trình quản lý dự án kỹ nghệ phần mềm bắt đầu với một tập các hoạt động gọi chung là *lập kế hoạch dự án*. Hoạt động đầu tiên trong những hoạt động này là ước lượng.

III.1 Ước lượng

Đặc trưng quan trọng nhất cho một người quản lý dự án là "một người có khả năng biết được cái gì sẽ không ổn trước khi nó thực sự xảy ra". Việc ước lượng tài nguyên, chi phí, và lập lịch cho nỗ lực phát triển phần mềm đòi hỏi phải có kinh nghiệm, thâm nhập thông tin lịch sử tốt và dũng cảm sử dụng các cách đo định lượng khi đã có mọi dữ liệu định lượng. Ước lượng mang sẵn trong nó yếu tố rủi ro và những nhân tố làm tăng sự rủi ro như được minh họa trong Hình 3.1.

**H3.1 Ước lượng và rủi ro**

Các trục được vẽ trong hình biểu thị cho các đặc trưng của dự án cần được ước lượng.

Độ phức tạp dự án có ảnh hưởng mạnh lên tính bất trắc cố hữu có trong việc lập kế hoạch, là cách đo bị ảnh hưởng bởi sự quen thuộc với nỗ lực quá khứ. Các cách đo này được áp dụng ở mức thiết kế hay mã hoá và do đó khó dùng được cho việc lập kế hoạch phần mềm (trước khi thiết kế và mã hoá tồn tại). Tuy nhiên người ta lại có thể thiết lập các định giá chủ quan về độ phức tạp (như các nhân tố điều chỉnh độ phức tạp điểm chức năng) ngay từ đầu trong tiến trình kỹ nghệ phần mềm.

Kích cỡ dự án là một nhân tố quan trọng khác có thể ảnh hưởng tới độ chính xác và tính hiệu quả của ước lượng. Khi kích cỡ tăng lên, sự phụ thuộc lẫn nhau giữa nhiều phần tử của phần mềm cũng tăng lên nhanh chóng. Việc phân rã bài toán có thể thành khó khăn hơn bởi vì các phần tử được phân rã còn quá lớn.

Mức độ cấu trúc của dự án cũng có ảnh hưởng lên việc ước lượng rủi ro. Cấu trúc nói tới sự dễ dàng mà các chức năng có thể được phân nhóm và bản chất thứ bậc của thông tin phải được xử lý. Mức độ cấu trúc tăng lên nên khả năng ước lượng chính xác cũng được cải thiện và rủi ro giảm đi.

Việc có sẵn thông tin lịch sử cũng xác định ra rủi ro ước lượng. Bằng cách nhìn lại quá khứ chúng ta có thể mô phỏng mọi việc đã xảy ra và cải thiện các yếu tố kỹ nghệ có vấn đề nảy sinh. Khi các độ đo phần mềm đã có sẵn cho các dự án quá khứ thì có thể tiến hành với các ước lượng với sự đảm bảo lớn hơn; có thể thiết lập việc lập lịch để tránh vấp phải những khó khăn trong quá khứ và toàn bộ rủi ro có thể được thu lại.

Rủi ro được đo bằng mức độ bất trắc theo ước lượng định lượng được thiết lập cho tài nguyên, chi phí và lập lịch. Nếu phạm vi dự án còn chưa được hiểu rõ hay các yêu cầu dự án còn là chủ đề thay đổi thì sự bất trắc và rủi ro trở nên càng nguy hiểm. Người lập kế hoạch phần mềm phải có đầy đủ chức năng, hiệu suất và định nghĩa giao diện (trong đặc tả hệ thống) và quan trọng hơn, khách hàng phải thừa nhận rằng sự thay đổi trong các yêu cầu phần mềm có nghĩa là có sự không ổn định trong chi phí lịch biểu.

III.2 Mục tiêu lập kế hoạch dự án

Người quản lý dự án phần mềm phải đương đầu với một vấn đề khó khăn ngay lúc bắt đầu phát triển: Cần có ngay các ước lượng định lượng nhưng lại không có thông tin vững chắc. Một phân tích chi tiết về các yêu cầu phần mềm sẽ đưa ra thông tin cần thiết cho các ước lượng, nhưng việc phân tích thường việc phân tích thường mất vài tuần hay vài tháng mới hoàn tất.

Mục tiêu của việc lập dự án phần mềm là cung cấp một khuôn khổ cho phép nhà quản lý lập ra các ước lượng định lượng hợp lý về tài nguyên, chi phí và lịch biểu. Các ước lượng này được tiến hành trong một khoảng thời gian giới hạn lúc bắt đầu dự án phần mềm, và được cập nhật đều đặn trong tiến trình dự án.

Mục tiêu của lập kế hoạch dự án đạt được thông qua một tiến trình phát hiện thông tin để dẫn đến các ước lượng hợp lý. Sau đây là các hoạt động liên quan tới việc lập kế hoạch dự án phần mềm.

III.3 Xác định phạm vi phần mềm

Hoạt động đầu tiên trong lập kế hoạch dự án là xác định phạm vi phần mềm. Chức năng và hiệu suất của phần mềm cần phải đạt tới trong kỹ nghệ hệ thống máy tính cần phải được định giá để thiết ra phạm vi dự án một cách dễ hiểu đối với các cấp lãnh đạo và kỹ thuật. Dữ liệu định lượng cần được thiết lập tường minh, các ràng buộc/hoặc giới hạn như chi phí sản phẩm hay kích cỡ lên bộ nhớ... cần được ghi lại và các nhân tố hỗ trợ (như các thuật toán có sẵn...) được mô tả.

Phạm vi phần mềm mô tả chức năng, hiệu suất, các ràng buộc, giao diện và độ tin cậy. Chức năng được mô tả trong phát biểu về phạm vi sẽ được đánh giá, làm mịn để đưa ra các mức ưu tiên chi tiết hơn cho việc ước lượng. Cả ước lượng chi phí và lịch biểu đều hướng theo chức năng nên có ích hơn cả là có được một mức độ phân rã nào đó. Các xem xét về hiệu suất bao gồm các yêu cầu về tiến trình và thời gian đáp ứng. Ràng buộc xác định ra các ranh giới áp đặt lên phần mềm bởi phần cứng bên ngoài, bộ nhớ có sẵn hay các hệ thống hiện đang tồn tại khác.

Chức năng, hiệu suất, và ràng buộc phải được đánh giá cùng nhau. Cùng một chức năng có thể tạo ra sự khác biệt lớn trong nỗ lực phát triển khi được xem xét trong hoàn cảnh các giới hạn hoàn thiện khác nhau. Công sức, chi phí cần cho việc phát triển phần mềm sẽ khác biệt rất lớn nếu chức năng phức tạp lên - do phải đòi hỏi thêm

công sức phát triển. Chức năng, hiệu suất và ràng buộc có quan hệ mật thiết đến nhau.

Phần mềm tương tác với các phần tử khác trong hệ thống dựa trên máy tính. Người lập kế hoạch xem xét bản chất và độ phức tạp của từng giao diện để xác định ảnh hưởng lên tài nguyên, chi phí và lịch biểu của việc phát triển. Khái niệm về giao diện được hiểu là:

- (1) Phần cứng
- (2) Phần mềm đã có (bộ chương trình con, hệ điều hành...)
- (3) Con người sử dụng phần mềm thông qua thiết bị cuối hay thiết bị đầu vào/ đầu ra
- (4) Các thủ tục thực hiện trước hay sau phần mềm xem như chuỗi tuần tự các thao tác.

Trong mỗi trường hợp thông tin truyền qua các giao diện phải hoàn toàn được hiểu rõ.

Khía cạnh ít chính xác nhất của phạm vi phần mềm là độ tin cậy. Việc đo độ tin cậy phần mềm còn ít được dùng tại giai đoạn này của dự án. Các đặc trưng độ tin cậy phần cứng cổ điển như thời gian trung bình giữa hai lần hỏng (MTBF) rất khó chuyển sang lĩnh vực phần mềm. Tuy nhiên, bản chất chung của phần mềm có thể chỉ đạo các xem xét đặc biệt để đảm bảo độ tin cậy của nó. Chẳng hạn, phần mềm cho hệ thống kiểm soát không lưu phải không được hỏng hóc nếu không tính mạng con người có thể bị nguy hiểm. Một hệ thống kiểm soát kho hay phần mềm xử lý văn bản không nên hỏng nhưng ảnh hưởng hỏng hóc thì ít nguy hiểm cho con người hơn. Mặc dầu không thể có định lượng độ tin cậy phần mềm chính xác như ta muốn trong phát biểu về phạm vi, chúng ta có thể dùng bản chất của dự án để giúp việc hình thành các ước lượng về công sức và chi phí để đảm bảo độ tin cậy.

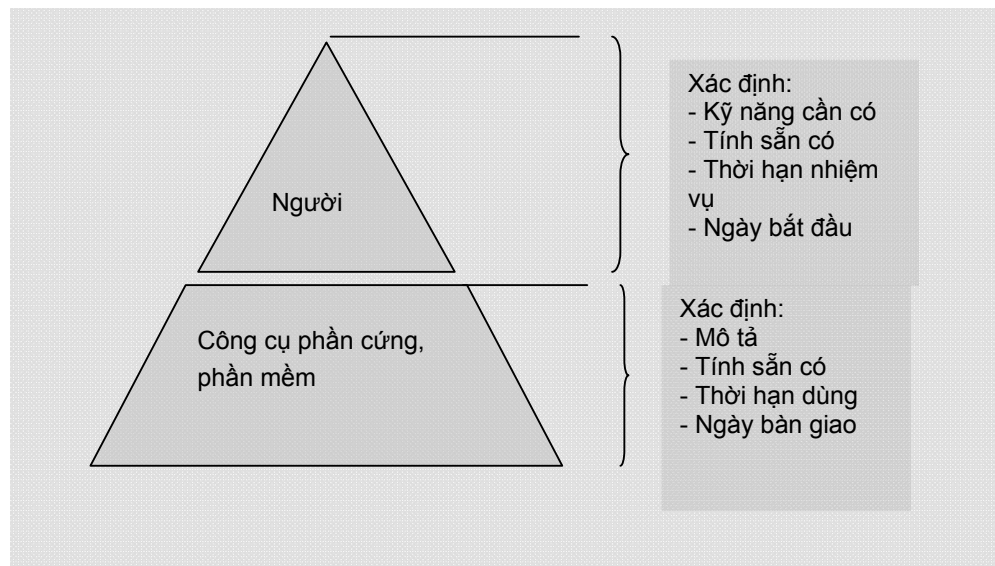
Nếu một Bản Đặc tả hệ thống đã được phát triển đúng đắn thì gần như tất cả thông tin cần cho một mô tả về phạm vi phần mềm đều có sẵn và được làm tư liệu trước khi việc lập dự án bắt đầu. Trong trường hợp còn chưa xây dựng được đặc tả thì

người lập kế hoạch phải đóng vai trò của nhà phân tích hệ thống để xác định các thuộc tính và giới hạn sẽ ảnh hưởng tới nhiệm vụ ước lượng.

III.4 Ước lượng về tài nguyên

Nhiệm vụ thứ hai của lập kế hoạch phần mềm là ước lượng về tài nguyên chi việc hoàn thành nỗ lực phát triển phần mềm. Hình 3.2 minh họa cho tài nguyên phát triển dưới dạng hình tháp.

Ở đáy là các công cụ - phần cứng và phần mềm - phải có để hỗ trợ cho nỗ lực phát triển. Ở mức cao hơn tài nguyên chính là con người - yếu tố bao giờ cũng cần tới. Mỗi tài nguyên đều được xác định với bốn thuộc tính: mô tả tài nguyên, phát biểu về tính có sẵn, thời gian hạn định cần có tài nguyên, thời hạn tài nguyên cần được sử dụng. Hai đặc trưng cuối có thể xem như cửa sổ thời gian. Tính sẵn có của tài nguyên đối với một cửa sổ phải được thiết lập ở giai đoạn thực hành sớm nhất.



H 3.2 Tài nguyên

III.4.1 Tài nguyên con người

Người lập kế hoạch bắt đầu bằng việc ước lượng phạm vi và lựa chọn kỹ năng cần để hoàn thành việc phát triển. Cả vị trí trong tổ chức (như người quản lý, kỹ sư phần mềm cấp cao...) và tính chuyên môn (như viễn thông, cơ sở dữ liệu, bộ vi xử lý) đều phải xác định rõ. Với những dự án tương đối nhỏ (1 người-năm hay ít hơn) riêng

một người có thể thực hiện tất cả các bước kỹ nghệ phần mềm, với việc tư vấn các chuyên gia cần thiết.

Số người cần cho dự án phần mềm có thể được xác định chỉ sau khi đã xây dựng được ước lượng nỗ lực phát triển (như người-tháng hay người-năm).

III.4.2 Tài nguyên phần cứng

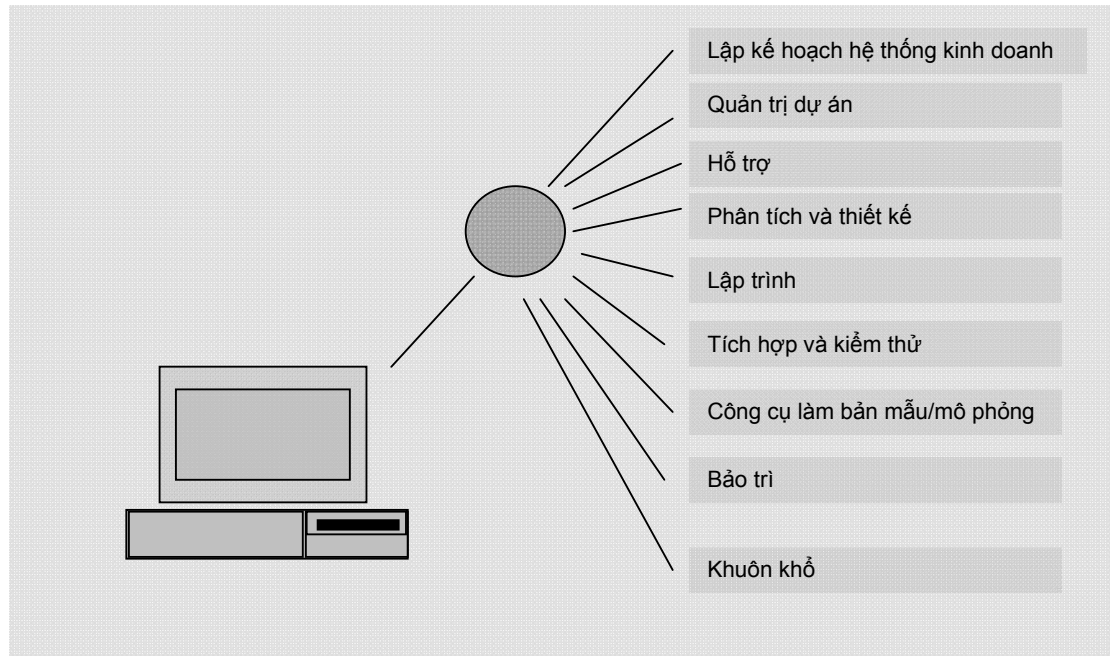
Ngay từ đầu, chúng ta đã tham khảo tới phần cứng như là tiềm năng tính toán. Bên trong vấn đề tài nguyên, phần cứng cũng là một công cụ cho việc phát triển phần mềm. Có ba loại phần cứng cần phải được xem xét tới trong việc lập kế hoạch dự án phần mềm: hệ thống phát triển, máy đích và các phần tử khác của hệ thống mới.

Hệ thống phát triển (cũng được gọi là hệ thống chủ) là một máy tính và các thiết bị ngoại vi có liên quan sẽ được dùng tới trong khi phát triển phần mềm. Chẳng hạn, máy tính 32-bit có thể phục vụ như hệ thống phát triển cho một bộ vi xử lý 16-bit máy đích – trên đó phần mềm cuối cùng sẽ được thực hiện. Hệ phát triển được dùng bởi vì nó có thể hỗ trợ cho nhiều người dùng, duy trì khối lượng thông tin lớn có thể dùng chung cho nhóm phát triển phần mềm, và hỗ trợ cho sự phân loại phong phú về công cụ phần mềm (sẽ được thảo luận trong mục tiếp).

Vì phần lớn các tổ chức phát triển đều có nhiều vị trí đòi hỏi thâm nhập vào hệ thống phát triển nên người lập kế hoạch phải mô tả cẩn thận cửa sổ thời gian cần thiết và kiểm chứng rằng tài nguyên có sẵn.

III.4.3 Tài nguyên phần mềm

Như ta dùng phần cứng làm công cụ để xây dựng phần cứng mới, chúng ta dùng phần mềm để hỗ trợ cho việc phát triển phần mềm mới. Bộ dịch hợp ngữ nguyên thủy đã được viết trong ngôn ngữ máy và được dùng để phát triển nhiều trình hợp dịch phức tạp. Xây dựng trên khả năng của các phần mềm phiên bản trước, người phát triển cuối cùng được chuyển sang các trình biên dịch ngôn ngữ cấp cao và các công cụ khác.



H3.3 CASE

Ngày nay, các kỹ sư kỹ nghệ phần mềm dùng một tập công cụ tương tự như các công cụ thiết kế với sự trợ giúp của máy tính và kỹ nghệ với trợ giúp của máy tính (CAD/CAE) dùng cho các kỹ sư phần cứng. Tập công cụ, còn được gọi là kỹ nghệ phần mềm với sự trợ giúp của máy tính (CASE) được vẽ trong H3.3.

Tổng quan về các loại công cụ chính:

Công cụ lập kế hoạch hệ thống nghiệp vụ: Bằng cách mô hình hoá các yêu cầu thông tin chiến lược của một tổ chức, các công cụ lập kế hoạch hệ thống nghiệp vụ cung cấp một siêu mô hình để dẫn ra các hệ thống tin chuyên dụng. Các công cụ này trả lời cho một số câu hỏi đơn giản nhưng quan trọng:

- Dữ liệu chủ chốt cho nghiệp vụ xuất phát từ đâu?
- Thông tin đó đi đến đâu?
- Nó được sử dụng như thế nào?
- Nó được biến đổi như thế nào khi chuyển qua nghiệp vụ?

- Thông tin mới nào được thêm vào?

Các công cụ lập kế hoạch hệ thống: Giúp cho người phát triển phần mềm tạo ra hệ thống tin vận chuyển dữ liệu tới những người cần thông tin và kiểm tra lại các nhân viên với các thông tin phụ thêm. Trong việc phân tích cuối cùng, việc chuyển dữ liệu được cải tiến và việc ra quyết định được trù liệu.

Công cụ quản lý dự án: Bằng cách dùng cách dùng các công cụ quản lý này, một nhà quản lý dự án có thể sinh ra các ước lượng có ích về công sức, chi phí, và thời hạn của dự án phần mềm; xác định cấu trúc phân chia công việc (WBS); lập kế hoạch lịch biểu dự án khả thi; và theo dõi dự án trên cơ sở liên tục. Bên cạnh đó, nhà quản lý có thể dùng các công cụ thu thập độ đo để thiết lập một vạch ranh giới cho hiệu năng phát triển phần mềm và chất lượng sản phẩm.

Công cụ hỗ trợ: Các loại công cụ hỗ trợ bao gồm công cụ xây dựng tư liệu, phần mềm hệ thống mạng, cơ sở dữ liệu, thư điện tử, bản tin thường ngày, các công cụ quản lý cấu hình được dùng để kiểm soát và quản lý thông tin được tạo ra khi phần mềm được phát triển.

Công cụ phân tích và thiết kế: Công cụ phân tích và thiết kế làm cho kỹ sư phần mềm tạo ra được mô hình về hệ thống cần được xây dựng. Các công cụ này trợ giúp cho việc tạo ra mô hình và cũng giúp cho việc đánh giá chất lượng của mô hình. Bằng cách thực hiện kiểm tra tính nhất quán và hợp lệ trên mỗi mô hình, các công cụ phân tích và thiết kế cung cấp cho kỹ sư phần mềm cái nhìn sâu sắc và giúp khử bỏ lỗi trước khi chúng lan truyền trong chương trình.

Công cụ lập trình: Các trình tiện ích phần mềm hệ thống, trình soạn thảo, trình biên dịch và trình gỡ lỗi là thành phần hợp pháp của CASE. Nhưng bên cạnh những công cụ này, có thể bổ sung thêm các công cụ lập trình mới và mạnh hơn. Các công cụ lập trình hướng sự vật, các ngôn ngữ lập trình thể hệ bốn, các hệ thống hỏi cơ sở dữ liệu và một phạm vi rộng các công cụ PC (trang tính) tất cả đều nằm trong phạm trù công cụ CASE này.

Công cụ tích hợp và kiểm thử: Công cụ kiểm thử cung cấp một sự phong phú về mức độ hỗ trợ cho các bước kiểm thử phần mềm được áp dụng như một phần tiến trình kỹ nghệ phần mềm. Một số công cụ, như bộ phân tích bao quát đường dẫn, cung

cấp sự hỗ trợ trực tiếp cho việc thiết kế các trường hợp thử và được dùng trong các giai đoạn đầu của việc kiểm thử. Các công cụ khác, như kiểm thử quy hồi tự động và công cụ sinh phép thử, được dùng trong việc kiểm thử tích hợp và hợp lệ và có thể giúp thu nhỏ khối lượng nỗ lực được áp dụng cho tiến trình kiểm thử.

Công cụ làm bản mẫu và mô phỏng: Các công cụ làm bản mẫu và mô phỏng mở rộng tập công cụ tinh vi bao quát từ bộ soạn thảo màn hình đơn giản tới các sản phẩm mô phỏng cho việc phân tích thời gian và kích cỡ của các hệ thống nhúng thời gian thực. Tại mức cơ sở nhất, các công cụ làm bản mẫu tập trung vào việc tạo ra màn hình và báo cáo để làm cho người dùng hiểu được cái vào/cái ra của hệ thống tin hay ứng dụng kỹ nghệ. Tại mức tinh vi nhất, các công cụ mô phỏng được dùng để tạo ra các mô hình hệ thống cho những ứng dụng thời gian thực được nhúng (như mô hình hoá hệ thống kiểm soát tiến trình cho hệ thống lọc hay điện tử hàng không cho máy bay). Các mô hình được tạo ra bởi công cụ mô phỏng như vậy có thể được phân tích, trong một số trường hợp, có thể được thực hiện người ta có thể đánh giá được hiệu suất khi chạy của hệ thống trước khi hệ thống được xây dựng.

Công cụ bảo trì: Công cụ bảo trì có thể giúp việc phân rã một chương trình hiện có và cung cấp cho kỹ sư cái nhìn thông suốt, tuy nhiên, người kỹ sư phải dùng trực giác, cảm giác thiết kế, và trí thông minh con người để hoàn thiện tiến trình đảo kỹ nghệ hoặc tái kỹ nghệ cho ứng dụng. Thành phần con người là một phần tích hợp của công cụ kỹ nghệ đảo và tái kỹ nghệ và không thể nào bị thay thế bởi việc tự động hoá hoàn toàn trong tương lai có thể thấy trước được.

Công cụ khuôn khổ: Các công cụ này đưa ra một khuôn khổ từ môi trường hỗ trợ dự án tích hợp (IPSE) có thể được tạo ra. Trong phần lớn các trường hợp các công cụ khuôn khổ cung cấp khả năng quản lý cơ sở dữ liệu và quản lý cấu hình cùng với các tiện ích kèm theo công cụ này do các nhà cung cấp khác nhau, được tích hợp vào IPSE.

III.4.4 Tính sử dụng lại

Bất kỳ một thảo luận nào về tài nguyên phần mềm cũng đều sẽ không đầy đủ nếu không thừa nhận về *tính sử dụng lại* - tức là việc tạo ra và dùng lại các khối xây dựng phần mềm. Các khối xây dựng như vậy phải được lên danh mục cho dễ tham

khảo, phải được chuẩn hoá cho dễ ứng dụng và phải được làm hợp lệ cho việc tích hợp dễ dàng. Các khối phần mềm sẽ có sẵn để cho phép việc xây dựng các chương trình lớn với việc phát triển từ 'tay trắng' ở mức tối thiểu. Thư viện các phần mềm tái dụng đã có cho các ứng dụng đã có cho các ứng dụng thương mại, các hệ thống và công việc thời gian thực, các vấn đề khoa học và công nghệ. Tuy nhiên còn ít có các kỹ thuật hệ thống để bổ sung cho một thư viện, các giao diện chuẩn cho phần mềm dùng lại còn khó tăng cường, vấn đề chất lượng và bảo trì cần còn chưa được giải quyết, và điều cuối cùng, người phát triển thường không biết rằng các khối xây dựng phần mềm đã có sẵn.

Người lập kế hoạch phần mềm nên xem xét tới hai quy tắc khi phần mềm dùng lại được xác định như một tài nguyên:

- Nếu phần mềm hiện có đáp ứng được yêu cầu thì hãy dùng nó. Chi phí để có được phần mềm hiện có gần như bao giờ cũng ít hơn chi phí để phát triển một phần mềm tương đương.
- Nếu phần mềm hiện có đòi hỏi sửa đổi nào đó trước khi nó được tích hợp đúng đắn với hệ thống thì hãy xử lý cho cẩn thận. Chi phí để sửa đổi phần mềm hiện có đôi khi có thể còn lớn hơn chi phí phát triển phần mềm tương đương.

Trong thực tế, các tài nguyên phần mềm lại thường hay bị bỏ qua trong khi lập kế hoạch, chỉ trở thành mối quan tâm trong giai đoạn phát triển của tiến trình kỹ nghệ phần mềm. Ta nên xác định yêu cầu tài nguyên phần mềm sớm, nhờ đó, việc đánh giá kỹ thuật cho các phương án có thể được tiến hành và thu được đúng hạn.

III.5 Ước lượng dự án phần mềm

Trong những ngày đầu của tin học, chi phí phần mềm bao gồm một phần trăm nhỏ của toàn bộ chi phí cho hệ thống dựa trên máy tính. Một lỗi lầm lớn trong ước lượng chi phí phần mềm có thể tương đối ít ảnh hưởng. Ngày nay, phần mềm là yếu tố tốn kém nhất trong nhiều hệ thống dựa trên máy tính. Lỗi lầm ước lượng chi phí lớn có thể tạo ra sự chênh lệch giữa lợi nhuận và thất thoát. Việc bội chi phí có thể là thảm hoạ cho người phát triển.

Ước lượng chi phí và công sức phần mềm chắc chắn không là một khoa học chính xác. Có quá nhiều tham biến – con người, kỹ thuật, môi trường, chính trị - có thể ảnh hưởng tới chi phí chung cuộc của phần mềm và công sức cần để phát triển nó.

Để đạt được các ước lượng chi phí và công sức tin cậy, một tùy chọn nảy sinh:

- Trì hoãn việc ước lượng tới giai đoạn sau của dự án (hiển nhiên chúng ta có thể đạt được ước lượng chính xác 100% khi dự án đã hoàn tất)
- Dùng các **kỹ thuật phân rã** tương đối đơn giản để sinh ra ước lượng về chi phí và công sức dự án
- Phát triển **mô hình kinh nghiệm** cho chi phí và công sức làm phần mềm
- Thu được một hay nhiều **công cụ ước lượng tự động**

Tùy chọn đầu tiên mặc dầu rất ‘hấp dẫn’ nhưng lại không thực tế. Các ước lượng chi phí phải đưa ra ngay từ đầu. Tuy nhiên chúng ta đợi càng lâu thì chúng ta biết càng nhiều, và biết càng nhiều thì càng ít có khả năng phạm lỗi lầm trong ước lượng.

Ba tùy chọn còn lại là những cách tiếp cận có thể đứng được tương đối đối với ước lượng dự án phần mềm. Một cách lý tưởng, các kỹ thuật được lưu ý cho từng tùy chọn nên được áp dụng nối tiếp nhau, mỗi tùy chọn được dùng như một phép kiểm tra chéo cho truy chọn khác. Các *kỹ thuật phân rã* lấy quan điểm ‘chia để trị’ cho việc ước lượng dự án dự án phần mềm. Bằng cách phân rã một dự án thành các chức năng chính và các nhiệm vụ kỹ nghệ phần mềm có liên quan, việc ước lượng chi phí và nỗ lực có thể được thực hiện theo kiểu từng bước. Các *mô hình ước lượng theo kinh nghiệm* có thể được dùng để bổ sung cho các kỹ thuật phân rã và đưa ra một cách tiếp cận ước lượng tiềm năng có giá trị theo đúng nghĩa của nó. Một mô hình dựa trên kinh nghiệm (dữ liệu lịch sử) có dạng:

$$d = f(v_i)$$

d : là một trong một số các giá trị ước lượng (như công sức, chi phí, thời hạn dự án..)

vì: các tham biến độc lập được chọn lựa (LOC hoặc FP được ước lượng)

Có các công cụ ước lượng tự động cài đặt cho một hay nhiều kỹ thuật phân rã (hay mô hình kinh nghiệm). Khi được tổ hợp với một giao diện người máy tương tác, các công cụ tự động hoá đưa ra một tùy chọn cho việc ước lượng. Trong một hệ thống như vậy, các đặc trưng của tổ chức phát triển (như kinh nghiệm, môi trường...) và phần mềm cần phát triển sẽ được mô tả. Các ước lượng chi phí và nỗ lực được suy ra từ dữ liệu này.

III.6 Kỹ thuật phân rã

Con người đã phát triển cách tiếp cận tự nhiên đến việc giải quyết vấn đề: nếu vấn đề cần giải quyết quá phức tạp thì ta có khuynh hướng chia nhỏ nó ra tới khi gặp được các vấn đề có thể giải quyết được, sau đó chúng ta giải quyết riêng rẽ từng vấn đề nhỏ ấy với hy vọng có thể tổ hợp các giải pháp thành giải pháp tổng thể.

Việc ước lượng dự án phần mềm là một dạng giải quyết vấn đề, trong hầu hết các trường hợp vấn đề cần giải quyết là quá phức tạp không thể xem xét được trong một đơn vị (cụ thể ở đây là việc phát triển một ước lượng về chi phí, công sức cho một dự án phần mềm).

III.6.1 Ước lượng LOC và FP

Ở trên chúng ta đã mô tả dòng mã (LOC) và điểm chức năng (FP) như những dữ liệu cơ sở để từ đó tính ra độ đo hiệu năng. Dữ liệu LOC và FP được dùng theo hai cách trong việc ước lượng dự án phần mềm như:

- (1) Các biến ước lượng được dùng để lấy “kích cỡ” cho từng phần tử phần mềm
- (2) Độ đo vạch ranh giới được thu thập từ những dự án quá khứ và được dùng chung với các biến ước lượng để xây dựng các dự tính chi phí và công sức.

Ước lượng LOC và FP là các kỹ thuật ước lượng phân biệt, nhưng cả hai đều có một số đặc trưng chung. Người lập dự án bắt đầu với một tuyên bố về phạm vi phần mềm, từ đó, dự định phân rã phần mềm thành các chức năng nhỏ hơn có thể được áp dụng riêng biệt. LOC và FP sẽ được ước lượng cho từng chức năng con đó. Độ đo tính hiệu năng vạch ranh giới (như LOC/người-tháng hay FP/người-tháng) sau đó

được áp dụng cho biến ước lượng thích hợp và ước lượng về chi phí và công sức từ đó được suy dẫn ra. Các ước lượng chức năng con được tổ hợp lại để tạo ra một thiết kế ước lượng tổng thể cho toàn bộ dự án.

Các ước lượng kỹ thuật LOC và FP khác nhau ở mức độ chi tiết cần cho việc phân rã. Khi LOC được dùng như biến ước lượng thì việc phân rã chức năng thường được thực hiện cho các mức chi tiết đáng kể. Vì dữ liệu cần cho việc ước lượng điểm chức năng ở mức vĩ mô hơn, nên mức độ phân rã được dùng khi FP là biến ước lượng sẽ ít chi tiết hơn. Cũng cần lưu ý rằng LOC được ước lượng trực tiếp trong khi FP được xác định gián tiếp bằng cách ước lượng số cái vào, cái ra, tệp dữ liệu, câu hỏi, và giao diện ngoài kèm theo 14 giá trị điều chỉnh độ phức tạp sau:

F_i:

1. Hệ thống có đòi hỏi sao lưu và khôi phục đáng tin cậy 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 môi trường hiện có mà nặng về vận hành tiện ích 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 thao tác đưa vào thông qua nhiều màn hình thao tác không?
8. Tệp chính có phải cập nhật trực tuyến 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 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 không?

Bất kể là biến ước lượng nào được dùng, người lập kế hoạch dự án về cơ bản phải đưa ra một phạm vi các giá trị cho từng chức năng đã phân rã. Bằng cách dùng dữ liệu lịch sử hay trực giác, người lập kế hoạch ước lượng một giá trị LOC hay FP lạc quan có thể nhất và bi quan cho từng chức năng.

Giá trị kỳ vọng cho LOC hay FP sau đó được tính. Giá trị kỳ vọng cho biến ước lượng, E, có thể được tính như trung bình trọng số của ước lượng LOC hay FP lạc quan (a), có thể nhất (m) và bi quan (b).

Ví dụ:

$$E = \frac{a + 4m + b}{6}$$

cho niềm tin đối với ước lượng có thể nhất và tiếp đó là phân bố xác suất beta.

Chúng ta giả sử rằng có một xác suất rất nhỏ là kết quả của LOC hay FP thực tại sẽ ra ngoài các giá trị và bi quan. Bằng cách áp dụng các kỹ thuật thống kê chuẩn chúng ta có thể tính độ lệch của ước lượng, tuy nhiên, độ lệch dựa trên một dữ liệu không chắc chắn (ước lượng) phải được dùng một cách thận trọng.

Một khi giá trị kỳ vọng của biến ước lượng đã được xác định thì ta áp dụng dữ liệu hiệu năng LOC và FP. Tại giai đoạn này, người lập kế hoạch áp dụng một trong hai cách tiếp cận khác nhau:

1. Giá trị biến ước lượng toàn bộ cho tất cả các chức năng con có thể được nhân lên với độ đo hiệu năng trung bình tương ứng với biến ước lượng đó. Chẳng hạn, nếu ta giả sử rằng đã ước lượng 310 FP về tổng số và rằng hiệu năng trung bình FP dựa trên các dự án quá khứ 5.5 FP/người-tháng thì tổng công sức cho dự án sẽ là:

$$\text{Công sức} = \frac{310}{5.5} = 56 \text{ người-tháng}$$

2. Giá trị biến ước lượng cho từng chức năng con có thể được nhân lên với giá trị hiệu năng được điều chỉnh dựa trên một mức độ nhận biết được về độ phức tạp của các chức năng con. Đối với các chức năng có độ phức tạp trung bình người ta dùng độ đo hiệu năng trung bình. Tuy nhiên nó được điều chỉnh lên hay xuống dựa trên việc cao hơn hay thấp hơn độ phức tạp trung bình đối với chức năng con đặc biệt. Ví dụ, nếu độ phức tạp trung bình là 490 LOC/người-tháng, chức năng con phức tạp hơn đáng kể so với độ trung bình đó, có thể phản ánh một hiệu năng được ước lượng chỉ vào cỡ 300 LOC/người-tháng và chức năng đơn giản là 650 LOC/người-tháng.

Các ước lượng này có đúng không? Câu trả lời duy nhất cho câu hỏi này là: Chúng ta không thể chắc chắn được. Mọi kỹ thuật ước lượng, dù phức tạp đến đâu cũng phải được kiểm tra chéo bằng các tiếp cận khác. Mặt khác, cảm giác thông thường và kinh nghiệm vẫn chiếm ưu thế.

III.6.2 Ước lượng công sức

Ước lượng công sức là kỹ thuật thông thường nhất cho việc xác định chi phí cho bất kỳ dự án phát triển công nghệ nào. Số người – tháng, tháng hay năm được áp dụng cho giải pháp của từng nhiệm vụ dự án. Chi phí theo đồng được gắn với từng đơn vị công sức và suy ra được chi phí ước lượng.

Giống như kỹ thuật LOC và FP, ước lượng công sức bắt đầu với việc phân tích các chức năng phần mềm thu được từ phạm vi dự án. Một loạt các nhiệm vụ kỹ nghệ phần mềm - phân tích yêu cầu, thiết kế, mã hoá và kiểm thử - phải thực hiện cho từng chức năng. Chức năng và các nhiệm vụ kỹ nghệ phần mềm có liên quan có thể được biểu diễn như một phần của bảng được minh hoạ trong Hình 3.4.

H3.4 Phát triển ma trận công sức

Chi phí và công sức cho từng chức năng và nhiệm vụ kỹ nghệ phần mềm được tính vào bước cuối. Nếu ước lượng công sức được thực hiện độc lập với ước lượng LOC và FP thì bây giờ có hai ước lượng cho chi phí và công sức để so sánh và củng cố. Nếu cả hai tập các ước lượng này biểu lộ sự phù hợp hợp lý với nhau thì có lý do vững chắc để tin rằng các ước lượng là tin cậy. Mặt khác, nếu các kết quả của hai kỹ

thuật phân rã này không chỉ ra sự phù hợp thì phải nghiên cứu thêm và phải tiến hành phân tích.

Điều gì xảy ra khi thiếu sự phù hợp giữa các ước lượng? Việc trả lời câu hỏi yêu cầu phải đánh giá lại thông tin đã dùng làm ước lượng. Những ước lượng khác biệt lớn thường do hai nguyên nhân:

1. Phạm vi của dự án chức được người lập kế hoạch hiểu thích đáng hoặc bị hiểu sai.
2. Dữ liệu hiệu năng được dùng trong kỹ thuật dòng mã là không thích hợp đối với ứng dụng hiện tại, đã lạc hậu hoặc đã bị hiểu sai.

Người lập kế hoạch phải xác định được nguyên nhân của sự phân tán và củng cố lại ước lượng.

III.6.3 Thí dụ về các ước lượng

Ta xem xét một bộ trình phần mềm cần được phát triển cho ứng dụng thiết kế có máy tính hỗ trợ (CAD). Một tổng quan về *Đặc tả hệ thống* chỉ ra rằng phần mềm cần thực hiện trên máy trạm có tích hợp với nhiều thiết bị ngoại vi đồ hoạ máy tính, chuột, bộ số hoá, màn hình màu độ phân giải cao và máy in lazer.

Dùng bản *Đặc tả hệ thống* như một bản hướng dẫn, có thể xây dựng sơ bộ phạm vi phần mềm CAD như sau:

Phần mềm CAD chấp nhận dữ liệu hình học hai và ba chiều từ người kỹ sư. Người kỹ sư sẽ tương tác và kiểm soát hệ thống CAD thông qua một giao diện người dùng. Giao diện này sẽ thể hiện đặc trưng của một thiết kế giao diện người – máy tốt. Tất cả các dữ liệu hình học và thông tin hỗ trợ khác sẽ được duy trì trong cơ sở dữ liệu CAD. Các modul phân tích thiết kế sẽ được xây dựng để tạo ra cái ra cần thiết, được hiển thị trên nhiều thiết bị đồ hoạ. Phần mềm này sẽ được thiết kế để điều khiển và tương tác các thiết bị ngoại vi kể cả chuột, bộ số hoá, máy in lazer và máy vẽ.

Với mục đích của chúng ta, chúng ta giả sử rằng việc làm mịn thêm phạm vi phần mềm đã có, các chức năng chủ chốt sau đây đã được xác định:

- Giao diện người dùng và tiện nghi điều khiển (UICF)
- Phân tích hình học hai chiều (2DGA)
- Phân tích hình học ba chiều (3DGA)
- Quản trị cơ sở dữ liệu (DBM)
- Tiện nghi hiển thị đồ hoạ máy tính (CGDF)
- Kiểm soát ngoại vi (PC)
- Đơn thể phân tích thiết kế (DAM)

III.6.3.1 Ước lượng LOC và FP

Chúng ta dùng LOC như biến ước lượng. Tuy nhiên FP cũng có thể được dùng và sẽ đòi hỏi các ước lượng về giá trị miền thông tin.

Ta xây dựng một bảng ước lượng như được vẽ trong Hình 3.5. Sau đó xây dựng một phạm vi các ước lượng LOC. Xem ba cột đầu của bảng ta có thể thấy rằng người lập kế hoạch khá chắc chắn về LOC cần cho chức năng kiểm soát thiết bị ngoại vi (chỉ có 450 dòng mã phân cách các ước lượng lạc quan và bi quan). Mặt khác chức năng phân tích hình học ba chiều thì còn chưa được biết rõ, được chỉ ra bởi sự chênh lệch 4000 LOC giữa các giá trị lạc quan và bi quan.

Chức năng	Lạc quan	Có thể	Bi quan	Kỳ vọng	\$/ dòng	Đồng/ tháng	Chi phí	Tháng
Kiểm soát giao diện người dùng	1800	2400	2650					
Phân tích hình học hai chiều	4100	5200	7400					
Phân tích hình học 3 chiều	4600	6900	8600					
Quản lý cấu trúc dữ liệu	2950	3400	3600					
Hiển thị đồ hoạ máy tính	4050	4900	6200					

Điều khiển ngoại vi	2000	3100	2450					
Phân tích thiết kế	6600	8500	9800					

H3.5 Bảng ước lượng

Các tính toán cho giá trị kỳ vọng được thực hiện cho từng chức năng và được đặt vào trong bốn cột của bảng. Bằng cách lấy tổng theo chiều đứng trong cột giá trị kỳ vọng ta lập ra một ước lượng 33360 dòng mã cho hệ thống CAD

Chức năng	Lạc quan	Có thể	Bi quan	Kỳ vọng	\$/ dòng	Đồng/ tháng	Chi phí	Tháng
Kiểm soát giao diện người dùng	1800	2400	2650	2340				
Phân tích hình học hai chiều	4100	5200	7400	5380				
Phân tích hình học 3 chiều	4600	6900	8600	6800				
Quản lý cấu trúc dữ liệu	2950	3400	3600	3350				
Hiển thị đồ hoạ máy tính	4050	4900	6200	4950				
Điều khiển ngoại vi	2000	3100	2450	2140				
Phân tích thiết kế	6600	8500	9800	8400 33360				

H3.5 Bảng ước lượng

Phần còn lại của bảng ước lượng cần cho kỹ thuật phân rã được vẽ trong hình (3.7). Độ đo hiệu năng (suy từ vạch ranh giới lịch sử) có thêm \$/LOC và LOC/người-tháng. Trong trường hợp này người lập kế hoạch dùng các giá trị khác nhau của độ đo hiệu năng cho từng chức năng dựa trên độ phức tạp. Các giá trị được chứa trong cột chi phí và tháng của bảng này được xác định bằng cách lấy tích của LOC kỳ vọng và \$/LOC và LOC/người-tháng tương ứng.

Chức năng	Lạc quan	Có thể	Bi quan	Kỳ vọng	\$/ dòng	Đồng/ tháng	Chi phí	Tháng
-----------	----------	--------	---------	---------	----------	-------------	---------	-------

Kiểm soát giao diện người dùng	1800	2400	2650	2340	14	315	32760	7,4
Phân tích hình học hai chiều	4100	5200	7400	5380	20	220	107600	24,4
Phân tích hình học 3 chiều	4600	6900	8600	6800	20	220	136000	30,9
Quản lý cấu trúc dữ liệu	2950	3400	3600	3350	18	240	60300	13,9
Hiển thị đồ hoạ máy tính	4050	4900	6200	4950	22	200	108900	24,7
Điều khiển ngoại vi	2000	3100	2450	2140	28	140	59920	15,2
Phân tích thiết kế	6600	8500	9800	8400 33360 LOC dự tính	18	300	151200 656680 Chi phí dự tính	28,0 144,5

H3.6 Bảng ước lượng

Từ bảng ước lượng, chi phí dự án được ước lượng vào quãng 657000 \$ và công sức ước lượng là 145 người –tháng. Để suy ra một ước lượng về thời gian cho dự án (theo tháng) ta sử dụng công sức được ước lượng với mô hình kinh nghiệm.

III.6.3.2 Ước lượng chi phí và công sức

Bằng cách tham khảo ước lượng công sức đầy đủ trong Hình 3.7, có thể đưa ra các ước lượng công sức (theo người-tháng) cho từng nhiệm vụ cho kỹ nghệ phần mềm đối với mỗi chức năng phần mềm CAD. Các tổng theo chiều ngang và chiều đứng đưa ra một chỉ dẫn về công sức cần có. Cần lưu ý rằng 75 người – tháng được mở rộng theo nhiệm vụ phát triển phân tích yêu cầu và thiết kế đã chỉ ra tầm quan trọng tương đối của công việc này.

Nhiệm vụ Chức năng	Phân tích yêu cầu	Thiết kế	Mã hoá	Kiểm thử	Tổng cộng
UICF	1.0	2.0	0.5	3.5	7
2 DGA	2.0	10.0	4.5	9.5	26

3 DGA	2.5	12.0	6.0	11.0	31.5
DBM	2.0	6.0	3.0	4.0	15
CGDF	1.5	11.0	4.0	10.5	27
PC	1.5	6	3.5	5	16
DAM	4	14	5	7	30
Tổng	14.5	61	26.5	50.5	152.5
Tỷ lệ	5200	4800	4250	4500	
Chi phí	75 400	292 800	112 625	227 250	708 075

H3.7 Bảng ước lượng

Mọi ước lượng đều theo **người – tháng** trừ khi được chú thích rõ

Tỷ lệ lao động được gắn với từng nhiệm vụ kỹ nghệ phần mềm và được đưa vào trong hàng Tỷ lệ (\$) của bảng. Những dữ liệu này phản ánh chi phí lao động bắt buộc, tức là chi phí lao động chứa tổng phí công ty. Trong thí dụ này ta giả thiết rằng chi phí lao động cho phân tích yêu cầu (\$5200/người) sẽ là 22 phần trăm lớn hơn chi phí cho mã hoá và kiểm thử đơn vị.

Toàn bộ chi phí và công sức được ước lượng cho phần mềm CAD là \$708000 và 153 người – tháng tương ứng. So sánh các giá trị này với dữ liệu được suy ra từ cách dùng dòng mã (LOC) có độ chênh lệch chi phí 7 % và chênh lệch công sức là 5 %. Với tỷ lệ này, có thể đánh giá chúng ta đã đạt tới sự chênh lệch rất ít.

III.7 Mô hình ước lượng kinh nghiệm

Mô hình ước lượng cho phần mềm máy tính dùng các công thức suy ra từ kinh nghiệm để dự đoán dữ liệu vốn là phần cần thiết cho bước lập kế hoạch dự án phần mềm. Dữ liệu kinh nghiệm hỗ trợ cho hầu hết các mô hình được suy dẫn ra từ một mẫu dự án có giới hạn. Vì vậy không có mô hình ước lượng nào thích hợp cho mọi lớp phần mềm và trong mọi môi trường phát triển.

Môi trường tài nguyên: bao gồm nhiều hay một phương trình suy diễn theo kinh nghiệm dự đoán công sức (người-tháng), thời hạn dự án (tháng), hay các dữ liệu dự án thích hợp khác. Có 4 lớp mô hình tài nguyên:

- Mô hình đơn biến tĩnh
- Mô hình đa biến tĩnh
- Mô hình đa biến động
- Mô hình lý thuyết

Mô hình đơn biến tĩnh:

$$\text{Tài nguyên} = c_1 * (\text{đặc trưng được ước lượng}) c_2$$

với tài nguyên có thể là công sức, thời hạn dự án, đội ngũ nhân viên, hay dòng tư liệu phần mềm. Các hằng c_1 và c_2 được suy từ dữ liệu thu thập từ các dự án quá khứ. Đặc trưng được ước lượng là số dòng mã gốc, công sức (nếu được ước lượng), hay các đặc trưng phần mềm khác. Các mô hình lấy dạng được mô tả trên có thể được suy diễn cho môi trường cục bộ nếu có sẵn đủ các dữ liệu lịch sử.

VD: Mô hình chi phí xây dựng (COCOMO) được trình bày trong mục sau

Mô hình đa biến tĩnh, giống như mô hình đơn biến tĩnh, dùng dữ liệu lịch sử để suy diễn ra mối quan hệ kinh nghiệm. Một mô hình điển hình có dạng:

$$\text{Tài nguyên} = c_{1i}e_i + c_{2i}e_2 + \dots$$

Với e_i là đặc trưng phần mềm thứ i ; c_{1i} , c_{2i} là các hằng suy dẫn từ kinh nghiệm cho đặc trưng thứ i .

Mô hình đa biến động phóng chiếu các yêu cầu tài nguyên như một hàm của thời gian. Nếu mô hình này được suy dẫn theo kinh nghiệm thì các tài nguyên được xác định theo các chuỗi thời gian với việc cấp cho một số phần trăm công sức (hay các tài nguyên khác) cho từng bước trong tiến trình kỹ nghệ phần mềm. Mỗi bước có thể chia nhỏ hơn nữa thành các nhiệm vụ. Một cách tiếp cận lý thuyết tới các mô hình đa biến

động giả thiết về một 'đường cong chi phí tài nguyên' liên tục và từ đó suy dẫn ra phương trình mô phỏng hành vi của tài nguyên.

VD: Mô hình ước lượng PUTNAM được trình bày trong mục sau

Mỗi một trong các mô hình được thảo luận ở trên đều đề cập tới các vấn đề vĩ mô trong việc phát triển dự án phần mềm. Việc phân loại mô hình tài nguyên cuối cùng xem xét phần mềm theo quan điểm vi mô, tức là đặc trưng của mã gốc (như các số toán tử và toán hạng).

COCOMO (Constructive Cost Model)

Trong một cuốn sách về kinh tế kỹ nghệ phần mềm, một thứ bậc các mô hình ước lượng phần mềm mang tên COCOMO được trình bày với các dạng sau:

Mô hình 1. COCOMO cơ sở là một đơn trị tính ra công sức để phát triển phần mềm xem như hàm kích cỡ chương trình được diễn đạt theo số dòng mã ước lượng.

Mô hình 2. COCOMO trung bình tính công sức phát triển phần mềm như một hàm của kích cỡ chương trình và một tập các hướng dẫn chi phí bao hàm cả các định giá chủ quan về sản phẩm, phần cứng, nhân sự và thuộc tính dự án.

Mô hình 3. COCOMO nâng cao tổ hợp tất cả các đặc trưng của các bản trung bình với việc định giá của ảnh hưởng của hướng dẫn chi phí lên từng bước (phân tích, thiết kế...) của tiến trình kỹ nghệ phần mềm.

COCOMO có thể được áp dụng cho 3 lớp phần mềm sau

- (1) Mode Tổ chức: Các dự án phần mềm tương đối nhỏ, đơn giản, làm việc với một tập các yêu cầu ít chặt chẽ (vd: chương trình phân tích nhiệt dùng cho nhóm truyền nhiệt)
- (2) Mode Vừa gần: Một dự án phần mềm trung bình về kích cỡ và độ phức tạp, phải đáp ứng các yêu cầu chặt chẽ và kém chặt chẽ (vd: hệ thống xử lý giao tác cho phần cứng là thiết bị đầu cuối và phần mềm là cơ sở dữ liệu)

- (3) Mode Nhúng: Một dự án phần mềm phải được phát triển bên trong một tập phần cứng, phần mềm và các ràng buộc chặt chẽ (như phần mềm kiểm soát bay của máy bay)

Phương trình **COCOMO cơ bản** có dạng:

$$E = a_b (KLOC) \exp (b_p)$$

$$D = c_b (E) \exp (d_b)$$

với E là công sức được áp dụng theo người-tháng, D là thời gian phát triển theo tháng còn KLOC là số được ước lượng về số dòng mã phải bàn giao cho dự án (biểu diễn theo nghìn dòng). Hệ số a_b và c_b và phần mũ b_p và d_b được trong Bảng 3.1

Mô hình cơ sở được mở rộng để xem xét một tập các “thuộc tính hướng dẫn chi phí” có thể được gộp lại thành bốn loại chính:

1. Thuộc tính sản phẩm

- a. Độ tin cậy của phần mềm cần có
- b. Kích cỡ cơ sở dữ liệu ứng dụng
- c. Độ phức tạp của sản phẩm

2. Thuộc tính phần cứng

- a. Ràng buộc hiệu suất khi chạy
- b. Ràng buộc bộ nhớ
- c. Tính dễ thay đổi của môi trường máy ảo
- d. Thời gian quay vòng cần thiết

3. Thuộc tính nhân viên

- a. Khả năng phân tích
- b. Khả năng kỹ nghệ phần mềm

- c. Kinh nghiệm ứng dụng
- d. Kinh nghiệm máy ảo
- e. Kinh nghiệm ngôn ngữ lập trình

4. Thuộc tính dự án

- a. Dùng công cụ phần mềm
- b. Ứng dụng phương pháp kỹ nghệ phần mềm
- c. Lập lịch phát triển theo yêu cầu

Dự án phần mềm	a_b	b_b	c_b	d_b
Tổ chức	2.4	1.05	2.5	0.38
Nửa gần	3.0	1.12	2.5	0.35
Nhúng	3.6	1.20	2.5	0.32

Bảng 3.1 COCOMO cơ sở

Phương trình COCOMO trung bình có dạng:

$$E = a_i (LOC) \exp (b_i) * EAF$$

với E là công sức được áp dụng theo người – tháng còn LOC là số ước lượng về dòng mã phải bàn giao cho dự án. Hệ số a_i và số mũ b_i được cho trong bảng 3.1.

Dự án phần mềm	a_i	b_i
Tổ chức	3.2	1.05
Nửa gần	3.0	1.12

Nhúng	2.8	1.20
-------	-----	------

Bảng 3.2 COCOMO trung bình

COCOMO biểu thị cho một mô hình kinh nghiệm toàn diện cho việc ước lượng phần mềm. Để minh họa cho việc dùng COCOMO chúng ta áp dụng mô hình cơ sở cho thí dụ phần mềm CAD đã mô tả trước đây. Dùng ước lượng LOC được phát triển trong Hình 3.5 và hệ số cho trong Bảng 3.1 để được:

$$E = 3.0 (LOC) \exp (1.12)$$

$$= 3.0 (33.3)^{1.12}$$

$$= 152 \text{ người – tháng}$$

Giá trị này khá phù hợp với ước lượng được suy dẫn trong các mục trước (ước lượng LOC, FP và công sức).

Để tính thời hạn dự án ta dùng ước lượng công sức được mô tả ở trên:

$$D = 2.5 (E) \exp (0.35)$$

$$= 2.5 (152)^{0.35}$$

$$= 14.5 \text{ tháng}$$

Giá trị cho thời hạn dự án giúp cho người lập kế hoạch xác định một số người cần cho dự án

$$N = E/D$$

$$= 152/14.5$$

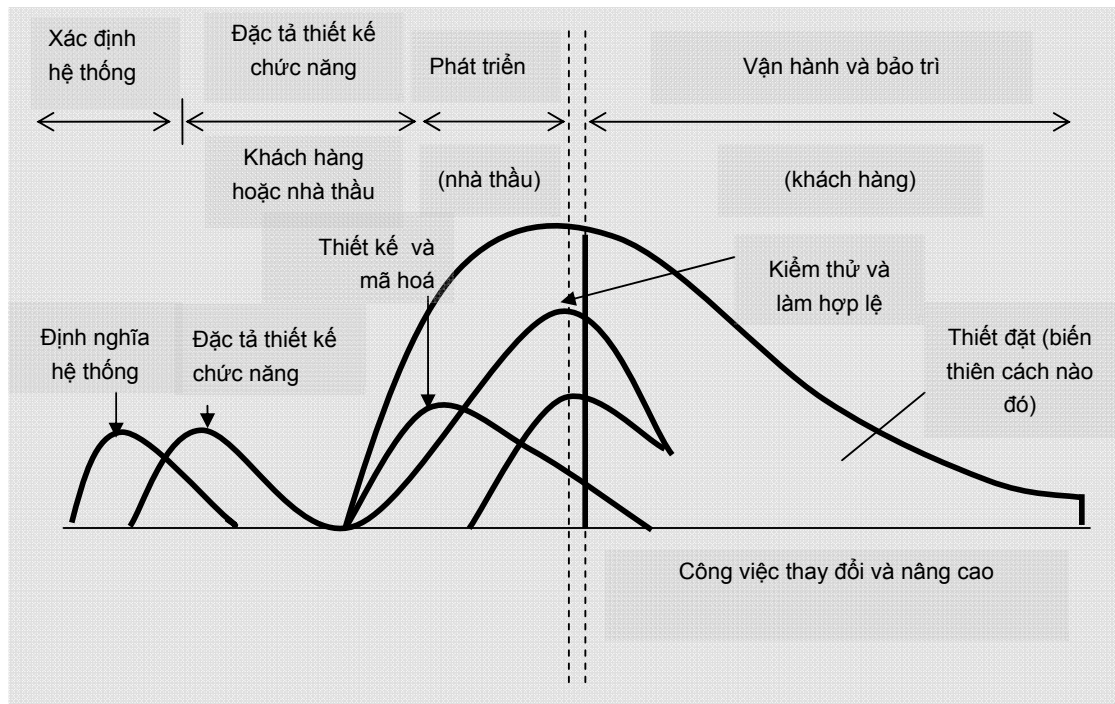
$$\sim 11 \text{ người}$$

Trong thực tế người lập kế hoạch có thể quyết định dùng bốn người và kéo dài thời hạn dự án tương ứng.

III.8 Mô hình ước lượng Putnam

Mô hình ước lượng PUTMAN là một mô hình đa biến động giả thiết có một sự phân bố riêng về nỗ lực trên vòng đời của dự án phát triển phần mềm. Mô hình này đã được suy dẫn từ phân bố lao động thường gặp phải trong các dự án lớn (công sức tổng cộng khoảng 30 người – năm hoặc hơn). Tuy nhiên, có thể làm phép ngoại suy cho các dự án phần mềm nhỏ hơn.

Phân bố công sức cho các dự án phần mềm lớn có thể được biểu diễn như hình 3.8. Đường cong được vẽ trong hình này có dạng cổ điển lần đầu tiên được Rayleigh mô tả theo cách phân tích. Dữ liệu kinh nghiệm trên việc phát triển hệ thống, do Norden thu thập, đã được dùng để thay thế cho đường cong. Do đó, phân bố công sức được vẽ trên Hình vẽ còn được gọi là đường cong *Rayleigh – Norden*.



H3.8 Phân bố công sức - điểm lớn

Đường cong Rayleigh – Norden có thể được dùng để suy dẫn ra “phương trình phần mềm” đặt quan hệ giữa số các dòng mã bàn giao (câu lệnh gốc) với công sức và thời gian phát triển như sau:

$$L = C_k K^{1/3} t_d^{4/3}$$

Với C_k là hằng trạng thái công nghệ và phản ánh “ràng buộc hiệu năng là trở ngại tiến độ của người lập trình”. Các giá trị điển hình có thể là: $C_k = 2000$ cho môi trường phát triển phần mềm nghèo nàn (như không có phương pháp luận, ít tư liệu, mốt thực hiện theo lô); $C_k = 8000$ cho môi trường phát triển phần mềm tốt (tức là có phương pháp luận, tài liệu thích hợp, một thực hiện tương tác); $C_k = 11000$ cho một môi trường “tuyệt vời” (công cụ và kỹ thuật tự động hoá). Hằng C_k có thể được suy dẫn cho các điều kiện cục bộ dùng trữ liệu lịch sử thu thập từ công thức phát triển quá khứ. Bố trí lại phương trình phần mềm trên chúng ta có thể đi tới một biểu thức cho công sức phát triển K^* :

$$K = \frac{L^3}{C_k^3 t_d^4}$$

Với K là công sức dành ra (theo người - năm) cho toàn bộ vòng đời phát triển và bảo trì phần mềm, còn t_d là thời gian phát triển theo năm. Phương trình cho công sức phát triển có thể được liên hệ với chi phí phát triển bằng việc bao hàm nhân tố tỷ lệ lao động (\$/người - năm).

Bởi mối quan hệ lũy thừa cấp cao cho trong phương trình phần mềm có thể chỉ ra rằng sự mở rộng tương đối nhỏ trong ngày giao hàng có thể tạo ra sự tiết kiệm khá lớn cho công sức người dùng trong dự án. Nói cách khác, mối quan hệ giữa công sức cần dùng và thời hạn giao hàng là phi tuyến cao độ.

III.9 Công cụ ước lượng tự động

Kỹ thuật phân rã và các mô hình ước lượng kinh nghiệm được mô tả ở các mục trên có thể được cài đặt trong các phần mềm. Những công cụ ước lượng tự động hoá này cho phép người lập kế hoạch ước lượng chi phí và công sức và thực hiện các phân tích. Mặc dù đã có nhiều công cụ ước lượng tự động hoá nhưng tất cả đều biểu lộ cùng đặc trưng tổng quát và đều yêu cầu một hay nhiều phân loại dữ liệu sau:

1. Ước lượng định lượng về cỡ dự án (LOC) hay chức năng (FP)
2. Các đặc trưng định lượng về dự án như độ phức tạp, độ tin cậy cần có hay sự gắt gao nghiệp vụ.
3. Một mô tả nào đó về đội ngũ phát triển và/hoặc môi trường phát triển.

Từ những dữ liệu này, mô hình được cài đặt bởi công cụ ước lượng tự động hoá đưa ra những ước lượng về công sức cần bỏ ra để hoàn thành dự án, chi phí, đội ngũ nhân viên, và trong một số trường hợp, phát triển cả lịch biểu và rủi ro có liên hệ.

Một số công cụ ước lượng tự động hoá dựa trên COCOMO như: **BYL** (Before – You Leap - Trước khi bạn hành động) được nhóm GORDEN phát triển, **WICOMO** (Wang Institute Cost Model) do viện Wang phát triển, **DECPlan** do công ty DEC phát triển. Mỗi một trong các công cụ này đều yêu cầu người dùng phải cung cấp các ước lượng LOC ban đầu. Những ước lượng này được phân loại theo ngôn ngữ lập trình. Người dùng cũng phải xác định giá trị cho các thuộc tính hướng dẫn chi phí đã mô tả trong các mục trên.

Các công cụ giúp chúng ta đưa ra: thời gian dự án (tháng), công sức (người - tháng), số nhân viên trung bình/một tháng, tính hiệu năng trung bình theo LOC/phần mềm, chi phí theo tháng. Những dữ liệu này có thể được phát triển cho từng giai đoạn trong tiến trình kỹ nghệ phần mềm hay cho toàn bộ dự án.

Một hệ thống tính chi phí tự động dựa trên đường cong Rayleigh – Norden cho vòng đời phần mềm và mô hình ước lượng Putnam là **SLIM**. Hệ thống này giúp cho người lập kế hoạch phần mềm thực hiện các chức năng: (1) Định cỡ môi trường phát triển phần mềm bằng cách diễn giải dữ liệu lịch sử do người lập kế hoạch cung cấp;

(2) tạo ra một mô hình thông tin về phần mềm cần phát triển bằng cách suy luận ra các đặc trưng phần mềm cơ sở, thuộc tính nhân sự và những xem xét về môi trường; (3) tiến hành định cỡ phần mềm. Cách tiếp cận trong SLIM là một bản phức tạp hơn, tự động hoá hơn trong kỹ thuật kỹ thuật đánh giá LOC.

ESTIMACS là một mô hình ước lượng vĩ mô dùng phương pháp ước lượng điểm chức năng có nâng cao để kiểm soát sự biến thiên của các nhân tố dự án và con người. Công cụ này chứa một các mô hình làm cho người lập kế hoạch có khả năng ước lượng được (1) công sức phát triển hệ thống, (2) nhân viên và chi phí, (3) cấu hình phần cứng, (4) rủi ro, (5) hiệu quả vốn đầu tư phát triển.

SPQR/20 do Software Productivity Research phát triển, cho người dùng một tập đơn giản các câu hỏi nhiều chọn lựa, đề cập tới: kiểu dự án (chương trình mới, bảo trì), phạm vi dự án (bản mẫu, modul dùng lại), mục tiêu (thời hạn tối thiểu, chất lượng cao nhất), lớp dự án (chương trình, sản phẩm cá nhân), kiểu ứng dụng (theo lô, hệ chuyên gia), tài liệu người dùng (hình thức, không hình thức), thời gian đáp ứng, kinh nghiệm nhân viên, phần trăm mã gốc dùng lại, ngôn ngữ lập trình, độ phức tạp logic của thuật toán, độ phức tạp mã và dữ liệu ...là những cái vào. SPQR/20 sẽ đưa ra các dữ liệu đã được mô tả cho các công cụ khác, ngoài ra còn ước lượng tổng số trang tư liệu dự án, tổng số khiếm khuyết tiềm năng của dự án, tổng số khiếm khuyết khi bàn giao, số các khiếm khuyết theo KLOC...

Từng công cụ ước lượng tự động hoá này đều có một chế độ đối thoại với người lập kế hoạch, lấy được dự án thích hợp và tạo cái ra theo bảng hay đồ hoạ. Tất cả những công cụ được mô tả ở trên đều đã được cài đặt trên máy tính cá nhân hay máy trạm làm việc công nghệ.

Nếu mỗi công cụ đều được áp dụng vào cùng một dự án, cũng không có gì ngạc nhiên là người ta sẽ gặp phải sự biến thiên tương đối lớn giữa những kết quả thu được, đôi khi còn khác biệt với dữ liệu thực tế. Điều này lưu ý là cái ra của các công cụ ước lượng nên được dùng như một “điểm dữ liệu” để suy dẫn ra các ước lượng.

Tóm tắt

Người lập kế hoạch dự án phần mềm phải ước lượng ba điều trước khi một dự án bắt đầu: dự án kéo dài bao lâu, cần bao nhiêu nỗ lực và bao nhiêu người cần tham gia. Bên cạnh đó, người lập kế hoạch phải dự kiến các tài nguyên (phần cứng và phần mềm) sẽ cần tới và rủi ro trong dự án.

Xác định phạm vi phần mềm giúp cho người lập kế hoạch phát triển các ước lượng dùng một hay nhiều kỹ thuật sau: phân rã, mô hình kinh nghiệm, công cụ ước lượng tự động hoá. Các kỹ thuật phân rã đòi hỏi mô tả các chức năng phần mềm chính, tiếp đó là các ước lượng hoặc về số các LOC hay FP hay số người – tháng cần cho việc thực thi từng chức năng. Các kỹ thuật kinh nghiệm dùng các biểu thức dựa trên kinh nghiệm về công sức và thời gian dự kiến định lượng cho dự án. Các công cụ tự động hoá cài đặt một mô hình kinh nghiệm xác định.

Việc ước lượng dự án chính xác nói chung đòi hỏi phải dùng ít nhất hai trong ba kỹ thuật đã nêu trên. Bằng cách so sánh và điều tiết các ước lượng được suy ra khi dùng các kỹ thuật khác nhau, người lập kế hoạch có thể suy ra một ước lượng chính xác. Việc ước lượng dự án phần mềm không phải là một khoa học chính xác, nhưng tổ hợp các dữ liệu lịch sử tốt cũng như các kỹ thuật hệ thống có thể cải thiện độ chính xác của ước lượng.

CHƯƠNG IV

LẬP KẾ HOẠCH DỰ ÁN PHẦN MỀM

Bước đầu tiên của việc lập kế hoạch phần mềm là việc ước lượng đã được giới thiệu trong chương trước. Việc ước lượng cung cấp cho người lập kế hoạch dự án thông tin cần để hoàn chỉnh các hoạt động lập kế hoạch dự án còn lại. Trong chương này các hoạt động như: phân tích rủi ro, lập lịch, việc ra quyết định tái kỹ nghệ và lập kế hoạch tổ chức sẽ được trình bày.

IV.1 Phân tích rủi ro

Mối quan tâm của chúng ta là: Rủi ro nào có thể gây ra phần mềm bị thất bại? Làm sao những thay đổi về yêu cầu, sự phát triển về công nghệ ...lại ảnh hưởng đến thời gian và sự thành công của toàn bộ dự án? Chúng ta nên dùng phương pháp và công cụ nào, cần huy động bao nhiêu người, nhấn mạnh bao nhiêu vào chất lượng là đủ?

IV.1.1 Xác định rủi ro

Để có thể xác định được đúng rủi ro trong một dự án phần mềm, điều quan trọng là phải xác định mọi rủi ro cho cả người quản lý và người phát triển.

Có thể phân loại rủi ro theo nhiều cách. Ở mức vĩ mô, gồm có các rủi ro: rủi ro dự án, rủi ro kỹ thuật và rủi ro nghiệp vụ.

Rủi ro dự án: xác định các vấn đề yêu cầu, khách hàng, tài nguyên, nhân sự, lịch biểu, ngân sách tiềm năng và ảnh hưởng của chúng ta lên dự án phần mềm. Độ phức tạp, kích cỡ và cấu trúc dự án cũng được xác định như các nhân tố rủi ro.

Rủi ro kỹ thuật: xác định các vấn đề tiềm năng về thiết kế, cài đặt, giao diện, kiểm chứng và bảo trì. Bên cạnh đó, độ bất trắc kỹ thuật, sự lạc hậu kỹ thuật cũng là những nhân tố rủi ro.

Rủi ro nghiệp vụ: là ở trên trong bởi vì chúng có thể làm sáng tỏ kết quả của ngay cả dự án phần mềm tốt nhất. Ứng cử viên cho năm loại rủi ro nghiệp vụ là:

- (1) xây dựng một sản phẩm tuyệt vời mà chẳng ai thực sự muốn (rủi ro tiếp thị)
- (2) xây dựng một sản phẩm chẳng thích hợp với chiến lược sản phẩm tổng thể của công ty
- (3) mất sự hỗ trợ của cấp quản lý cao do thay đổi mối quan tâm (rủi ro quản lý)
- (4) mất cam kết về tài chính hay nhân sự (rủi ro ngân sách)

Cần phải lưu ý rằng sự phân loại đơn giản trên không phải là triệt để. Một số rủi ro đơn thuần là không thể tiến đoán được.

Xác định rủi ro: Một trong những phương pháp tốt nhất để hiểu từng rủi ro là dùng một tập các câu hỏi giúp cho người lập kế hoạch dự án hiểu được rủi ro trong dự án. Người lập kế hoạch có thể đạt tới cảm nhận về rủi ro nghiệp vụ bằng cách trả lời các câu hỏi sau:

- Có người giỏi nhất không?
- Mọi người có tổ hợp đúng tài năng không?
- Có đủ người sẵn có không?
- Các nhân viên có cam kết theo đuổi toàn bộ dự án không?
- Một số nhân viên dự án chỉ dành một phần thời gian cho dự án?
- Các nhân viên có mong đợi đúng việc hiện được giao không?
- Các nhân viên có được huấn luyện đầy đủ không?
- Có hay luân chuyển nhân viên làm gián đoạn công việc hay không?

Sự chắc chắn tương đối của các câu trả lời cho những câu hỏi này sẽ cho phép người lập kế hoạch ước lượng được ảnh hưởng của rủi ro.

IV.1.2 Dự phòng rủi ro

Dự phòng rủi ro hay còn được gọi là ước lượng rủi ro, cố gắng xác định tỷ lệ từng rủi ro theo hai cách - *có thể xảy ra* tức là rủi ro là có thực và *hậu quả* của vấn đề liên quan tới rủi ro, nếu nó xuất hiện.

Người lập kế hoạch dự án cùng với các nhà quản lý khác và các nhân viên kỹ thuật thực hiện 4 hoạt động dự phòng rủi ro:

- (1) lập thang phản ánh khả năng khả năng có thể xảy ra của rủi ro
- (2) phác hoạ những hậu quả của rủi ro
- (3) ảnh hưởng ước lượng ảnh hưởng của rủi ro lên dự án và sản phẩm
- (4) chú ý đến độ chính xác của dự phòng rủi ro sao cho không có sự hiểu lầm

Thang có thể được xác định theo thuật ngữ có – không, định tính hoặc định lượng. Mỗi câu hỏi trong trong bảng khoản mục rủi ro trong mục IV.1.2 đều có thể trả lời bằng có hoặc không. Tuy nhiên, hiếm khi có thể định giá rủi ro theo cách tuyệt đối như vậy. Cách tiếp cận tốt hơn là trả lời theo *tỷ lệ xác suất* định lượng có giá trị sau: gần như không thể có, vừa phải, có thể, rất có thể. Theo một cách khác người lập kế hoạch có thể ước lượng xác suất toán học mà một rủi ro có thể xảy ra (VD: 0.90 là rủi ro rất có thể xảy ra). Người ta có thể ước lượng xác suất bằng cách dùng các phân tích thống kê về các độ đo thu được từ các dự án quá khứ, từ trực giác hay từ các thông tin khác. Chẳng hạn, nếu độ đo thu thập được từ 45 dự án chỉ ra rằng 37 dự án đã kinh qua gấp đôi số những thay đổi khách hàng so với dự kiến thì xác suất một dự án mới bị thay đổi rất có thể vượt qua con số thay đổi $37/45 = 0.82$.

Cuối cùng rủi ro được đánh trọng số bởi ảnh hưởng thu được cảm nhận (theo dự án) rồi được sắp thứ tự ưu tiên. Ba nhân tố chi phối ảnh hưởng: bản chất rủi ro, phạm vi của nó và thời gian:

Bản chất của rủi ro chỉ ra vấn đề có thể xảy ra nếu nó xuất hiện. Chẳng hạn, một giao diện ngoài kém xác định đối với phần cứng khách hàng (một rủi ro kỹ thuật) sẽ ngăn cản thiết kế sớm và rất có thể sẽ dẫn tới vấn đề tích hợp hệ thống về sau trong dự án.

Phạm vi của một rủi ro tổ hợp cả sự ngặt nghèo (vấn đề nghiêm trọng đến đâu?) với phân bổ tổng thể của nó (dự án sẽ bị ảnh hưởng đến đâu hay bao nhiêu khách hàng sẽ phải chịu thiệt hại)

Thời gian của một rủi ro là thời điểm rủi ro có tác động và kéo dài trong bao lâu.

Bảng 4.1 Đại cương kế hoạch quản lý và điều phối rủi ro

I. Giới thiệu	
1. Phạm vi và mục tiêu của tài liệu	
2. Tổng quan	
a. Mục tiêu	
b. Ưu tiên và khắc phục rủi ro	
3. Tổ chức	
a. Quản lý	
b. Trách nhiệm	
c. Mô tả công việc	
4. Mô tả chương trình khắc phục	
a. Lịch biểu	
b. Các mốc và cuộc họp chính	
c. Ngân sách	
II. Phân tích rủi ro	
1. Xác định	
a. Tổng quan về rủi ro, nguồn rủi ro	
b. Phân loại rủi ro	
2. Ước lượng rủi ro	

a. Ước lượng xác suất rủi ro	
b. Ước lượng hậu quả rủi ro	
c. Ước lượng tiêu chí	
d. Nguồn có thể về lỗi ước lượng	
3. Đánh giá	
a. Phương pháp đánh giá được dùng	
b. Giả thiết và giới hạn của phương pháp đánh giá	
c. Đánh giá tham khảo rủi ro	
d. Đánh giá kết quả	
III. Quản lý rủi ro	
1. Khuyến cáo	
2. Các tùy chọn khác	
3. Đánh giá tham khảo rủi ro	
4. Thủ tục điều phối rủi ro	
IV. Phụ lục	
1. Ước lượng tình huống rủi ro	
2. Kế hoạch giảm rủi ro	

IV.1.3 Quản lý rủi ro

Giả sử việc quay vòng nhân viên nhiều lần có thể được coi như một rủi ro của dự án P nào đó. Với trường hợp này, quản lý rủi ro gồm các bước sau đây:

- Hợp với nhóm nhân viên hiện tại để xác định các nguyên nhân thay đổi nhân lực như điều kiện làm việc kém, lương thấp, thị trường công việc cạnh tranh...

- Hành động để giảm nhẹ các nguyên nhân dưới quyền kiểm soát của chúng ta ngay khi dự án bắt đầu
- Khi dự án bắt đầu, phải giả sử việc thay đổi người sẽ xuất hiện, phải xây dựng các kỹ thuật để đảm bảo sự liên tục khi có người ra đi
- Tổ chức nhóm dự án sao cho thông tin về từng hoạt động được phổ biến rộng rãi.
- Xác định các chuẩn tài liệu và thiết lập cơ chế để đảm bảo rằng tài liệu được xây dựng theo đúng hạn
- Tiến hành xét duyệt ngang cấp cho toàn bộ công việc cho người người được thúc đẩy tăng tốc
- Xác định các nhân viên dự phòng cho mọi nhà kỹ thuật chủ chốt

Điều quan trọng cần lưu ý là những bước quản lý rủi ro này phải chịu thêm chi phí dự án phụ. Vì dành thời gian dự phòng cho các nhà kỹ thuật chủ chốt sẽ làm tốn thêm tiền của. Chính vì vậy phải đánh giá khi nào lợi ích được tích lũy bởi các bước quản lý rủi ro trội hơn chi phí để thực hiện chúng. Dựa trên lịch sử quá khứ và trực giác quản lý, nếu các bước khắc phục rủi ro được dự phòng làm tăng chi phí lên 5% và thời hạn kéo dài thêm 3% thì cấp quản lý có thể đưa chúng vào kế hoạch.

Với một dự án lớn, có thể xác định 30 đến 40 rủi ro. Nếu xác định cho mỗi rủi ro từ ba đến bảy bước quản lý thì bản thân việc quản lý rủi ro lại trở thành một dự án. Thực tế chỉ ra rằng, 80% của toàn bộ rủi ro dự án có thể được tính chỉ bởi 20% của các rủi ro được xác định. Công việc ban đầu trong các bước phân tích rủi ro sẽ giúp người lập kế hoạch xác định các rủi ro rơi vào 20% đó. Vì vậy một số rủi ro được xác định, được dự phòng không thể trở thành kế hoạch quản lý rủi ro - chúng không chứa 20% cần thiết các rủi ro với ưu tiên cao nhất.

IV.2 Lập lịch dự án phần mềm

Lập lịch cho các dự án phát triển phần mềm có thể được xem xét theo hai bối cảnh khác nhau. Trước hết ngày cuối cần phải bàn giao hệ thống đã được ấn định rõ. Tổ chức phần mềm phải phân bổ nỗ lực trong khuôn khổ thời gian định. Thứ hai, giả

sử thời hạn dự án đã được thảo luận nhưng ngày cuối do tổ chức phần mềm ấn định. Nỗ lực được phân bổ để làm cho việc sử dụng tài nguyên một được tốt nhất, và ngày cuối đó sẽ được xác định sau khi phân tích cẩn thận về phần mềm. Trong thực tế, trường hợp thứ nhất thường gặp hơn rất nhiều so với trường hợp thứ hai.

Khi chúng ta tiếp cận với việc lập lịch dự án, có một số câu hỏi cần được trả lời:

- Làm sao chúng ta có thể phối hợp thời gian với nỗ lực con người?
- Ta trông đợi nhiệm vụ và cơ chế song song nào?
- Vạch mốc nào có thể được dùng để chỉ ra tiến độ?
- Phân phối nỗ lực thế nào trong toàn tiến trình kỹ nghệ phần mềm?
- Phương pháp lập lịch nào hiện có?
- Làm sao ta có thể biểu diễn một lịch biểu và theo dõi tiến độ khi dự án bắt đầu?

Các mục sau đây sẽ đề cập đến từng câu hỏi trên.

IV.2.1 Mối quan hệ con người – công việc

Trong một dự án phát triển phần mềm nhỏ, một người có thể phân tích các yêu cầu, thực hiện thiết kế, sinh mã và tiến hành kiểm thử. Khi dự án lớn ra, sẽ phải có nhiều người hơn cùng tham dự.

Việc thêm người về sau trong dự án thường có ảnh hưởng phá vỡ dự án, gây cho dự án lịch biểu càng bị trượt thêm nữa vì người được bổ xung thêm phải học về hệ thống, hơn nữa những người dạy họ lại cũng là những người đang làm công việc đó. Bên cạnh vấn đề thời gian cần để học hệ thống, số người tăng lên thì số đường liên lạc cũng tăng lên và độ phức tạp của liên lạc trong hệ thống cũng tăng lên.

Dựa trên mối quan hệ người – công việc như vậy thì nhóm có phải là không hiệu quả không? Câu trả lời là không, trong thực tế các cuộc họp tổng kết kỹ thuật chính thức (sẽ được trình bày sau) do các nhóm phát triển phần mềm tiến hành có thể dẫn tới việc phân tích và thiết kế phần mềm tốt hơn và quan trọng là giảm số lỗi chưa được

phát hiện vào lúc kiểm thử. Do vậy tính hiệu quả và chất lượng dự án khi đo theo thời gian có thể được cải tiến.

Mô hình Rayleigh – Norden cho các dự án lớn, dự đoán một mối quan hệ phi tuyến cao độ giữa thời gian để hoàn thành dự án và công sức bỏ ra. Điều này dẫn tới một số kết quả thú vị, trong ví dụ về phần mềm CAD, công sức 12 người – năm, 33 000 LOC đã được ước lượng sẽ hoàn thành với 8 người làm việc trong 1,3 năm. Tuy nhiên, nếu chúng ta mở rộng hạn cuối tới 1,75 năm thì từ phương trình phần mềm của Putnam:

$$K = L3 (C_k^3 + t_d^4) \sim 3,8 \text{ người – năm}$$

Điều này kéo theo rằng bằng cách kéo dài thời hạn chót thêm 6 tháng chúng ta có thể giảm số người từ 8 xuống 4. Vì thế, lợi ích có thể thu được bằng cách dùng ít người hơn với một thời gian dài hơn để hoàn thành mục tiêu.

IV.2.2 Xác định nhiệm vụ

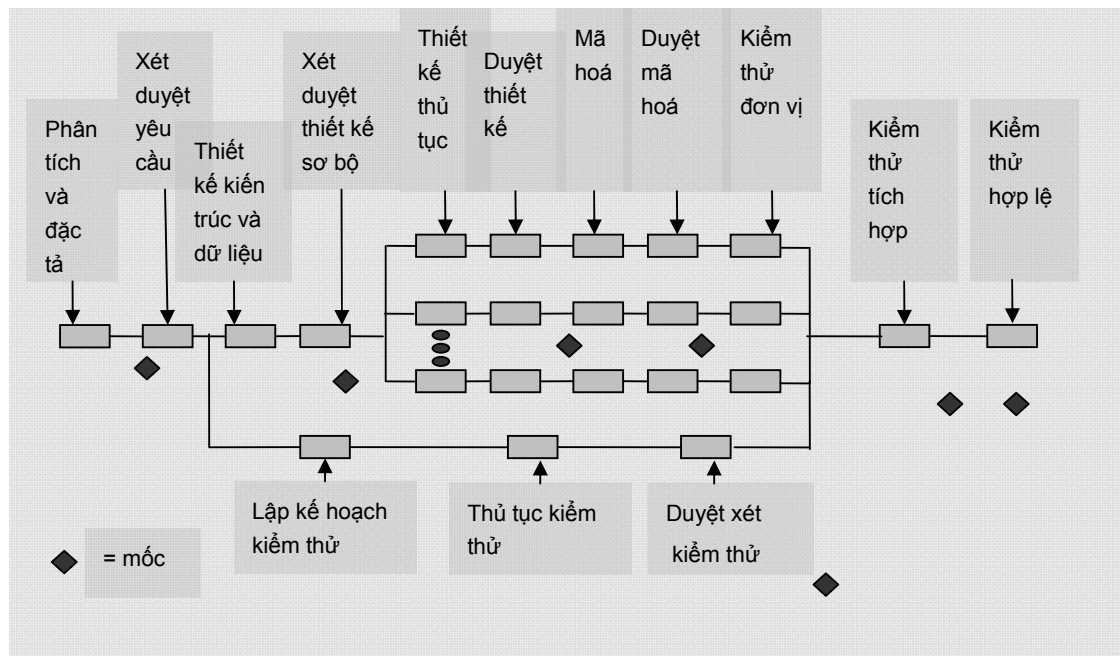
Khi có nhiều người cùng tham gia trong một dự án kỹ nghệ phần mềm thì rất có thể là các hoạt động phát triển sẽ được thực hiện song song. Hình 4.1 chỉ ra một sơ đồ *mạng nhiệm vụ* cho một dự án kỹ nghệ phần mềm có nhiều người. Mạng này biểu thị cho tất cả các nhiệm vụ dự án, trình tự của chúng và sự phụ thuộc của chúng.

Phân tích, đặc tả và tổng quan về các yêu cầu đối với kết quả là những nhiệm vụ đầu tiên cần phải thực hiện, sau đó là đặt nền tảng cho các nhiệm vụ song song theo sau.

Một khi các yêu cầu đã được xác định và xem xét thì hoạt động thiết kế (thiết kế kiến trúc và dữ liệu) và việc vạch kế hoạch kiểm thử có thể bắt đầu song song. Bản chất modul của phần mềm thiết kế tốt tự bản thân nó đã giúp cho việc theo dõi sự phát triển song song cho thiết kế thủ tục, mã hoá và kiểm thử đơn vị. Khi các thành phần của phần mềm được hoàn tất, nhiệm vụ kiểm thử tích hợp bắt đầu. Cuối cùng, kiểm thử hợp lệ để làm cho phần mềm sẵn sàng bàn giao cho khách hàng.

Tham khảo đến hình 4.1, điều quan trọng cần lưu ý là các mốc được đặt ở những khoảng đều đặn trong toàn bộ tiến trình công nghệ cung cấp cho nhà quản lý

một chỉ báo đều đặn về tiến độ. Các mốc được đạt tới khi tài liệu được tạo ra, xem như một phần của nhiệm vụ kỹ nghệ phần mềm đã thành công.



H 4.1 Mạng nhiệm vụ và cơ chế song song

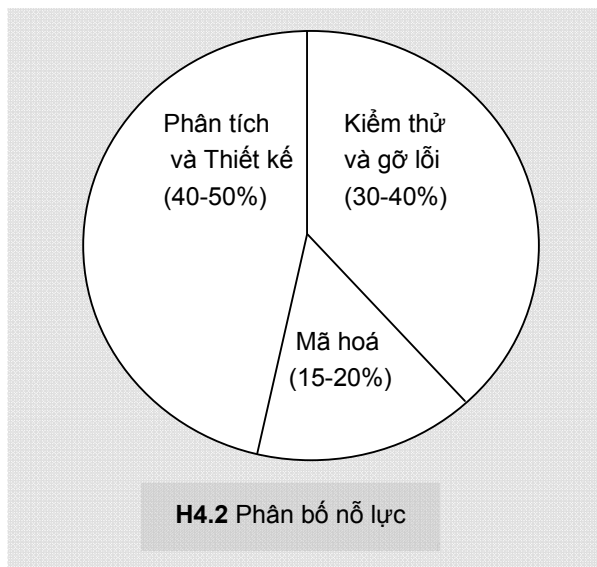
Bản chất tương tranh của các hoạt động kỹ nghệ phần mềm dẫn tới một số yêu cầu lập lịch quan trọng. Bởi vì các nhiệm vụ song song xuất hiện không đồng bộ, nên người lập kế hoạch phải xác định sự phụ thuộc giữa các nhiệm vụ để đảm bảo tiến độ liên tục hướng tới hoàn tất. Bên cạnh đó, người quản lý dự án cần biết về những nhiệm vụ phải được hoàn tất theo lịch biểu để dự án hoàn thành theo đúng tiến độ.

IV.2.3 Phân bổ nỗ lực

Từng kỹ thuật ước lượng dự án phần mềm đã được thảo luận dẫn tới các ước lượng về người – tháng (hoặc người - năm) cần để hoàn tất việc phát triển phần mềm.

Hình 4.2 minh hoạ cho một phân bố đề nghị về công sức cần thực hiện cho các giai đoạn xác định và phát triển. Phân bố này, còn được gọi là quy tắc 40-20-40, nhấn mạnh vào các nhiệm vụ phân tích và thiết kế đầu trước và kiểm thử đầu sau, phần mã hoá là không được nhấn mạnh.

Phân bố công sức được vẽ trong hình 4.2 chỉ nên được dùng như bản hướng dẫn. Các đặc trưng của từng dự án là phải quyết định ra cách phân phối công sức. Công sức dành cho việc lập kế hoạch hiếm khi được tính lớn hơn từ 2 đến 3 % công sức, trừ trường hợp kế hoạch dự án đáng đáng đến một tổ chức với chi phí lớn và rủi ro cao. Việc phân tích yêu cầu có thể chiếm 10 đến 25 % công sức dự án. Công sức dành cho phân tích là làm bản mẫu cần phải tăng lên trực tiếp với kích cỡ và độ phức tạp dự án. Phạm vi từ 20 đến 25 % công sức thông thường được áp dụng cho thiết kế phần mềm. Vì công sức tập trung cho thiết kế phần mềm nên việc mã hoá đi theo hướng tương đối ít khó khăn. Có thể đạt tới một phạm vi 15 đến 20 phần trăm trên tổng số công sức. Kiểm thử và gỡ lỗi sau đó có thể chiếm 30 đến 40 % công sức phát triển phần mềm, nếu phần mềm bị lỗi do con người thì có thể xem xét tăng số phần trăm nỗ lực có hơn dành cho việc này.



IV.2.4 Phương pháp lập lịch

Lập lịch cho dự án phát triển phần mềm không khác so với việc lập lịch cho bất kỳ nỗ lực phát triển đa nhiệm nào. Do vậy, các công cụ kỹ thuật lập lịch dự án tổng quát có thể được áp dụng cho phần mềm với một chút sửa đổi.

Kỹ thuật xem xét và đánh giá chương trình (PERT) và phương pháp đường găng (CPM) là hai phương pháp lập lịch dự án có thể được áp dụng cho việc phát triển phần mềm. Cả hai kỹ thuật này đều xây dựng việc mô tả dự án theo mạng nhiệm vụ, tức là việc biểu diễn theo hình hay bảng cho các nhiệm vụ cần phải tiến hành từ đầu tới cuối dự án. Mạng được xác định bằng cách xây dựng một danh sách tất cả các nhiệm vụ (đôi khi còn được gọi là cấu trúc phân việc dự án) liên kết với một dự án xác định và một danh sách có trật tự chỉ ra các nhiệm vụ phải thực hiện theo thứ tự nào.

Cả PERT và CPM đều cung cấp các công cụ định lượng cho phép người lập kế hoạch phần mềm:

- (1) xác định *đường găng* – dây chuyền các nhiệm vụ xác định ra thời hạn dự án
- (2) thiết lập ước lượng thời gian *có thể nhất* cho từng nhiệm vụ bằng cách áp dụng các mô hình thống kê.
- (3) tính toán thời gian biên, cái tạo ra “cửa sổ thời gian” cho một nhiệm vụ đặc biệt

Việc tính toán thời gian biên có thể rất có ích trong việc lập lịch dự án. Độ trượt trong thiết kế của một chức năng, có thể làm chậm thêm việc xây dựng các chức năng khác. Thời gian biên quan trọng có thể thấy được từ mạng PERT và CPM:

- (1) Thời gian sớm nhất cho một nhiệm vụ có thể bắt đầu
- (2) Thời gian muộn nhất cho khởi đầu một nhiệm vụ trước khi thời gian tối thiểu để hoàn thành dự án bị trễ
- (3) Sự kết thúc sớm nhất - cộng thời gian bắt đầu sớm nhất và thời hạn nhiệm vụ
- (4) Sự kết thúc muộn nhất - cộng thời gian bắt đầu muộn nhất với thời gian nhiệm vụ
- (5) Tổng độ nổi - khối lượng thời gian chậm trễ được phép theo các nhiệm vụ lập lịch để cho đường găng trên mạng được duy trì theo lịch biểu.

Việc tính toán thời gian biên dẫn tới việc xác định đường găng và cung cấp cho người quản lý một phương pháp định lượng để ước tính tiến độ cũng như nhiệm vụ cần hoàn thành.

Người lập kế hoạch phải nhận ra rằng nỗ lực trải trên phần mềm không kết thúc vào cuối việc xây dựng. Nỗ lực bảo trì, mặc dầu không dễ lập lịch tại giai đoạn này, cuối cùng sẽ trở thành nhân tố chi phí lớn nhất. Một trong những mục tiêu của kỹ nghệ phần mềm là giảm thiểu chi phí này.

IV.2.5 Thí dụ về lập lịch

Các kỹ thuật lập lịch được mô tả trong mục trước có thể được cài đặt với công cụ lập lịch dự án tự động có sẵn. Những công cụ như vậy tương đối dễ dùng và làm cho phương pháp phân tích được mô tả ở trên thành có sẵn với mọi người quản lý dự án phần mềm.

Mục này trình bày một thí dụ lập lịch dự án được xây dựng có dùng công cụ lập lịch tự động, MacProject II. Mạng nhiệm vụ dự án được nêu trên Hình 4.3 .a cho tới e., được vẽ theo chiều trao đổi tương tác giữa người lập kế hoạch dự án và phần mềm. Các hộp chữ nhật biểu thị cho các nhiệm vụ kỹ nghệ phần mềm, các hộp tròn góc là các cột mốc. Đường găng được công cụ tính toán ra và được hiển thị bằng đường đậm. Ngày bắt đầu và thời hạn cho từng nhiệm vụ cũng được xác định. Người lập kế hoạch còn xác định ra lực lượng lao động sẽ làm việc cho từng nhiệm vụ và chi phí cho các tài nguyên.

Một khi dữ liệu đã được đưa vào dưới dạng biểu đồ mạng như trong hình 4.3, MacProject II sẽ tự động sinh ra thông tin sau:

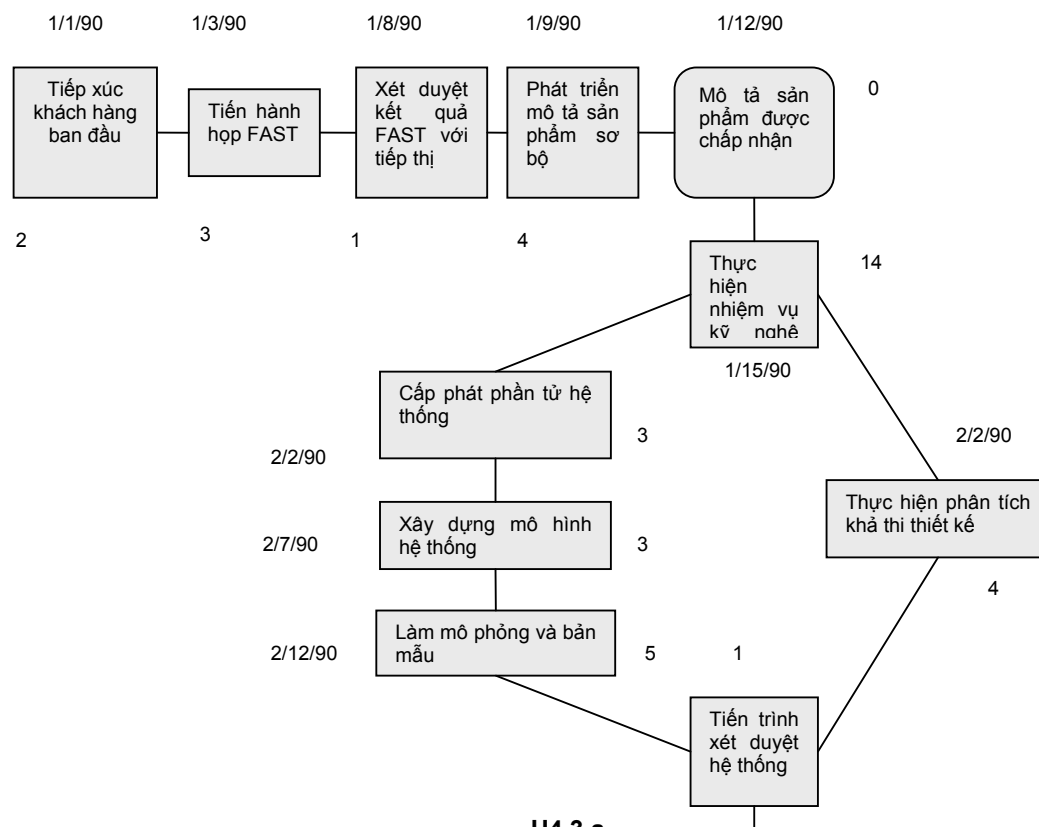
- (1) Sơ đồ thời gian mô tả các nhiệm vụ như một hàm của thời gian
- (2) Bảng cấp phát các tài nguyên chứa thời gian bắt đầu và kết thúc nhiệm vụ, công sức cần dùng và các dữ liệu khác giúp cho người quản trị dự án theo dõi tiến độ khi dự án bắt đầu.

IV.2.6 Theo dõi và kiểm soát dự án

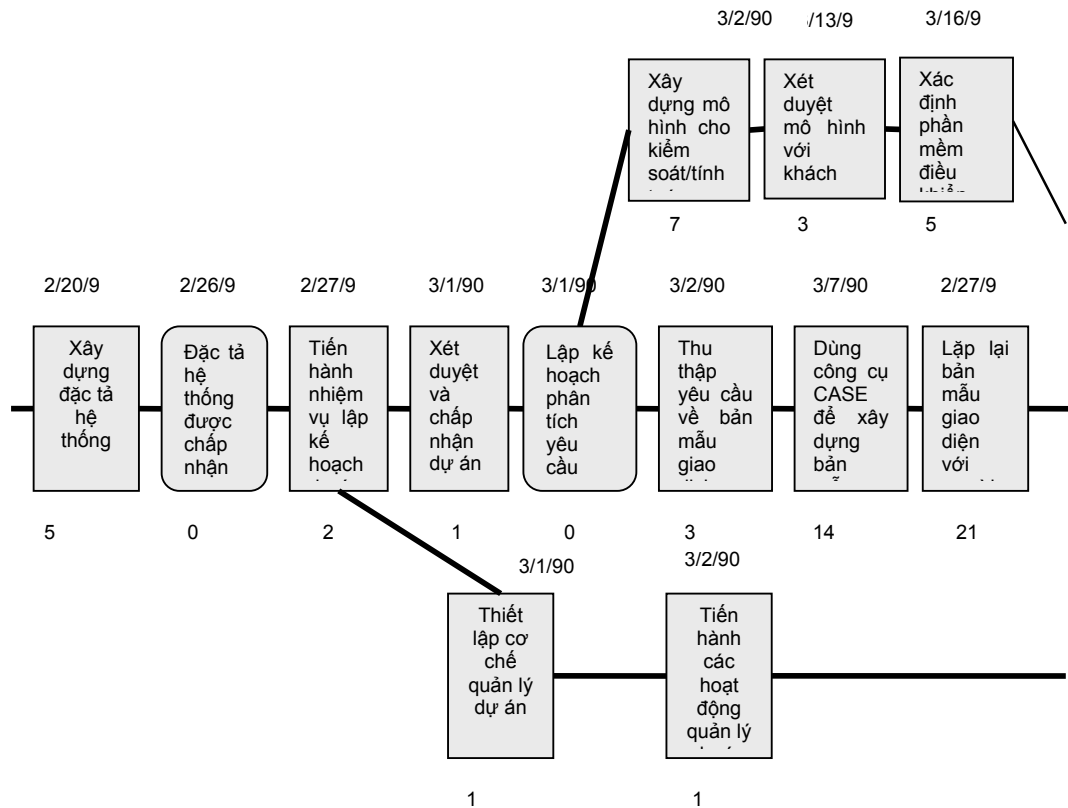
Một ngày trượt so với lịch thì hiếm khi làm hỏng cả dự án, nhưng qua toàn bộ chiều dài dự án những chậm trễ nhỏ có thể gây ra vấn đề lớn.

Vai trò của quản lý dự án là theo dõi và kiểm soát dự án phần mềm một khi nó đang tiến hành. Việc theo dõi có thể được thực hiện theo một số cách khác nhau:

- Tiến hành các cuộc họp xem xét hiện trạng dự án đều đặn trong đó mỗi thành viên nhóm đều báo cáo lại tiến độ và vấn đề gặp phải.
- Đánh giá kết quả của mọi cuộc họp xem xét được tiến hành qua toàn bộ tiến trình công nghệ.
- Xác định xem liệu các mốc dự án chính thức (hình chữ nhật tròn góc) đã được hoàn thành trước thời hạn đã lên lịch hay chưa.
- So sánh ngày bắt đầu thực tế với ngày bắt đầu theo kế hoạch cho từng nhiệm vụ dự án được liệt kê trong bảng tài nguyên.
- Họp không chính thức với người phát triển để thu được những đánh giá chủ quan của họ về tiến độ ngày tháng và vấn đề sắp xảy ra.

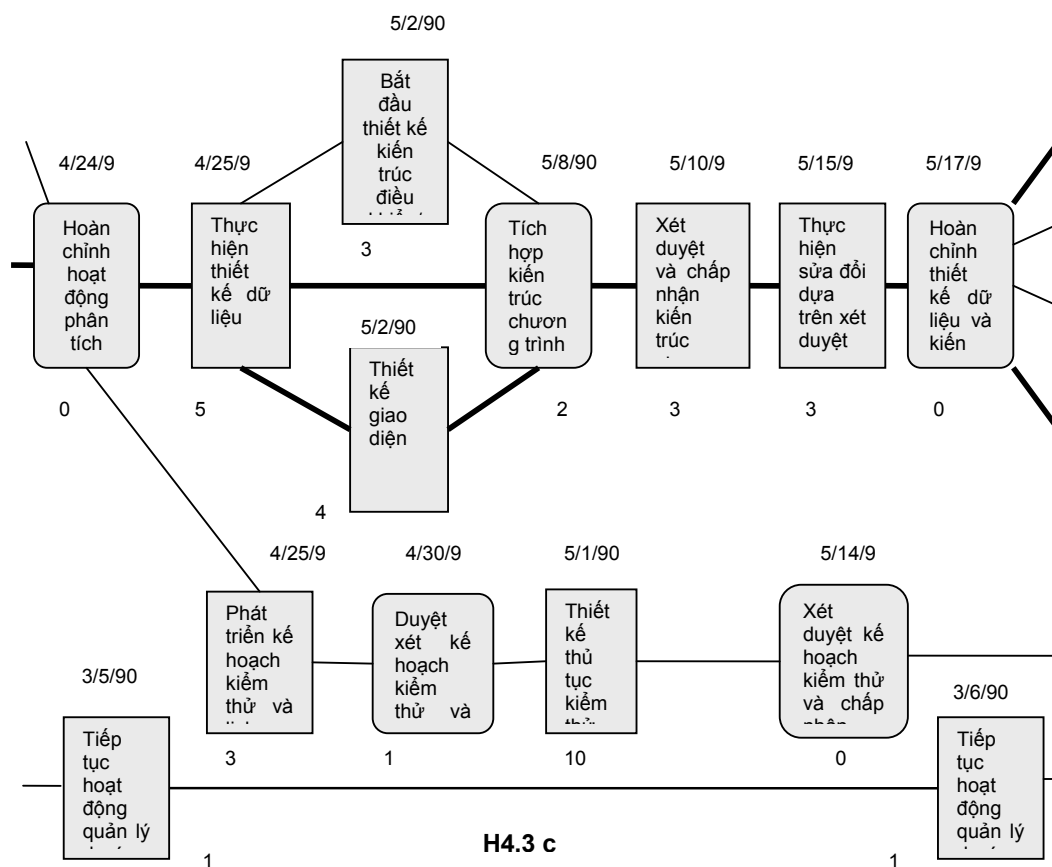


H4.3 a

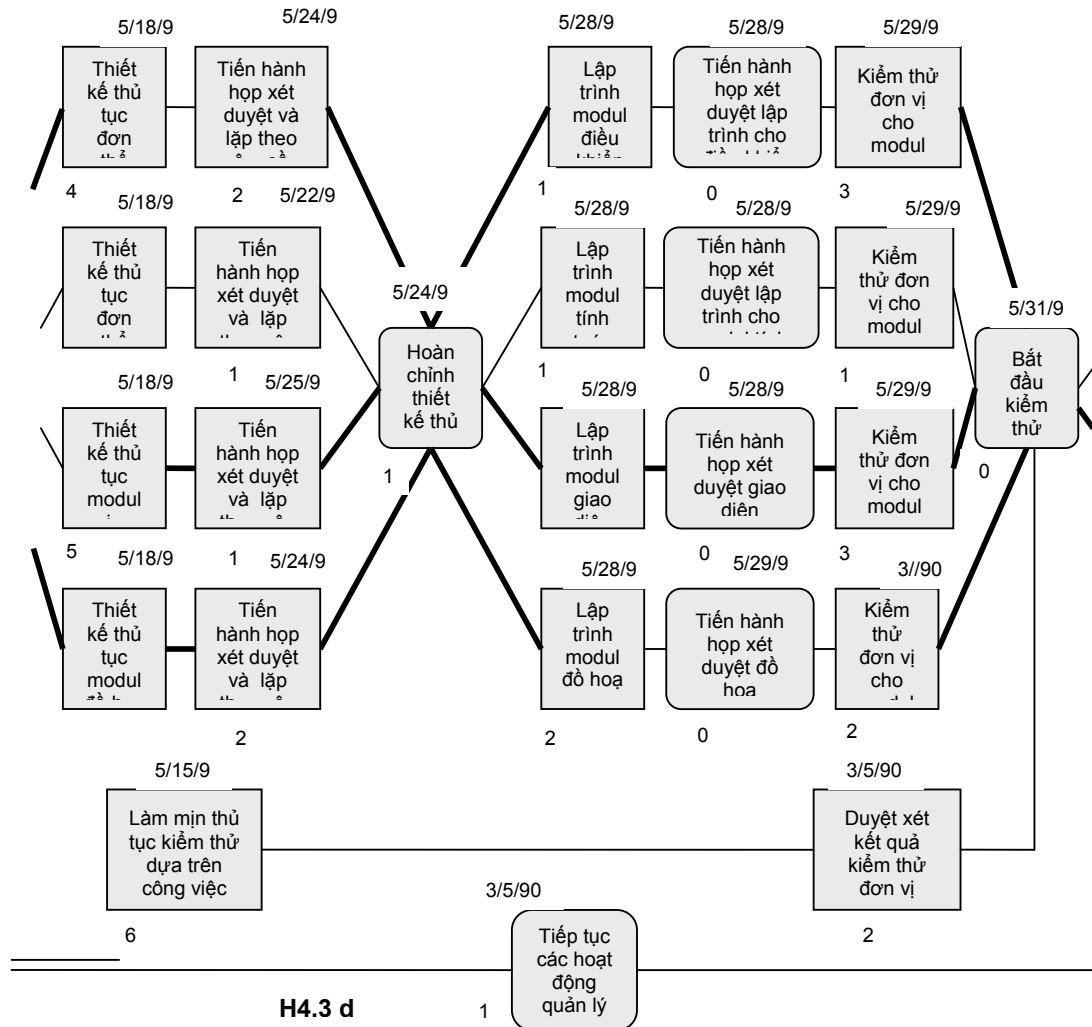


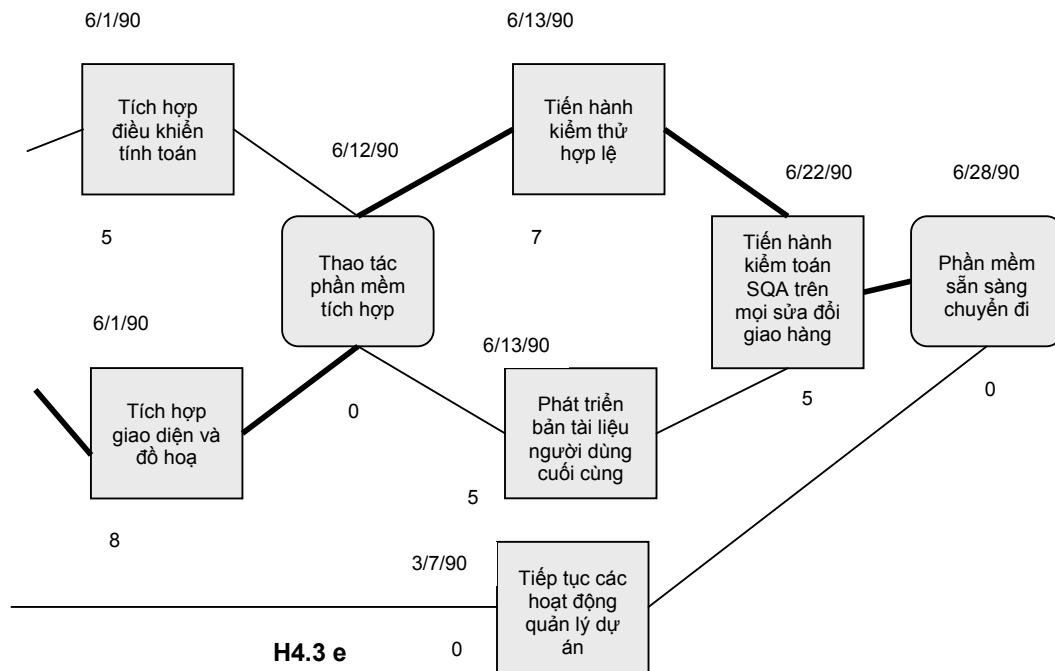
H4.3 b

Trong thực tế tất cả các kỹ thuật theo dõi này đều được các nhà quản lý có kinh nghiệm sử dụng. Việc kiểm soát được người quản lý dự án phần mềm sử dụng để quản trị tài nguyên dự án, giải quyết vấn đề và chỉ huy nhân viên dự án. Việc kiểm soát là nhẹ nhàng khi dự án theo đúng lịch và trong phạm vi ngân sách, các cuộc họp cho thấy có tiến bộ thực sự và đạt được tới các cột mốc. Nhưng khi vấn đề xuất hiện, người quản lý dự án phải thực thi kiểm soát để điều tiết chúng một cách nhanh chóng nhất. Sau khi vấn đề đã được chuẩn đoán, có thể tập trung thêm tài nguyên phụ vào miền có vấn đề, có thể bố trí lại nhân viên hoặc có thể chỉnh lại lịch biểu.



3/5/90

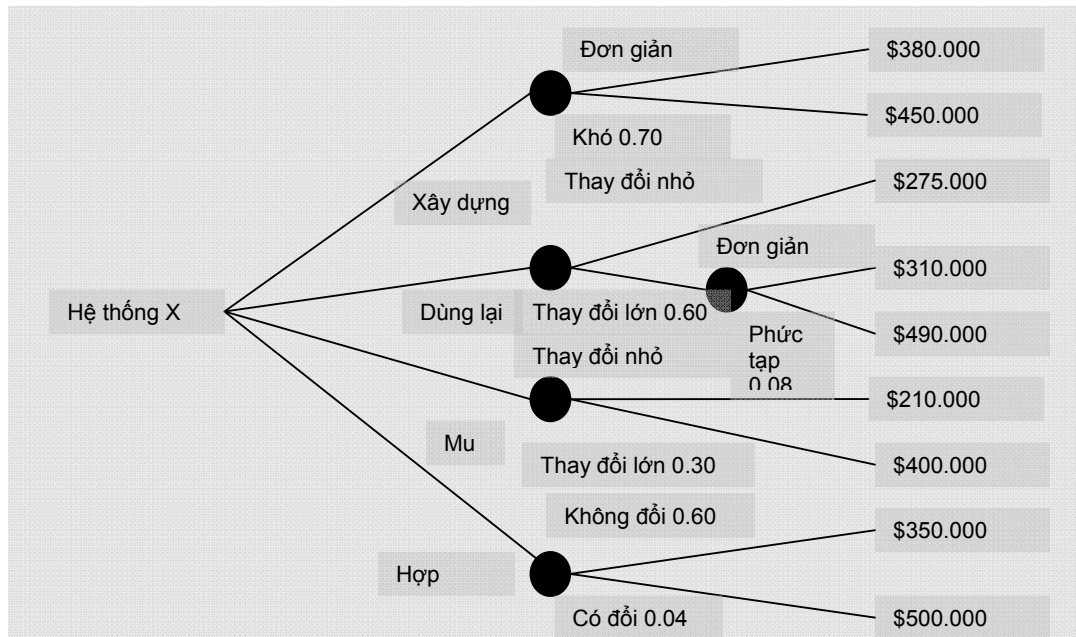




H4.3 Mạng nhiệm vụ điển hình (a-e)

IV.3 Tái kỹ nghệ phần mềm

Trong những năm 1990, nhiều chương trình ngày càng trở nên khó bảo trì hơn, miếng vá nọ đắp lên miếng vá kia, tạo ra những chương trình làm việc không hiệu quả, thường hay lỗi và không đáp ứng được nhu cầu người dùng. Trong thực tế việc bảo trì những chương trình đã qua nhiều năm sử dụng trở nên quá tốn kém đối với nhiều hệ thông tin, các tổ chức chế tạo và công nghệ.



H4.4 Cây quyết định để hỗ trợ cho việc quyết định làm hay mua

Trong thực tế, việc tái kỹ nghệ (làm lại và nâng cấp dựa trên những thứ đã có) có thể là phương án có chi phí thấp hơn so với việc bảo trì phần mềm. Sau đây là một số bước cần lưu ý trong việc tái kỹ nghệ phần mềm:

- (1) Chọn những chương trình hay được dùng và có thể được dùng trong 5 đến 10 năm nữa.
- (2) Ước lượng chi phí hàng năm để bảo trì những chương trình đã được chọn trong bước 1. Chi phí bảo trì nên chức cả việc sửa lỗi, làm thích ứng với môi trường và nâng cao chức năng.
- (3) Sắp thứ tự ưu tiên các chương trình đã được lựa chọn trong bước 1 theo tầm quan trọng và chi phí bảo trì.
- (4) Ước lượng chi phí tái kỹ nghệ trong bước 1 [vì đã có một bản mẫu là chương trình cũ nên chi phí tái kỹ nghệ có thể ít hơn]. Ước lượng chi phí hàng năm để bảo trì những chương trình đã tái kỹ nghệ này.

- (5) Với mỗi chương trình đã chọn, so sánh chi phí bảo trì nó với chi phí tái kỹ nghệ số.
- (6) Tính thời gian cần để thu lại đầu tư và tái kỹ nghệ
- (7) Xem xét những vấn đề vô hình như đáp ứng được nâng, độ tin cậy hệ thống tốt hơn, hiệu suất hệ thống cao hơn và giao diện người dùng được cải tiến.
- (8) Thu được sự chấp thuận của cấp quản lý để bắt đầu việc tái kỹ nghệ cho một chương trình.
- (9) Dùng các bài học thu được từ nỗ lực tái kỹ nghệ đầu tiên cho việc tái kỹ nghệ các chương trình khác.

Đến cuối các những năm 1990 rất có thể các công cụ CASE nâng cao việc tái kỹ nghệ sẽ làm giảm khá lớn chi phí liên quan tới hoạt động này.

IV.4 Lập kế hoạch tổ chức nhân sự

Gần như có nhiều *cấu trúc tổ chức con người* cho phát triển phần mềm cũng như các tổ chức phát triển phần mềm. Dù tốt hay xấu hơn, cấu trúc tổ chức vốn có cũng không thể dễ dàng thay đổi được. Mỗi bận tâm tới những ảnh hưởng thực tế và chính trị của việc thay đổi tổ chức không phải là phạm vi trách nhiệm của người lập dự án phần mềm. Tuy nhiên tổ chức của những người trực tiếp tham dự vào một dự án phần mềm có thể được xem xét tới vào lúc này.

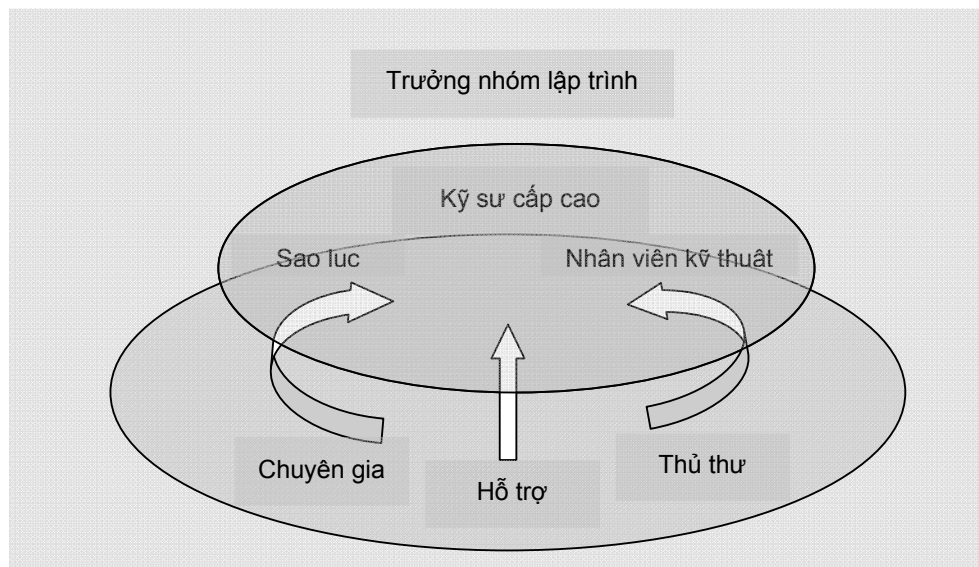
Các mô hình sau đây giúp tổ chức tài nguyên nhân lực cho dự án đòi hỏi n người làm m việc trong k năm:

- n người được trao cho m nhiệm vụ chức năng khác nhau ($n > m$): có tương đối ít công việc cần tổ hợp; việc điều phối là trách nhiệm của người quản lý phần mềm, người này có thể còn có một số các dự án khác đang tham dự.
- n người được trao m nhiệm vụ chức năng khác nhau ($n > m$) sao cho các nhóm không chính thức được thiết lập: có thể chỉ định một người lãnh đạo nhóm cơ sở; việc điều phối trong các nhóm là trách nhiệm của người quản lý phần mềm.

- n người được tổ chức thành t nhóm: mỗi nhóm đều được trao đổi một hay nhiều nhiệm vụ chức năng; mỗi nhóm có một tổ chức riêng; việc điều phối được cả người quản lý phần mềm và trưởng nhóm điều khiển.

Mặc dầu có nhiều tranh cãi đối với từng cách tiếp cận trên, đang có nhiều bằng chứng chỉ ra rằng tổ chức nhóm hình thức (mô hình 3) là hiệu quả nhất.

Tổ chức của nhóm phát triển phần mềm được minh họa trong Hình 4.5. Hạt nhân của nhóm bao gồm một *kỹ sư cao cấp* (trưởng nhóm lập trình) là người lập kế hoạch, điều phối và xét duyệt mọi hoạt động kỹ thuật của nhóm; *nhân viên kỹ thuật* (thông thường từ 2 đến 5 người) là những người tiến hành hoạt động phân tích và phát triển; và *kỹ sư dự phòng* là người hỗ trợ cho kỹ sư cao cấp trong các hoạt động và có thể thay thế kỹ sư cao cấp để tránh thiệt hại tối thiểu cho sự liên tục của dự án.



H4.5 Nhóm phát triển phần mềm

Nhóm phát triển phần mềm có thể có một hay nhiều chuyên gia phục vụ (chuyên gia viễn thông, chuyên gia về cơ sở dữ liệu...), có các nhân viên hỗ trợ (như người viết tài liệu kỹ thuật, thư ký riêng...). Thủ thư phục vụ cho nhiều nhóm, giúp duy trì và kiểm soát mọi phần tử cấu hình phần mềm như tài liệu, bản in chương trình gốc, dữ liệu, thiết bị từ, giúp thu thập và định dạng dữ liệu hiệu năng phần mềm, lập danh mục và chỉ số các modul phần mềm dùng lại được, trợ giúp cho các nhóm trong nghiên cứu,

ước lượng chuẩn bị tư liệu. Tầm quan trọng của các thủ thư không nên bị coi thường, thủ thư hành động như người kiểm soát, người điều phối và về tiềm năng sẽ là người ước lượng về cấu hình phần mềm.

Trong mục IV.2.1 chúng ta đã thảo luận mối quan hệ giữa con người – công việc. Sự liên lạc (như lập kế hoạch, phân tích, xét duyệt) tăng lên khi có nhiều nhân sự, điều cố hữu trong bất kỳ dự án nào, dường như cản trở hiệu năng phát triển khi áp dụng mô hình nhóm phần mềm. Tuy nhiên số phần trăm nỗ lực có ý nghĩa trong tiến trình kỹ nghệ phần mềm phải được dành cho liên lạc. Tổ chức nhóm rút gọn bớt nỗ lực liên lạc và khử bỏ nhiều hiểu lầm tốn thời gian thường xuất hiện khi người ta làm việc độc lập. Bên cạnh đó, tổ chức nhóm còn cổ vũ cho việc xét duyệt, do đó làm tăng chất lượng sản phẩm phần mềm.

Mặc dầu việc dùng nhóm phát triển phần mềm không phải bao giờ cũng thực tế (như với dự án nhỏ 1 người hay với những nỗ lực được chia nhỏ cao độ), những khái niệm nền tảng vẫn đúng đắn và nên được áp dụng.

IV.5 Kế hoạch dự án phần mềm

Mỗi bước trong tiến trình kỹ nghệ phần mềm cần phải tạo ra một sự bàn giao có thể được xét duyệt và điều đó có thể xem như cơ sở cho bước tiếp sau. *Kế hoạch dự án phần mềm* được tạo ra tại đỉnh cao của các nhiệm vụ lập kế hoạch. Nó cung cấp chi phí vạch ranh giới và thông tin lịch biểu sẽ được dùng trong suốt tiến trình kỹ nghệ phần mềm.

Kế hoạch dự án phần mềm là một tài liệu tương đối ngắn gọn (trang bên). Trước hết nó thông báo về phạm vi và tài nguyên cho cấp quản lý phần mềm, nhân viên kỹ thuật và khách hàng. Thứ hai là xác định các rủi ro, gợi ý các kỹ thuật để khắc phục rủi ro. Thứ ba là xác định chi phí và lịch biểu cho việc xét duyệt quản lý. Cuối cùng là cung cấp một cách tiếp cận tổng thể tới việc phát triển phần mềm cho tất cả mọi người liên quan tới dự án.

Đại cương về bản kế hoạch này được trình bày trong Bảng 4.2

I. Giới thiệu

A. Phạm vi mục tiêu của dự án
B. Mục tiêu dự án
1. Mục tiêu
2. Chức năng chính
3. Vấn đề hiệu năng
4. Ràng buộc quản lý và kỹ thuật
II. Ước lượng dự án
A. Dữ liệu lịch sử được dùng cho ước lượng
B. Kỹ thuật ước lượng
C. Ước lượng
III. Rủi ro dự án
A. Phân tích rủi ro
1. Xác định
2. Ước lượng rủi ro
3. Đánh giá
B. Quản lý rủi ro
1. Tùy chọn rủi ro
2. Thủ tục điều phối rủi ro
IV. Lập lịch
A. Cấu trúc phân việc dự án
B. Mạng nhiệm vụ
C. Sơ đồ đường thời gian
D. Bảng tài nguyên
V. Tài nguyên dự án
A. Con người

B. Phần cứng và phần mềm
C. Tài nguyên đặc biệt
VI. Tổ chức đội ngũ
A. Cấu trúc nhóm (nếu áp dụng được)
B. Làm báo cáo quản lý
VII. Cơ chế theo dõi và kiểm soát
VIII. Phụ lục

Bảng 4.2 Kế hoạch dự án phần mềm**Tóm tắt**

Lập kế hoạch là một hoạt động quản lý dự án phần mềm tổ hợp cả các kỹ thuật đo và phương pháp ước lượng đã được thảo luận trong các chương trước, với việc phân tích rủi ro, lập lịch và các hoạt động ra quyết định khác.

Rủi ro là một phần cố hữu của mọi dự án phần mềm nên cần phải được phân tích và quản lý. Việc phân tích rủi ro bắt đầu với việc xác định và tiếp đó là dự phòng rủi ro. Một khi đã biết được khả năng xuất hiện, tác động của nó thì các hoạt động quản lý và điều phối rủi ro có thể được tiến hành để giúp kiểm soát các rủi ro mà thực tại có thể xuất hiện.

Dùng nỗ lực con người đã được ước lượng cho một dự án, việc lập lịch bắt đầu với việc tạo ra một mạng biểu thị cho từng nhiệm vụ phát triển, sự phụ thuộc của nó với các nhiệm vụ khác, và thời hạn được dự phòng của nó. Mạng được dùng để tính đường găng dự án, biểu đồ đường thời gian và các thông tin khác về dự án. Dùng sơ đồ này như một hướng dẫn người quản lý dự án có thể theo dõi và kiểm soát từng bước trong tiến trình kỹ nghệ phần mềm.

Trong một số trường hợp, có thể mua bộ trình phần mềm có sẵn, hay hợp đồng thầu việc xây dựng nó, thay vì tự xây dựng nó. Quyết định mua hay làm có thể dựa trên việc dùng một tập các hướng dẫn đơn giản có bổ sung thêm kỹ thuật ra quyết định thống kê.

Kế hoạch dự án phần mềm tổ hợp thông tin được sinh ra như kết quả của tất cả các hoạt động lập kế hoạch và ước lượng. Nó cung cấp một bản lộ trình cho việc quản lý dự án.

CHƯƠNG V

HỆ THỐNG THỜI GIAN THỰC

Giống như bất kỳ hệ thống dự trên máy tính nào, hệ thống thời gian thực phải tích hợp với các yếu tố phần cứng, phần mềm, con người và cơ sở dữ liệu.

Thiết kế các hệ thống tính toán thời gian thực là nhiệm vụ phức tạp và có tính thách thức lớn nhất mà người kỹ sư phần mềm có thể tiến hành. Vấn đề đối với hệ thống thời gian thực là cấp phát đúng (vấn đề cấp phát liên quan tới *phần cứng* cho các hệ thống thời gian thực nằm ngoài phạm vi của giáo trình này). Phần mềm thời gian thực đi đôi chặt chẽ với thế giới bên ngoài, nó phải vận hành dưới những ràng buộc hiệu năng chặt chẽ nên việc thiết kế phần mềm thường bị chi phối bởi phần cứng cũng như kiến trúc phần mềm, các đặc trưng của hệ điều hành, cũng như các yêu cầu ứng dụng, cũng như các vấn đề thiết kế.

Trong chương này chúng ta sẽ xem xét phần mềm thời gian thực và thảo luận về một số kỹ năng cần để xây dựng nó.

V.1 Hệ thống thời gian thực

Hệ thống thời gian thực sinh ra một hành động nào đó để đáp ứng lại các sự kiện bên ngoài. Để hoàn thành chức năng này, chúng thực hiện việc thu thập và kiểm soát dữ liệu tốc độ cao trong những ràng buộc thời gian và độ tin cậy nghiêm ngặt. Vì những ràng buộc này là nghiêm ngặt nên các hệ thống thời gian thực thường chuyên dụng cho một ứng dụng.

Trong nhiều năm, khách hàng chính của các hệ thống thời gian thực là giới quân sự. Tuy nhiên ngày nay, hệ thống này có thể có cho phần lớn các công ty muốn có hệ thống thời gian thực đối với những ứng dụng đa dạng như điều khiển tiến trình, tự động hoá công nghiệp, nghiên cứu y học và khoa học, đồ hoạ máy tính, truyền thông mạng cục bộ và mạng diện rộng, hệ thống hành không, kiểm thử có máy tính trợ giúp và rất nhiều thiết bị công nghiệp.

V.1.1 Vấn đề tích hợp và hiệu năng

Hệ thống thời gian thực đặt ra cho người kỹ sư hệ thống những quyết định phần mềm và phần cứng khó khăn. Một khi phần tử phần mềm đã được cấp phát, các yêu cầu phần mềm chi tiết đã được thiết lập thì phải xây dựng thiết kế phần mềm nền tảng. Một trong những mối quan tâm thiết kế thời gian thực là sự điều phối giữa các nhiệm vụ thời gian thực, xử lý ngắt hệ thống, xử lý vào/ra để đảm bảo rằng không dữ liệu nào bị mất, xác định các ràng buộc thời gian bên trong và bên ngoài của hệ thống, và đảm bảo độ chính xác của cơ sở dữ liệu của nó.

Mối quan tâm thiết kế thời gian thực cho phần mềm phải được áp dụng trong ngữ cảnh hiệu năng hệ thống. Trong phần lớn các trường hợp, hiệu năng của hệ thống thời gian thực được đo theo những đặc trưng liên quan tới thời gian (nhưng cách đo khác như độ dung sai cũng có thể được dùng tới).

Một số hệ thống thời gian thực được thiết kế cho những ứng dụng trong đó chỉ có thời gian đáp ứng hay tỷ lệ truyền dữ liệu là cấp bách. Những ứng dụng thời gian thực khác đòi hỏi việc tối ưu hoá cả hai tham biến dưới điều kiện tải cao điểm. Hơn nữa, các hệ thống thời gian thực phải xử lý mức tải cao điểm trong khi thực hiện một số nhiệm vụ đồng thời.

Hiệu năng của hệ thống thời gian thực được xác định chủ yếu bởi thời gian đáp ứng hệ thống và tỷ lệ truyền dữ liệu nên điều quan trọng là phải hiểu hai tham biến này. *Thời gian đáp ứng hệ thống* là thời gian trong đó hệ thống phải phát hiện một sự kiện bên trong hay bên ngoài và đáp ứng lại bằng hành động. Thông thường, việc phát hiện hành động và phát sinh là đáp ứng đơn giản. Đó là việc xử lý thông tin về sự kiện để xác định những đáp ứng thích hợp có thể bao gồm những thuật toán phức tạp, tốn thời gian.

Trong số những tham biến mấu chốt ảnh hưởng tới thời gian đáp ứng có *chuyển hoàn cảnh và trễ ngắt*. Chuyển hoàn cảnh bao gồm thời gian và tổng phí để chuyển giữa các nhiệm vụ, còn trễ ngắt là thời gian trễ trước khi việc chuyển thực tế xảy ra. Các tham biến khác ảnh hưởng tới thời gian đáp ứng là tốc độ tính toán và việc thâm nhập bộ nhớ ngoài.

Tỷ lệ truyền dữ liệu chỉ ra dữ liệu phải được truyền vào hoặc ra hệ thống nhanh thế nào, qua đường tuần tự hay song song cũng như qua dạng tương tự hay số. Hiệu năng thiết bị vào/ra, độ trễ bus, kích cỡ bộ đệm, hiệu năng đĩa, và số đồng các nhân tố khác, mặc dầu quan trọng nhưng cũng chỉ là một phần trong thiết kế hệ thống thời gian thực. Do vậy các đặc tả hiệu năng phần cứng thường được đo một cách cô lập và thường có ít giá trị trong việc xác định hiệu năng thời gian thực tổng thể.

Hệ thống thời gian thực thường đòi hỏi xử lý luồng dữ liệu vào liên tục. Thiết kế phải đảm bảo rằng dữ liệu không bị bỏ lỡ. Bên cạnh đó, hệ thống thời gian thực phải đáp ứng cho các sự kiện không đồng bộ. Do đó, dãy và khối lượng dữ liệu vào không thể nào dễ dàng dự đoán trước được.

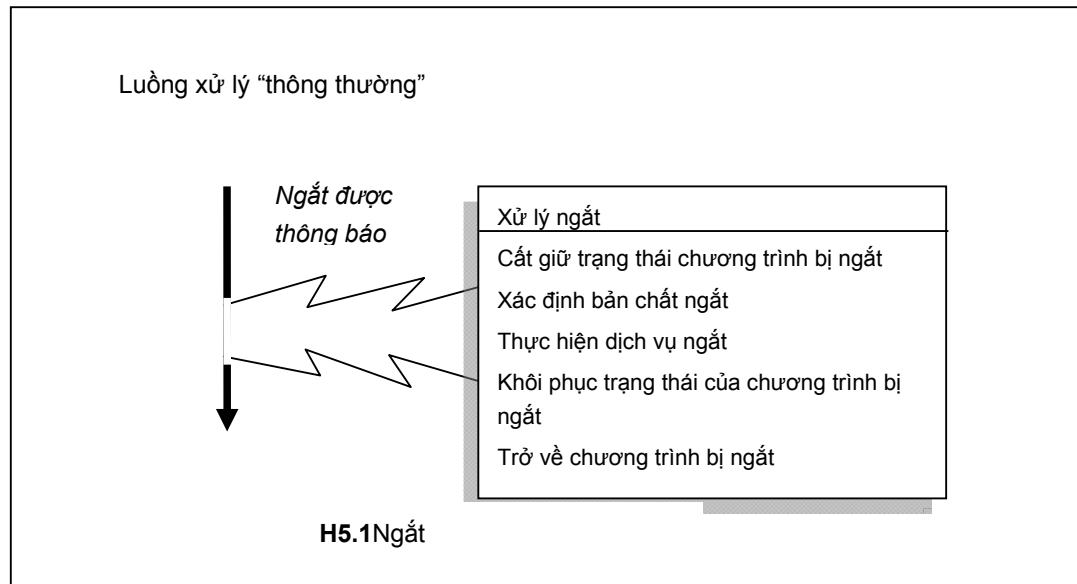
Các hệ thống thời gian thực có yêu cầu đặc biệt về độ tin cậy, việc chạy lại và phục hồi lỗi. Việc mất điều khiển hay kiểm soát là không thể dung thứ được trong nhiều hoàn cảnh (như hệ thống kiểm soát không lưu). Kết quả là các hệ thống thời gian thực phải chứa *cơ chế cho chạy lại* và khôi phục lỗi và thường có độ dư thừa để đảm bảo sao lưu.

V.1.2 Xử lý ngắt

Một đặc trưng hay dùng để phân biệt các hệ thống thời gian thực với các kiểu hệ thống khác việc *Xử lý ngắt*. Hệ thống thời gian thực phải đáp ứng với các kích thích bên ngoài – các ngắt – trong một khuôn khổ thời gian do thế giới bên ngoài ấn định. Bởi vì có nhiều kích thích thường hiện hữu nên phải thiết lập các ngắt ưu tiên. Nói cách khác, nhiệm vụ quan trọng nhất bao giờ cũng phải được giải quyết trong những ràng buộc thời gian định sẵn bất kể tới các sự kiện khác.

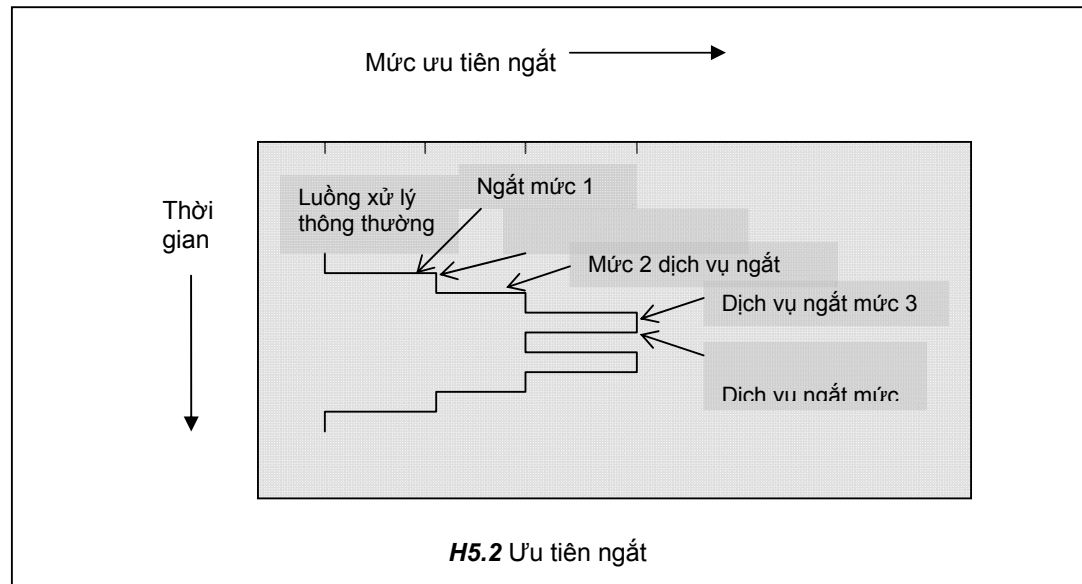
Xử lý ngắt bắt buộc không chỉ lưu trữ thông tin để cho máy tính có thể chạy lại đúng đắn nhiệm vụ đã bị ngắt mà còn phải tránh việc chết tắc và các chu trình vô hạn. Cách tiếp cận tổng thể tới xử lý ngắt được minh hoạ trong hình 5.1. Luồng xử lý thông tin thường bị “ngắt” bởi một *sự kiện* do phần cứng bộ xử lý phát hiện ra. Một *sự kiện* là bất kỳ sự xuất hiện nào đòi hỏi có việc phục vụ ngay lập tức và có thể được sinh ra hoặc bởi phần cứng hoặc phần mềm. Trạng thái của chương trình bị ngắt được cất giữ (tất cả nội dung thanh ghi, các khối điều khiển...) và điều khiển được truyền cho chương trình làm dịch vụ ngắt,

tức là nhảy đến một phần mềm thích hợp để xử lý ngắt. Khi hoàn thành xong dịch vụ ngắt thì trạng thái của máy lại được khôi phục và luồng xử lý thông thường được tiếp tục.



H5.1 Ngắt

Trong nhiều tình huống bản thân dịch vụ ngắt cho một sự kiện có thể lại bị ngắt bởi một sự kiện khác, có số ưu tiên cao hơn. Mức ưu tiên ngắt có thể được thiết lập. Nếu một tiến trình có số ưu tiên thấp ngẫu nhiên được phép ngắt một tiến trình có số ưu tiên cao thì có thể khó mà cho chạy lại các tiến trình theo đúng trật tự và có thể làm phát sinh chu trình vô hạn.



Để xử lý ngắt và vẫn đáp ứng được cho các ràng buộc thời gian hệ thống, nhiều hệ điều hành thời gian thực thực hiện tính toán đồng bộ để xác định xem mục tiêu hệ thống đã đạt được chưa. Những tính toán đồng bộ này đều dựa trên tần số trung bình số lần xuất hiện của sự kiện, khối lượng thời gian cần để phục vụ nó (nếu chúng ta có thể được phục vụ), hay có thể ngắt chúng và tạm thời ngăn cản chúng được phục vụ.

Nếu các tính toán đồng bộ chỉ ra rằng không thể nào xử lý được các sự kiện có thể được xuất hiện trong hệ thống mà vẫn đáp ứng các ràng buộc thời gian thì hệ thống phải quyết định theo sơ đồ hành động. Một sơ đồ có thể bao gồm việc đặt bộ đệm dữ liệu để cho nó có thể được xử lý nhanh chóng khi hệ thống sẵn sàng.

V.1.3 Cơ sở dữ liệu thời gian thực

Giống như nhiều hệ thống xử lý dữ liệu, hệ thống thời gian thực thường đi đôi với chức năng quản lý dữ liệu. Tuy nhiên, *cơ sở dữ liệu phân tán* là cách tiếp cận thích hợp trong hệ thống thời gian thực vì đa nhiệm là thông dụng và dữ liệu thường được xử lý song song. Nếu cơ sở dữ liệu là phân bố thì từng nhiệm vụ có thể thâm nhập và

dữ liệu của chúng nhanh hơn và tin cậy hơn, và với ít tắc nghẽn hơn là cơ sở dữ liệu tập trung. Với việc sử dụng cơ sở dữ liệu phân bố thì từng nhiệm vụ có thể vào dữ liệu của chúng nhanh hơn, tin cậy hơn và ít tắc nghẽn hơn. Việc dùng cơ sở dữ liệu phân tán cho các ứng dụng thời gian thực phân chia lưu thông vào/ra và làm ngắn đi hàng nhiệm vụ đang chờ để thâm nhập vào cơ sở dữ liệu. Hơn nữa một sai sót của một cơ sở dữ liệu sẽ hiếm khi gây hỏng hóc của toàn bộ hệ thống, nếu sự dư thừa được xây dựng sẵn.

Tính hiệu quả hiệu năng đạt được thông qua việc dùng cơ sở dữ liệu phân tán phải xứng với các vấn đề tiềm năng có liên quan tới việc phân hoạch dữ liệu và tái tạo. Mặc dầu việc dư thừa dữ liệu làm tăng thời gian đáp ứng vì cung cấp nhiều nguồn thông tin, các yêu cầu tái tạo cho các tệp phân tán cũng tạo ra vấn đề tổng phí và tất cả các bản sao tệp cần phải được cập nhật. Bên cạnh đó, việc dùng các cơ sở dữ liệu phân tán đòi hỏi việc điều khiển tương tranh. Điều khiển tương tranh bao gồm việc đồng bộ hoá các cơ sở dữ liệu sao cho mọi bản sao đều có thông tin đồng nhất, đúng đắn tự do cho việc thâm nhập.

Cách tiếp cận qui ước tới điều khiển tương tranh dựa trên việc *khoá* và *đánh dấu thời gian*. Theo những khoảng đều đặn các nhiệm vụ sau đây được khởi đầu:

- (1) cơ sở dữ liệu được đóng lại để cho việc điều khiển tương tranh được bảo đảm, không vào ra nào được phép thực hiện
- (2) việc cập nhật xuất hiện khi được yêu cầu
- (3) cơ sở dữ liệu được mở lại
- (4) các tệp được làm hợp lệ để đảm bảo rằng tất cả mọi cập nhật đều đã được thực hiện đúng
- (5) việc cập nhật hoàn tất được thừa nhận

Mọi việc khoá đều được điều phối bằng đồng hồ chính (dấu thời gian). Vấn đề trễ hay tránh cập nhật không nhất quán làm giảm bớt việc dùng rộng rãi cơ sở dữ liệu phân bố.

Tuy nhiên một số kỹ thuật đã được phát triển để tăng tốc cho việc cập nhật và để giải quyết vấn đề tương tranh. Một trong những kỹ thuật này là giao thức *loại trừ nơi ghi*, duy trì tính nhất quán của các tệp tái tạo bằng cách cho phép chỉ riêng một nhiệm vụ ghi là được cập nhật tệp. Do vậy có thể giảm bớt tổng phí cho việc khoá và đánh dấu thời gian.

V.1.4 Hệ điều hành thời gian thực

Việc chọn hệ điều hành thời gian thực (RTOS) cho một ứng dụng cụ thể không phải là một việc dễ dàng. Một số hệ điều hành thời gian thực áp dụng được cho một phạm vi rộng các cấu hình hệ thống, trong khi các hệ khác lại ăn khớp với một bộ vi xử lý đặc biệt, bất kể tới môi trường điện tử xung quanh. RTOS đạt tới khả năng của chúng qua một tổ hợp các đặc trưng phần mềm và sự đa dạng của các khả năng vi mã được cài đặt trong phần cứng.

Ngay nay người ta hay dùng hai lớp rộng các hệ điều hành cho công việc thời gian thực:

- (1) RTOS chuyên dụng được thiết kế chuyên cho các ứng dụng thời gian thực
- (2) hệ điều hành vạn năng đã được nâng cấp để cung cấp khả năng thời gian thực

Việc dùng hệ chấp hành thời gian thực làm cho hiệu năng thời gian thực thành khả thi với hệ điều hành vạn năng. Hành động như phần mềm ứng dụng, hệ chấp hành thực hiện một số chức năng của hệ điều hành - đặc biệt đối với những chức năng ảnh hưởng tới hiệu năng thời gian thực – nhanh hơn và hiệu quả hơn các hệ điều hành vạn năng.

Tất cả các hệ điều hành đều phải có một cơ chế lập lịch ưu tiên, nhưng RTOS phải cung cấp một cơ chế để ưu tiên cho các phép ngắt có mức độ ưu tiên cao sẽ được thực hiện trước so với các ngắt có độ ưu tiên thấp hơn. Hơn nữa, vì việc ngắt xảy ra để đáp ứng cho những sự kiện không đồng bộ, không tuần hoàn cho nên chúng phải được phục vụ mà không mất thời gian trao đổi chương trình từ bộ nhớ. Kết quả là, để đảm bảo thời gian theo yêu cầu, hệ điều hành thời gian thực phải có một cơ chế khóa bộ nhớ, tức là khoá ít nhất một số chương trình trong bộ nhớ chính sao cho tránh được tổng phí trao đổi.

Để xác định đúng loại hệ điều hành thời gian thực nào thích hợp nhất với ứng dụng, Thời gian chuyển hoàn cảnh và trễ ngắt (đã trình bày ở trên) xác định khả năng xử lý ngắt, khía cạnh quan trọng nhất của hệ điều hành thời gian thực. Thời gian chuyển hoàn cảnh là thời gian hệ điều hành cần để cất giữ trạng thái của máy tính và nội dung của các thành ghi sao cho nó có thể trở về một nhiệm vụ đang xử lý sau khi hoàn thành dịch vụ ngắt.

Trễ ngắt (thời gian trễ tối đa trước khi hệ thống chuyển sang một nhiệm vụ) xuất hiện vì trong một hệ điều hành thường có các đường xử lý gắng không đồng thời vào được mà phải được hoàn tất trước khi một ngắt có thể được xử lý. Chiều dài của những đường xử lý này (số các lệnh) trước khi hệ thống có thể phục vụ một ngắt chỉ ra độ trễ thời gian lâu nhất. Trường hợp tồi nhất xuất hiện nếu một ngắt ưu tiên cao được sinh ra ngay sau khi hệ thống đi vào một đường gắng giữa một ngắt và dịch vụ ngắt. Nếu thời gian quá lâu thì hệ

thống có thể bỏ lỡ một phần dữ liệu không thể khôi phục được. Điều quan trọng là người thiết kế biết về độ trễ thời gian cho hệ thống để bù lại cho nó.

Ngoài ra, nhiều hệ điều hành đa nhiệm hay xử lý tương tranh cũng là một trong những yêu cầu khác cho hệ thống thời gian thực. Nhưng để có thể dành cho các thao tác thời gian thực, thời gian chuyển và không gian bộ nhớ cần dùng phải có tổng phí hệ thống thấp.

V.1.5 Ngôn ngữ thời gian thực

Bởi các yêu cầu đặc biệt cho hiệu năng và độ tin cậy đối với các hệ thống thời gian thực, việc lựa chọn ngôn ngữ lập trình là quan trọng. Nhiều ngôn ngữ lập trình vạn năng như C, C++, FORTRAN, MODULA-2... có thể được dùng một cách có hiệu quả cho những ứng dụng thời gian thực. Tuy nhiên, một lớp các ngôn ngữ gọi là ngôn ngữ thời gian thực (như ADA, JOVIAL, HAL/S, CILL...) lại thường được dùng trong những ứng dụng truyền thông và quân sự.

Việc tổ hợp các đặc trưng làm cho ngôn ngữ thời gian thực khác biệt với ngôn ngữ vạn năng. Những khác biệt này bao gồm khả năng đa nhiệm, các kết cấu trực tiếp cài đặt chức năng thời gian thực, và các tính năng ngôn ngữ hiện đại giúp đảm bảo tính đúng đắn của chương trình.

Một ngôn ngữ lập trình trực tiếp hỗ trợ cho đa nhiệm là quan trọng bởi vì hệ thống thời gian thực phải đáp ứng cho các sự kiện không đồng bộ xuất hiện đồng thời. Mặc dầu nhiều RTOS cung cấp khả năng đa nhiệm, phần mềm thời gian thực nhúng vẫn thường tồn tại mà không có hệ điều hành. Thay vào đó, các ứng dụng nhúng được viết trong một ngôn ngữ cung cấp sự hỗ trợ thời gian chạy đủ cho việc thực hiện chương trình thời gian thực. Hỗ trợ thời gian chạy đòi hỏi ít bộ nhớ hơn là hệ điều hành, và nó có thể được may đo theo ứng dụng, do vậy sẽ làm tăng hiệu năng.

Một hệ thống thời gian thực đã được thiết kế để thích nghi cho nhiều nhiệm vụ thì cũng phải thích nghi cho việc đồng bộ hoá giữa các nhiệm vụ. Một số ngôn ngữ lập trình trực tiếp hỗ trợ cho sự đồng bộ hoá như: SCHEDULE, SIGNAL và WAIT làm đơn giản đi rất nhiều việc dịch từ thiết kế sang mã hoá. Chỉ lệnh SCHEDULE lập lịch cho một tiến trình dựa trên thời gian hoặc sự kiện; chỉ lệnh SIGNAL và WAIT thao tác một cờ (flag) đặc biệt gọi là cờ báo hiệu *semaphore*, làm cho các nhiệm vụ tương tranh có thể được đồng bộ hoá.

V.1.6 Đồng bộ hoá và truyền thông nhiệm vụ

Hệ thống đa nhiệm phải có cơ chế cho các nhiệm vụ truyền thông tin lẫn nhau cũng như đảm bảo sự đồng bộ của chúng. Với những chức năng này, hệ điều hành và ngôn ngữ hỗ trợ thời gian chạy thông thường hay dùng cách đặt cờ báo hiệu sắp hàng, hộp thư hay hệ thống báo. Cờ báo hiệu cung cấp việc đồng bộ hoá và đặt tín hiệu nhưng không chứa thông tin. Thông báo thì cũng tương tự như cờ báo hiệu ngoài trừ rằng chúng mang thông tin có liên quan. Hộp thư, mặt khác không báo hiệu thông tin nhưng lại chứa thông tin.

Cờ báo hiệu sắp hàng là các nguyên sơ phần mềm giúp cho việc quản lý lưu thông. Chúng cung cấp một phương pháp chỉ đạo nhiều hàng đợi - chẳng hạn, hàng đợi nhiệm vụ đang chờ tài nguyên, thâm nhập cơ sở dữ liệu, các thiết bị, cũng như các hàng đợi về các tài nguyên và các thiết bị. Cờ tín hiệu đồng bộ hóa (điều phối) các nhiệm vụ đang đợi cho dù chúng đang đợi bất kỳ cái gì mà không để cho các nhiệm vụ tài nguyên can thiệp lẫn nhau.

Trong hệ thống thời gian thực, cờ báo hiệu thường được dùng để cài đặt và quản lý *hộp thư*. Hộp thư là những vị trí lưu trữ tạm thời (cũng còn được gọi là *thùng*

hay *bộ đệm thông báo*) cho các thông báo được gửi từ tiến trình nọ sang tiến trình kia. Một tiến trình tạo ra một mẫu thông tin, đặt nó vào trong hộp thư, rồi báo hiệu cho một tiến trình tiêu thụ rằng có mẫu thông tin trong hộp thư của nó đấy lấy ra mà dùng.

Một số cách tiếp cận hệ thống thời gian thực hay hệ thống hỗ trợ, khi chạy coi hộp thư như là cách hiệu quả nhất để cài đặt liên lạc giữa các tiến trình. Một số hệ điều hành thời gian thực còn cung cấp chỗ để gửi và nhận con trỏ tới dữ liệu hộp thư. Điều này loại bỏ sự cần thiết phải truyền tất cả dữ liệu – do vậy tiết kiệm thời gian và tổng phí.

Cách tiếp cận thứ ba đối với truyền thông và đồng bộ hoá trong các tiến trình là hệ thống thông báo. Với hệ thống thông báo, một tiến trình gửi một thông báo cho một tiến trình khác. Tiến trình sau này tự động được kích hoạt bằng hệ thống hỗ trợ khi chạy trên hệ điều hành để xử lý thông báo. Một hệ thống như vậy phải chịu tổng phí vì nó truyền thông tin thực tại nhưng nó đã cung cấp một sự mềm dẻo linh hoạt và dễ dùng.

V.2 Phân tích và mô phỏng hệ thống thời gian thực

Có một số thuộc tính không thể tách khỏi các yêu cầu chức năng của hệ thống thời gian thực:

- Xử lý ngắt và chuyển hoàn cảnh
- Thời gian đáp ứng
- Tỷ lệ truyền dữ liệu và hiệu suất
- Cấp phát tài nguyên và xử lý ưu tiên
- Đồng bộ hoá các nhiệm vụ và truyền thông giữa các nhiệm vụ

Mỗi thuộc tính hiệu năng này đều có thể xác định được, nhưng khó kiểm chứng lại xem liệu các phần tử có đáp ứng như đúng mong muốn hay không, tài nguyên hệ thống có đủ để thoả mãn các yêu cầu tính toán hay không, tốc độ các thuật toán xử lý có đủ hay không...

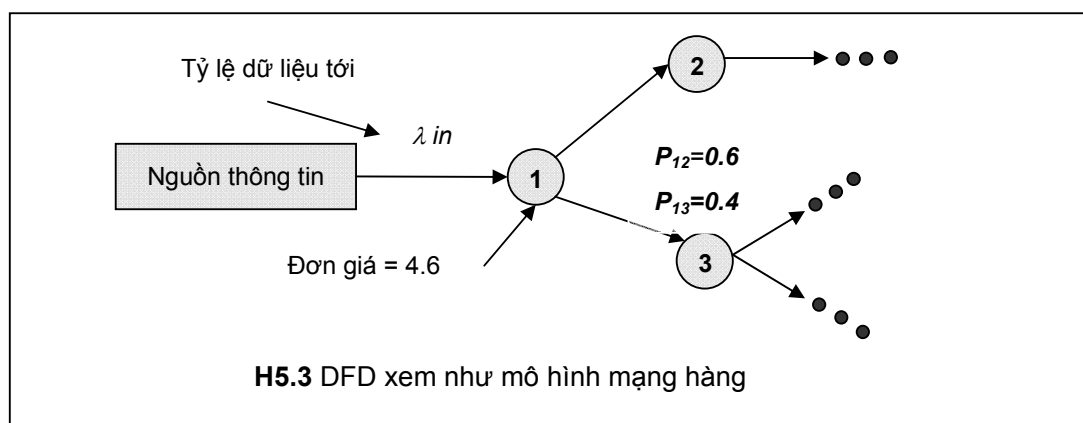
Việc phân tích hệ thống thời gian thực đòi hỏi việc mô hình hoá và mô phỏng để người kỹ sư hệ thống có thể đánh giá được các vấn đề về thời gian và kích cỡ. Đã có

rất nhiều kỹ thuật nhưng cách tiếp cận cho việc phân tích và thiết kế hệ thống thời gian thực vẫn còn trong giai đoạn non trẻ của nó.

V.2.1 Công cụ toán học cho phân tích hệ thống thời gian thực

Một tập các công cụ toán học giúp cho người kỹ sư hệ thống mô hình hoá các phần tử hệ thống thời gian thực, định giá các vấn đề thời gian và kích cỡ đã được Thomas McCabe đưa ra. Cách tiếp cận của McCabe dựa trên các kỹ thuật phân tích luồng dữ liệu giúp cho người phân tích có thể mô hình hoá cả các phần tử phần cứng và phần mềm của hệ thống thời gian thực, biểu diễn hành động theo xác suất, áp dụng các phân tích mạng, lý thuyết xếp hàng và đồ thị, mô hình toán học Markov (mô hình hàng đợi xác suất) để suy ra việc định thời gian hệ thống và định kích cỡ tài nguyên. Một tổng quan về kỹ thuật này sẽ được trình bày, sẽ cung cấp một cách tiếp cận phân tích tới kỹ nghệ hệ thống thời gian thực.

Thay vì việc dùng DFD theo cách quy ước như trong mô hình luồng dữ liệu, McCabe cho rằng các phép biến đổi (hình tròn) có thể được biểu thị như các trạng thái tiến trình của xích Markov và dữ liệu chảy qua chúng biểu thị cho các phép chuyển giữa các trạng thái tiến trình. Nhà phân tích có thể gán xác suất chuyển cho từng đường chảy dữ liệu. Tham khảo tới hình 5.3 một giá trị $0 < p_{ij} < 1.0$ có thể được xác định cho từng đường chảy, với p_{ij} biểu thị cho xác suất luồng chảy đó sẽ xuất hiện giữa tiến trình i và tiến trình j . Các tiến trình tương ứng với các biến đổi thông tin (hình tròn) trong DFD.



Mỗi tiến trình trong mô hình tựa DFD có thể được gán cho một đơn giá biểu thị thời gian thực hiện được ước lượng cần cho việc thực hiện chức năng của nó và một

giá trị vào mô tả cho các số ngắt hệ thống tương ứng với tiến trình đó. Mô hình này do vậy được phân tích bằng cách dùng một công cụ toán học sẽ tính ra:

- kỳ vọng việc dùng tới một tiến trình
- thời gian dành cho hệ thống khi tiến trình bắt đầu tại một tiến trình xác định
- tính ra toàn bộ thời gian dành cho hệ thống

Để minh họa kỹ thuật của McCabe trong thí dụ thực tế, ta xét một hệ thống đo điện tử được vẽ trong hình 5.4 một biểu đồ luồng dữ liệu dạng chuẩn nhưng căn cước luồng dữ liệu thì đã được thay bởi p_{ij} . Mô hình mạng hàng đợi được suy dẫn từ DFD và được vẽ trên hình 5.6. Giá trị λ_{ij} tương ứng với tỷ lệ tới (theo giây) tại mỗi tiến trình. Tùy theo kiểu hàng đợi gặp phải, người phân tích phải xác định thông tin thống kê như tỷ lệ dịch vụ trung bình (theo giây), phương sai tỷ lệ dịch vụ công bằng, phương sai của tỷ lệ tới...

Tỷ lệ tới cho từng tiến trình được xác định bằng cách dùng xác suất đường chảy p_{ij} và tỷ lệ tới trong hệ thống λ_{in} . Một tập các phương trình cân bằng luồng chảy được suy dẫn và giải quyết đồng thời để tính cho luồng chảy qua từng tiến trình. Với thí dụ được vẽ trong hình 5.6, phương trình cân bằng sau đây là kết quả:

$$\lambda_1 = \lambda_{in} + \lambda_4$$

$$\lambda_2 = p_{12}\lambda_1$$

$$\lambda_3 = p_{13}\lambda_1 + p_{23}\lambda_2$$

$$\lambda_4 = p_{64}\lambda_6$$

$$\lambda_5 = p_{25}\lambda_2 + \lambda_3$$

$$\lambda_6 = \lambda_5$$

$$\lambda_7 = p_{67}\lambda_6$$

Với p_{ij} như đã cho và tỷ lệ với $\lambda_{in} = 5$ lần tới trong một giây, có thể giải các phương trình trên và nhận được:

$$\lambda_1 = 8.3$$

$$\lambda_2 = 5.8$$

$$\lambda_3 = 8.3$$

$$\lambda_4 = 3.3$$

$$\lambda_5 = 8.3$$

$$\lambda_6 = 8.3$$

$$\lambda_7 = 5.0$$

Một khi các tỷ lệ tới đã được tính ra thì có thể dùng lý thuyết hàng đợi chuẩn để tính thời gian hệ thống. Mỗi hệ con (hàng đợi Q và bộ phục vụ S) có thể được tính bằng cách dùng các công thức tương ứng với kiểu xếp hàng. Với hàng đợi ($m/m/1$):

Việc sử dụng: $\rho = \lambda / \mu$

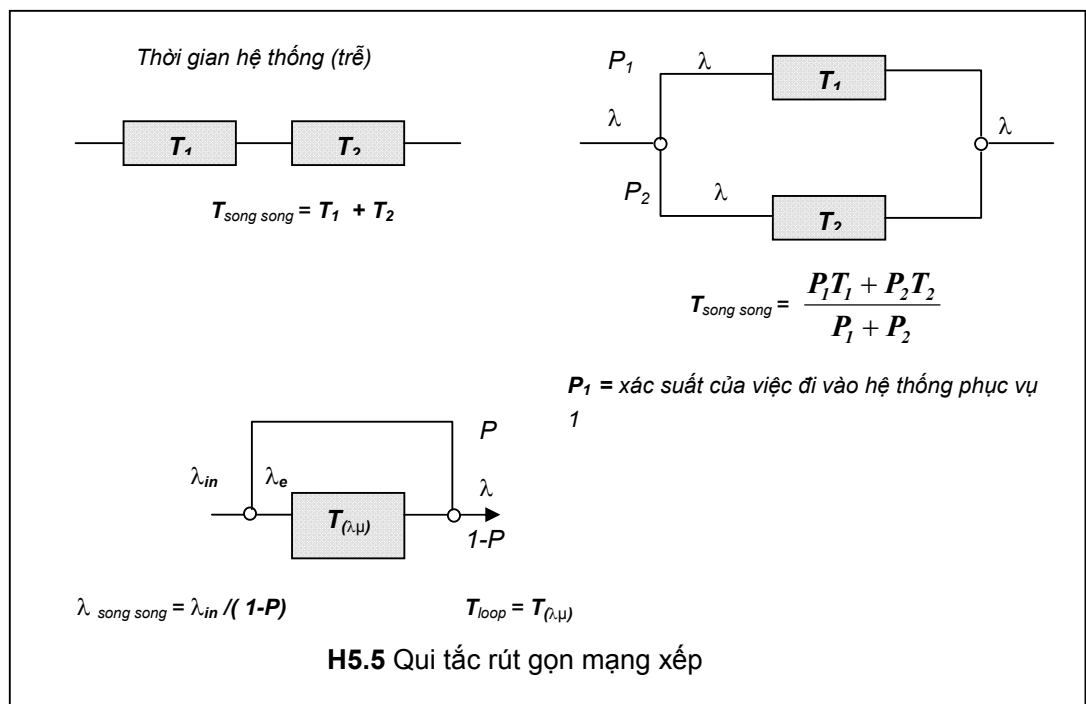
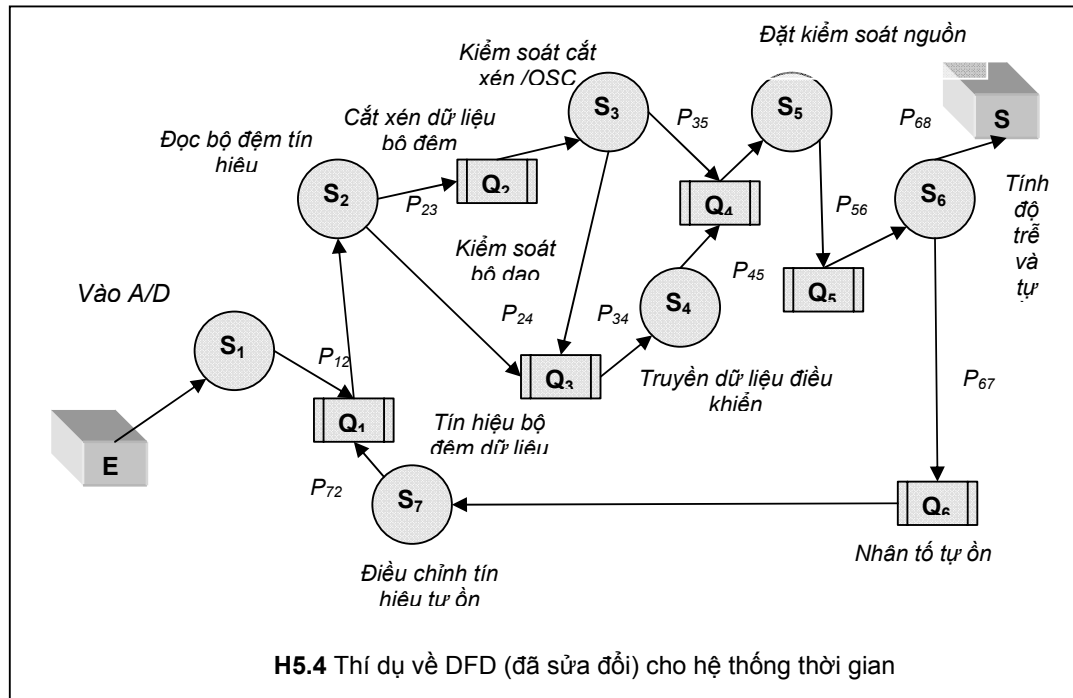
Chiều dài hàng đợi kỳ vọng: $N_q = \rho^2 / (1 - \rho)$

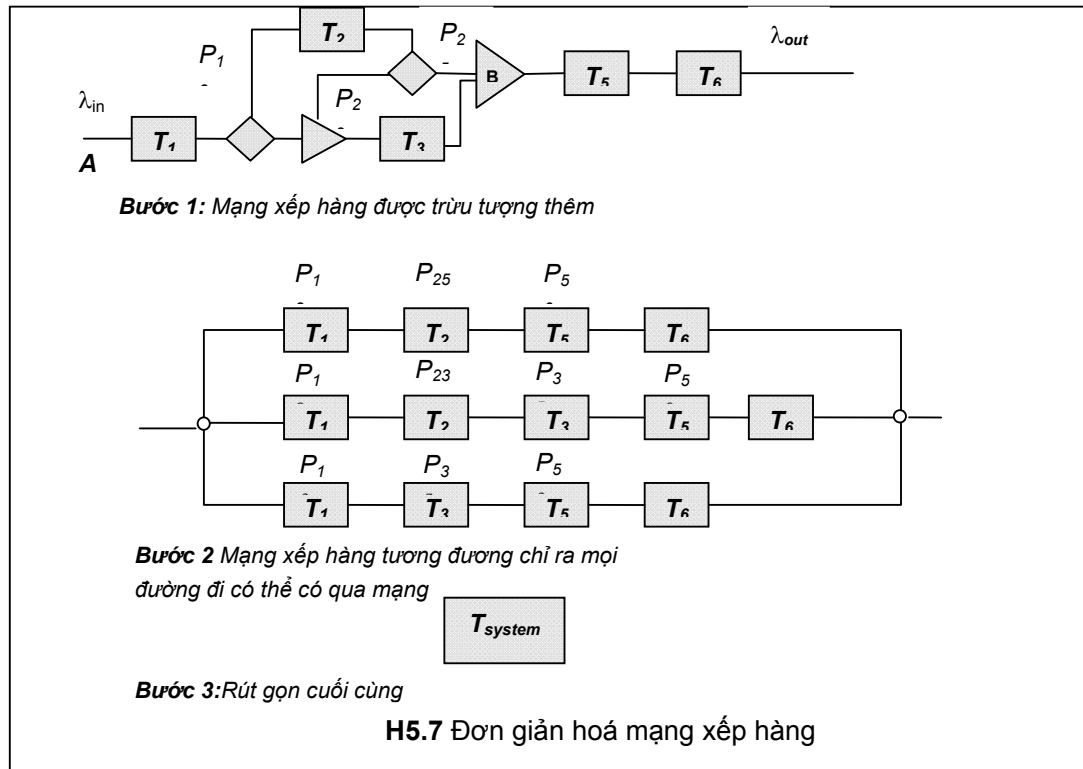
Số kỳ vọng trong hệ con: $N_s = \rho / (1 - \rho)$

Thời gian kỳ vọng trong hàng đợi: $T_q = \lambda / (\mu(\mu - \lambda))$

Thời gian kỳ vọng trong hệ con: $T_s = 1 / (\mu - \lambda)$

Với μ là tỷ lệ hoàn thành (số hoàn thành/giây). Với việc áp dụng các quy tắc rút gọn mạng hàng đợi, minh hoạ trong hình 5.5, mạng xếp hàng ban đầu được suy dẫn từ biểu đồ luồng dữ liệu có thể được đơn giản hoá bằng cách áp dụng các bước được vẽ trong hình 5.7. Toàn bộ thời gian dành cho hệ thống sẽ được tính là 2,37 giây.





V.2.2 Các kỹ thuật mô hình hoá cho hệ thống thời gian thực

V.2.2.1 Cách nhìn quan niệm

Các vấn đề chức năng được xử lý bằng cách dùng các **hoạt động** biểu thị cho khả năng xử lý của hệ thống.

Việc giải quyết yêu cầu xác nhận chỗ của khách hàng trong hệ thống giữ chỗ máy bay là một ví dụ về hoạt động, cũng như việc cập nhật vị trí máy bay trong hệ thống kiểm soát không lưu.

Các hoạt động có thể lồng nhau, tạo nên một cấp bậc mà thiết lập sự phân rã chức năng của hệ thống. Các khoản mục thông tin, như khoảng cách tới mục tiêu hay tên khách hàng, sẽ 'chảy' giữa các hoạt động, và cũng có thể được giữ trong các kho dữ liệu.

Cách nhìn chức năng này về hệ thống được thu tóm trong các sơ đồ *hoạt động*, tương tự như biểu đồ luồng dữ liệu qui ước.

Các hành vi điều khiển (hành vi động) được giải quyết bằng sơ đồ *trạng thái*. Tại đây, trạng thái có thể được lồng nhau và được nối theo một số cách để biểu diễn hành vi tuần tự hay tương tranh.

Ví dụ: Máy tính có nhiệm vụ điều khiển không lưu có thể ở một trong ba trạng thái: không đối không, không đối đất hay hoa tiêu. Đồng thời nó phải ở trong trạng thái hoặc là điều khiển bay tự động hay điều khiển tay. Việc chuyển trạng thái được bố trí theo sự kiện, chúng có thể được tính theo các điều kiện. Ví dụ, bật một khoá nào đó trên bàn điều chỉnh, là một sự kiện sẽ gây ra việc chuyển trạng thái từ trạng thái hoa tiêu sang trạng thái tiếp đất, nhưng với điều kiện là máy bay đã có thiết bị hạ cánh.

Có hai cách nhìn về hệ thống là sơ đồ hoạt động và sơ đồ trạng thái, chúng được tích lại theo cách sau:

Sơ đồ trạng thái, gọi là hoạt động điều khiển, được liên kết với từng mức của sơ đồ hoạt động. Vai trò của sơ đồ trạng thái là điều khiển các hoạt động và luồng dữ liệu ở mức đó. *Một sơ đồ trạng thái có thể thực thi điều khiển trên các hoạt động*, ví dụ nó có thể chỉ thị cho các hoạt động bắt đầu hay dừng lại, tạm ngừng hay chạy tiếp công việc. Nó có thể thay đổi các giá trị của biến và do vậy ảnh hưởng đến việc xử lý của các hoạt động liên quan. Nó cũng có thể gửi các tín hiệu tới các hoạt động khác và gây cho chúng thay đổi hành vi riêng của mình.

Ngoài ra, một sơ đồ trạng thái kiểm soát cũng có thể cũng có thể cảm nhận các hành động được các sơ đồ trạng thái khác tiến hành. Ví dụ, nếu một sơ đồ trạng thái bắt đầu một hoạt động hay tăng giá trị của một biến thì một sơ đồ trạng thái khác có thể cảm thấy sự kiện đó và sẽ có phản ứng chẳng hạn bố trí một việc chuyển.

Sơ đồ hoạt động và sơ đồ trạng thái là có liên kết chặt chẽ với nhau nhưng chúng không phải là biểu diễn khác nhau của cùng một đối tượng. Sơ đồ hoạt động phần chung là không đầy đủ khi xét như mô hình hệ thống vì chúng không đề cập tới hành vi. Sơ đồ trạng thái cũng không đầy đủ vì không có các hoạt động thì chúng không có gì mà điều khiển cả. Một sơ đồ hoạt động chi tiết gắn với một sơ đồ trạng thái điều khiển của nó đưa ra một mô hình khái niệm. Sơ đồ hoạt động là xương sống

của mô hình, trong khi đó sơ đồ trạng thái là lực điều khiển đằng sau hành vi của hệ thống.

V.2.2.2 Cách nhìn vật lý

Một đặc tả dùng sơ đồ hoạt động và sơ đồ trạng thái dưới dạng mô hình quan niệm là một nền tảng cực tốt, nhưng nó không phải là hệ thống thực. Cái bị bỏ mất là phương tiện để mô tả hệ thống từ việc cài đặt, và một phương tiện để đảm bảo rằng hệ thống được cài đặt một cách đúng đắn theo đặc tả của nó. Một phần quan trọng của điều này là mô tả cho việc phân rã vật lý của hệ thống và mối quan hệ của nó với mô hình khái niệm.

Các khía cạnh vật lý được giải quyết bằng cách dùng ngôn ngữ sơ đồ module. Thuật ngữ ‘vật lý’ hay ‘module’ được dùng một cách tổng quát để chỉ các thành phần của hệ thống, dù đó là phần cứng, phần mềm hay lai. Giống như các hoạt động trong sơ đồ hoạt động, các module được bố trí theo thứ bậc để chỉ ra việc phân rã của hệ thống thành các thành phần của nó và các thành phần con. Các module được nối bởi các luồng chảy, cái mà người ta có thể coi như cái mang thông tin giữa các module.

V.2.2.3 Phân tích và mô phỏng

Một khi chúng ta đã xây dựng được một *mô hình khái niệm*, bao gồm cả sơ đồ hoạt động và sơ đồ trạng thái điều khiển của nó thì ta có thể phân tích và kiểm thử kỹ càng nó. Mô hình này có thể mô tả toàn bộ hệ thống, xuống đến tận mức chi tiết thấp nhất, hay nó chỉ có thể là một đặc tả bộ phận.

Trước hết chúng ta phải chắc chắn rằng mô hình khái niệm này là đúng đắn về cú pháp. Điều này làm phát sinh nhiều phép kiểm thử tương đối trực tiếp, ví dụ, kiểm thử tính đầy đủ của mô hình (thiếu nhãn hay tên...), kiểm thử các định nghĩa của các phần tử không đồ họa (như sự kiện, điều kiện), kiểm thử tinh vi (tính đúng đắn của cái vào và cái ra).

Ví dụ: kiểm thử cho các phần tử được dùng trong sơ đồ trạng thái, không phải là cái vào mà cũng không bị ảnh hưởng bên trong, như sự kiện bật nguồn thường có ý nghĩa gây nên việc chuyển trạng thái trong sơ đồ trạng thái nhưng không xác định nhưng không được xác định trong sơ đồ trạng thái như một *cái vào*. Tất cả những điều

này thường được nói đến như các phép kiểm thử tính nhất quán và đầy đủ, phần lớn chúng đều tương tự với việc kiểm tra được thực thi bởi trình biên dịch trước khi dịch thực tế cho một ngôn ngữ lập trình.

V.2.2.4 Kịch bản chạy

Một mô hình đúng đắn về cú pháp mô tả chính xác một hệ thống nào đó. Tuy nhiên, nó có thể không phải là hệ thống ta tưởng tượng trong đầu. Trong thực tế, hệ thống đã mô tả có thể còn nhược điểm nghiêm trọng - việc đúng về cú pháp không đảm bảo tính đúng đắn cho chức năng hay hành vi.

Mục tiêu thực của việc phân tích mô hình là để tìm ra liệu nó có mô tả đúng hệ thống chúng ta muốn hay không. Việc phân tích làm cho chúng ta có thể biết được nhiều hơn về mô hình đã được xây dựng để xem xét lại cách thức hệ thống dựa trên việc nó sẽ hành xử thế nào và để kiểm chứng rằng nó thực sự đáp ứng với sự mong đợi. Điều này đòi hỏi một ngôn ngữ mô hình hoá với nhiều cú pháp hình thức hơn.

Nếu mô hình được dựa trên ngữ nghĩa hình thức thì người kỹ sư hệ thống có thể thực hiện mô hình đó. Người kỹ sư có thể tạo ra và cho chạy một kịch bản để qua đó cho phép mình ‘nhấn phím’ và quan sát hành vi của mô hình trước khi hệ thống thực tế được xây dựng.

Ví dụ: Thực hiện mô hình về máy trả tiền tự động ATM các bước sau đây sẽ được khởi tạo

- (1) Mô hình quan niệm được tạo ra
- (2) Người kỹ sư đóng vai trò khách hàng và máy tính, như ngân hàng, sinh ra một sự kiện như đưa vào một thẻ ngân hàng, nhấn các phím và thông tin số dư tiền có mới đưa ra.
- (3) Phản ứng của hệ thống với những sự kiện này được điều phối
- (4) Tính không nhất quán với hành vi được ghi lại
- (5) Mô hình quan niệm được sửa đổi để phản ánh hành vi đúng
- (6) Lặp đi lặp lại điều này cho tới khi hệ thống tiến hoá đến chỗ mong muốn

Người kỹ sư hệ thống cho chạy kịch bản và xem đáp ứng của hệ thống bằng đồ hoạ. Các phần tử active (trạng thái hệ thống đang ở thời điểm đó hay các phần tử đang kích hoạt) đều được phát sáng về đồ hoạ, việc thực hiện hoạt động nảy sinh trong cách biểu diễn hoạt hình của mô hình. Việc thực hiện kịch bản mô phỏng hệ thống chạy trong thời gian thực và lưu giữ dấu vết về thông tin phục thuộc thời gian. Tại bất kỳ điểm nào trong khi thực hiện, người kỹ sư hệ thống cũng đều có thể xem trạng thái của bất kỳ phần tử không có đồ hoạ khác, như giá trị biến hay một điều kiện.

V.2.2.5 Mô phỏng lập trình

Kịch bản cho phép người kỹ sư hệ thống tập duyệt chương trình theo cách tương tác. Tuy nhiên nhiều khi chúng ta có thể mong muốn có một sự mô phỏng chặt chẽ hơn, khi đó, Ngôn ngữ điều khiển mô phỏng (SCL) làm cho người kỹ sư giữ được điều khiển chung về cách xử lý thực hiện, nhưng đồng thời khai thác được sức mạnh của công cụ để chuyển giao nhiều chi tiết.

Một trong những điều đơn giản nhất có thể làm được với SCL là đọc danh sách các sự kiện từ một tệp theo lô. Tức là có thể chuẩn bị trước các kịch bản dài hay một phần của chúng rồi cho chạy tự động. Người kỹ sư hệ thống có thể quan sát những kịch bản này, mặt khác, có thể lập trình bằng SCL để đặt các điểm đứt và kiểm soát các biến, trạng thái hay điều kiện nào đó.

Ví dụ: cho chạy việc mô phỏng một hệ thống điều khiển không lưu, người kỹ sư có thể yêu cầu chương trình SCL dừng lại bất kỳ chỗ nào mà ra-đa bám vào mục tiêu và chuyển sang một tương tác. Một khi 'việc bám' được nghi nhận thì người kỹ sư chuyển sang làm việc tương tác để xem xét trạng thái này chi tiết hơn.

Việc dùng kịch bản và mô phỏng cũng cho phép người kỹ sư thu thập được những thống kê có ý nghĩa về sự vận hành của hệ thống cần phải xây dựng.

Ví dụ: chúng ta có thể muốn biết bao nhiêu lần trong một chuyến bay điển hình của máy bay, mà ra-đa bị mất mục tiêu đeo bám.

Vì người kỹ sư khó có thể gắn mọi thứ vào một kịch bản bay duy nhất (nhưng phải bao hàm tất cả) nên có thể xây dựng một mô phỏng được lập trình, bằng cách dùng các kết quả tích lũy từ các kịch bản khác để thu được số thống kê các trường

hợp trung bình. Chương trình điều khiển mô phỏng sinh ra các sự kiện ngẫu nhiên tương ứng theo xác suất định sẵn, do vậy, các sự kiện xuất rất hữu hạn (ví dụ, bật ra khỏi ghế ngồi trong máy bay chiến đấu) có

thể được gán các xác suất rất thấp, trong khi sự kiện khác lại được gán xác suất cao hơn và việc lựa chọn ngẫu nhiên cho các sự kiện do vậy trở thành *thực*. Để thu được các số thống kê mong muốn, chúng ta nên thêm vào điểm dứt trong chương trình SCL.

V.2.2.6 Dịch tự động thành mã

Một khi mô hình hệ thống đã được xây dựng nên thì nó có thể được dịch toàn bộ thành mã máy chạy được bằng các chức năng tạo bản mẫu. Các sơ đồ hoạt động và sơ đồ trạng thái điều khiển của chúng có thể được dịch trong ngôn ngữ lập trình cấp cao, như ADA hay C, C++. Ngày nay việc dùng chủ yếu của mã kết quả là để quan sát một hệ thống thực hiện trong những hoàn cảnh gần với thế giới thực nhất. Ví dụ, mã bản mẫu có thể được thực hiện theo bộ mô phỏng thật đầy đủ cho môi trường đích hay trong bản thân môi trường cuối cùng. Đây chưa phải là mã sản xuất hay mã cuối cùng. Do vậy, bao giờ nó cũng có thể phản ánh hiệu năng thời gian thực chính xác của hệ thống dự định. Tuy nhiên, việc kiểm thử hiệu năng của hệ thống gần với hoàn cảnh thực nhất bao giờ cũng có ích.

V.3 Phương pháp thiết kế

Việc thiết kế phần mềm thời gian thực phải tổ hợp tất cả những khái niệm nền tảng liên kết với phần mềm chất lượng cao. Bên cạnh đó, phần mềm thời gian thực còn đặt ra một tập các vấn đề duy nhất cho người thiết kế:

- Biểu diễn các ngắt và việc chuyển hoàn cảnh
- Tương tranh như được biểu hiện trong đa nhiệm và đa xử lý
- Truyền thông và đồng bộ hoá giữa các nhiệm vụ
- Độ biến thiên lớn trong tỉ lệ dữ liệu và truyền thông
- Biểu diễn các ràng buộc thời gian

- Xử lý không đồng bộ
- Việc móc nối không tránh khỏi và cần thiết với hệ điều hành, phần cứng và các phần tử hệ thống ngoài khác.

Trong suốt hai thập kỷ qua, một số các phương pháp thiết kế phần mềm thời gian thực đã được đề nghị để giải quyết một số hay tất cả những vấn đề đã được đề cập ở trên. Một số phương pháp thiết kế mở rộng một trong ba lớp phương pháp luận thiết kế (như các phương pháp luận luồng dữ liệu, cấu trúc dữ liệu hay hướng sự vật). Nhưng người khác thì giới thiệu cách tiếp cận hoàn toàn tách biệt, dùng mô hình máy hữu hạn trạng thái hay hệ thống truyền thông báo, mạng Petri hay một ngôn ngữ đặc biệt làm cơ sở.

V.4 Phương pháp thiết kế hướng luồng dữ liệu

Phương pháp hướng luồng dữ liệu là được dùng rộng rãi nhất trong công nghiệp. Các kỹ thuật ánh xạ được áp dụng cho biểu đồ luồng dữ liệu không thích hợp để hỗ trợ cho tất cả các vấn đề thiết kế thời gian thực đã được lưu ý trước đây trong chương này.

Hasan Gomaa đã xây dựng các mở rộng cho cách biểu diễn gian thực. Cách tiếp cận của Gomaa, còn gọi là *Phương pháp thiết kế cho hệ thống thời gian thực* (DARTS), cho phép người thiết kế hệ thống thời gian thực thích nghi các kỹ thuật luồng dữ liệu với những nhu cầu đặc biệt của ứng dụng thời gian thực. Các mục sau đây được trích lại từ công trình của Gomaa.

V.4.1 Các yêu cầu về phương pháp thiết kế hệ thống thời gian thực

ác kỹ thuật thiết kế hướng luồng dữ liệu đưa ra một nền tảng đáng giá cho thiết kế thời gian thực. Các biểu đồ luồng dữ liệu mô tả luồng thông tin giữa các chức năng hệ thống (hay nhiệm vụ) và các kỹ thuật ánh xạ làm cho người thiết kế có thể suy dẫn ra cấu trúc chương trình từ các đặc trưng luồng dữ liệu. Các biện pháp chất lượng thiết kế như tính modul và độc lập chức năng được tăng cường với phương pháp hướng luồng dữ liệu.

Phương pháp thiết kế phần mềm thời gian thực DARTS xây dựng trên kĩ pháp và cách tiếp cận cho thiết kế hướng luồng dữ liệu của phần mềm qui ước. Để hỗ trợ cho thiết kế thời gian thực, các phương pháp luồng dữ liệu phải được mở rộng thêm bằng cách cung cấp:

(1) một cơ chế để biểu diễn việc truyền thông và đồng bộ hoá nhiệm vụ

(2) một kĩ pháp để biểu diễn sự phụ thuộc trạng thái, (3) một cách tiếp cận "nổi" các phương pháp luồng dữ liệu qui ước với thế giới thời gian thực

Vì phần lớn các hệ thống thời gian thực sinh ra nhiều nhiệm vụ mà hoặc cùng dùng chung một bộ xử lí hay thực hiện đồng thời trên các bộ xử lí phân tán, nên những cơ chế này phải có được khả năng đồng bộ hoá các nhiệm vụ và đưa ra việc truyền thông giữa các nhiệm vụ này. Việc đồng bộ hoá xảy ra thông qua việc *loại trừ lẫn nhau* hay *kích thích chéo*.

Loại trừ lẫn nhau được áp dụng khi hai nhiệm vụ có thể đồng thời thâm nhập vào một vùng dữ liệu chung. Cờ báo hiệu được dùng để *loại trừ* nhiệm vụ này khỏi việc thâm nhập vào dữ liệu trong khi nhiệm vụ khác đang dùng (đọc hay ghi) cùng dữ liệu đó.

Kích thích chéo được cài đặt khi nhiệm vụ này báo hiệu cho nhiệm vụ khác (đang đợi) rằng nó đã hoàn thành một hoạt động nào đó và có thể tiến hành nhiệm vụ được báo.

Việc truyền thông nhiệm vụ xuất hiện khi nhiệm vụ này phải truyền thông tin cho nhiệm vụ khác. *Liên lạc thông báo* là cách tiếp cận thông thường. Khi *nhiệm vụ sản xuất* gửi một thông báo cho *nhiệm vụ tiêu thụ*, và rồi đợi sự đáp ứng từ nhiệm vụ tiêu thụ thì việc liên lạc được gọi là *gắn chặt*. Khi các nhiệm vụ sản xuất và tiêu thụ tiếp tục xử lí theo tỉ lệ riêng của chúng và dùng hàng đợi thông báo để làm bộ đệm thông báo thì việc liên lạc được gọi là *gắn lỏng*.

Để cài đặt việc liên lạc thông báo, Gomaa mô tả ba cách tiếp cận khác nhau:

1. Hệ điều hành thời gian thực có thể cung cấp các nguyên sơ liên lạc giữa các nhiệm vụ.

2. Cơ chế liên lạc giữa các nhiệm vụ có thể được cài đặt bên trong ngữ cảnh ngôn ngữ lập trình (như Ada).

3. Một bộ khiếm giải (handler) liên lạc đặc biệt có thể được tạo ra bằng cách dùng các nguyên sơ đồng bộ hoá do hệ điều hành cung cấp

Cách tiếp cận DARTS cung cấp một kĩ pháp hỗ trợ cho cả liên lạc gắn chặt và gắn lỏng.

V.4.2 DARTS

Phương pháp thiết kế DARTS bắt đầu với việc áp dụng các nguyên tắc phân tích phần mềm nền tảng (như phân tích lĩnh vực thông tin phân hoạch vấn đề) được áp dụng trong ngữ cảnh ký pháp luồng dữ liệu. Các biểu đồ luồng dữ liệu được tạo ra, từ điển dữ liệu tương ứng được định nghĩa và giao diện giữa các chức năng (các phép biến đổi) hệ thống chính được thiết lập.

Một khi mô hình luồng và từ điển dữ liệu đã được tạo ra thì hệ thống phải được xem xét theo một quan điểm khác. DFD không mô tả các nhiệm vụ không đồng bộ và tương tranh. Do đó chúng ta cần một cách tiếp cận xác định ra các nhiệm vụ hệ thống thời gian thực trong hoàn cảnh của chức năng hệ thống được rút ra từ một DFD. Các phép biến đổi được gộp thành nhóm các nhiệm vụ thời gian thực. Luồng dữ liệu giữa các nhiệm vụ mới được xác định sẽ định ra các yêu cầu liên lạc giữa các nhiệm vụ. Một số tiêu chuẩn để xác định xem liệu các phép biến đổi DFD có nên được xác định như các nhiệm vụ tách biệt hay được gộp nhóm với các phép biến đổi khác thành một nhiệm vụ:

Phụ thuộc vào (vào/ra): Phụ thuộc vào cái vào hay cái ra, một phép biến đổi thường bị ràng buộc phải chạy với tốc độ do tốc độ của các thiết bị vào/ra chi phối khi nó thực hiện tương tác. Trong trường hợp này, phép biến đổi cần phải là một nhiệm vụ tách biệt.

Chức năng gắng về thời gian: Một chức năng gắng về thời gian cần phải chạy với mức ưu tiên cao và do đó cần phải là một nhiệm vụ tách biệt.

Yêu cầu tính toán: Chức năng đòi hỏi tính toán nhiều (một tập các chức năng) có thể chạy như một nhiệm vụ với số ưu tiên thấp – tiêu thụ chu kỳ CPU dự phòng.

Tính cố kết chức năng: Nhưng phép biến đổi thực hiện một tập các chức năng có liên quan chặt chẽ có thể được gộp nhóm với nhau trong một nhiệm vụ. Vì lưu lượng dữ liệu giữa những chức năng này có thể cao nên việc để chúng là các nhiệm vụ tách biệt bên trong cùng nhiệm vụ lại bảo đảm tính cố kết chức năng cả tại mức module lẫn các nhiệm vụ.

Tính cố kết thời gian: Một số phép biến đổi thực hiện các chức năng được tiến hành đồng thời. Những chức năng có thể được gộp nhóm thành một nhiệm vụ sao cho chúng được thực hiện mỗi lần nhiệm vụ nhận được một kích thích.

Thực hiện theo định kỳ: Một phép biến đổi cần được thực hiện định kỳ cũng có thể được cấu trúc như một nhiệm vụ tách biệt và được kích hoạt theo từng khoảng đều đặn.

Một khi các nhiệm vụ đã được xác định thì DARTS cung cấp một cơ chế để giải quyết liên lạc giữa các nhiệm vụ bằng cách xác định ra hai lớp “module giao diện nhiệm vụ”: module liên lạc nhiệm vụ (TCM) và module đồng bộ hóa nhiệm vụ (TSM). Một TCM được sinh ra bởi nhiệm vụ liên lạc và dùng các nguyên sơ đồng bộ hóa của hệ điều hành để đảm bảo việc thâm nhập đúng đắn vào dữ liệu. TCM được chia thành hai loại:

Module liên lạc thông báo (MCM) hỗ trợ cho việc liên lạc thông báo và cài đặt các cơ chế thích hợp để quản lý hàng đợi thông báo (đối với liên lạc gián lỏng) và các nguyên sơ đồng bộ hóa (đối với liên lạc gắn chặt). Ký pháp DARTS cho việc liên lạc giữa các nhiệm vụ được cài đặt bởi MCM.

Các module che dấu thông tin (IHM) cung cấp việc thâm nhập vào nhóm dữ liệu hay kho dữ liệu. IHM cài đặt cấu trúc dữ liệu cũng như các phương pháp thâm nhập cho phép các nhiệm vụ khác thâm nhập cấu trúc dữ liệu này.

Khi điều khiển, chứ không phải dữ liệu, được truyền giữa các nhiệm vụ thì module đồng bộ hóa nhiệm vụ bắt đầu làm việc. Một nhiệm vụ có thể báo cho một nhiệm vụ khác rằng một sự kiện đã xuất hiện, hay một nhiệm vụ có thể đợi một tín hiệu

như vậy. Một TSM có thể được coi như “module giám sát”, nó quản lý điều khiển và điều phối các sự kiện xuất hiện.

V.4.3 Thiết kế nhiệm vụ

Một nhiệm vụ hệ thống thời gian thực là một chương trình có thể được thiết kế bằng cách dùng các phương pháp quy ước, tức là, một nhiệm vụ có thể được coi như một tiến trình tần tự được cài đặt bằng một cấu trúc chương trình có thể được thiết kế bằng cách dùng triết lý lập trình có cấu trúc.

Do đó, DDF vốn biểu thị cho luồng dữ liệu bên trong biên giới một nhiệm vụ, có thể được ánh xạ trong một cấu trúc chương trình bằng cách dùng các kỹ thuật ánh xạ giao tác hay phép biến đổi qui ước. Tuy nhiên, với các hệ thống thời gian thực điều khiển còn phụ thuộc vào cả cái vào do nhiệm vụ cung cấp và trạng thái hiện tại của hệ thống (nghĩa là điều gì đã xảy ra để đặt hệ thống vào trạng thái hiện tại của nó). Bởi lý do này mà điều khiển gọi là phụ thuộc trạng thái.

Gomaa gợi ý một cơ chế để giải quyết tình huống này mà ông gọi là bộ quản lý chuyển trạng (STM). STM là một cài đặt của bảng kích hoạt tiến trình, nhận biết trạng thái hiện tại của hệ thống và dùng một bảng chuyển trạng thái để hướng dẫn xử lý.

Tóm tắt

Việc thiết kế phần mềm thời gian thực bao gồm tất cả các khía cạnh của thiết kế phần mềm quy ước nhưng đồng thời lại đưa vào một tập các tiêu chuẩn thiết kế và mối quan tâm mới. Vì phần mềm thời gian thực phải đáp ứng với các sự kiện thế giới thực trong khuôn khổ thời gian do những sự kiện này khống chế cho tất cả các lớp thiết kế (thiết kế kiến trúc, thủ tục và dữ liệu) trở nên phức tạp hơn.

Điều khó khăn và thường không thực tế là phải cách ly thiết kế phần mềm khỏi các vấn đề hướng hệ thống lớn hơn. Vì phần mềm thường bị điều khiển bởi thời gian hay sự kiện nên người thiết kế phải xem xét về chức năng và sự hoàn thiện của phần cứng và phần mềm. Xử lý ngắt và tỷ lệ truyền dữ liệu, cơ sở dữ liệu và hệ điều hành phân tán, ngôn ngữ lập trình chuyên dụng và các phương pháp đồng bộ hoá chỉ mới là một trong số những quan tâm của người thiết kế hệ thống thời gian thực.

Việc phân tích hệ thống thời gian thực bao quát việc làm mô hình hoá và mô phỏng toán học. Các mô hình mạng và sắp hàng làm cho người thiết kế hệ thống định giá được thời

gian đáp ứng toàn bộ, tỷ lệ xử lý và các vấn đề thời gian và ràng buộc khác. Chẳng hạn, cách tiếp cận DARTS mở rộng thiết kế hướng luồng dữ liệu bằng cách đưa ra một kỹ pháp và cách tiếp cận đề cập tới các đặc trưng của hệ thống thời gian thực.

Thiết kế phần mềm cho các hệ thống thời gian thực vẫn còn là một thách thức. Tiến bộ đã có, phương pháp đã tồn tại nhưng vẫn còn phải thực hiện việc đánh giá thực tế cho những gợi ý hiện đại này.

CHƯƠNG VI**KỸ NGHỆ WEB****VI.1 Các thuộc tính của ứng dụng dựa trên Web**

Hầu như không có tranh luận về hệ thống và ứng dụng dựa trên Web (chúng ta sẽ gọi chung là ứng dụng Web) khác với nhiều loại phần mềm máy tính khác. Powell tóm tắt những sự khác biệt chính khi ông ấy nói rằng hệ thống dựa trên Web "bao gồm sự trộn lẫn giữa việc xuất bản in ấn và việc phát triển phần mềm, giữa tiếp thị và tính toán, giữa truyền thông nội bộ và mối quan hệ bên ngoài, và giữa nghệ thuật và công nghệ". Các thuộc tính sau đây thường hay gặp phải trong đại đa số ứng dụng Web:

Dùng nhiều về mạng: Bởi bản chất của nó, ứng dụng mạng chủ yếu dùng trên mạng. Nó nằm trên mạng và phải phục vụ nhu cầu của cộng đồng khác hàng đa dạng. Một ứng dụng mạng có thể nằm trên Internet (do đó tạo khả năng truyền thông toàn cầu mở). Như một sự lựa chọn, một ứng dụng có thể được đặt trên mạng nội bộ (thực hiện truyền thông toàn tổ chức) hoặc trên mạng ngoại bộ (truyền thông liên mạng).

Điều khiển theo nội dung: Trong nhiều trường hợp, chức năng chính của ứng dụng Web là dùng multimedia để trình bày nội dung văn bản, đồ họa, âm thanh, video cho người dùng cuối.

Tiến hoá liên tục: Không giống phần mềm ứng dụng qui ước thường tiến hoá qua một loạt việc đưa ra theo kế hoạch, theo khoảng thời gian, ứng dụng Web tiến hoá liên tục. Điều thường thấy là với một số ứng dụng Web (đặc biệt nội dung của chúng) được cập nhật theo lịch hàng giờ.

Các đặc trưng ứng dụng Web sau hướng dẫn cho tiến trình:

Ngay tức khắc: Các ứng dụng Web có tính tức khắc không thấy có trong bất kì kiểu phần mềm nào. Tức là thời gian ra thị trường cho một vị trí Web có thể là vấn đề vài ngày hay vài tuần. Người phát triển phải dùng các phương pháp cho lập kế hoạch,

phân tích, thiết kế, cài đặt và kiểm thử đã được thích ứng cho lịch thời gian dồn nén cần cho việc phát triển ứng dụng Web.

An toàn: Bởi vì ứng dụng Web là có sẵn cho việc truy nhập mạng, nên khó, nếu không nói là không thể, giới hạn số người dùng cuối có thể truy nhập vào ứng dụng. Để bảo vệ nội dung nhạy cảm và cung cấp "mốt" an toàn cho việc truyền dữ liệu, các biện pháp an toàn mạnh phải được cài đặt qua toàn bộ kết cấu nền hỗ trợ cho ứng dụng Web và bên trong bản thân ứng dụng.

Thẩm mỹ: Phần không thể chối bỏ được của sự hấp dẫn của ứng dụng Web là việc nhìn và cảm nhận về nó. Khi một ứng dụng đã được thiết kế đưa ra thị trường hay bán sản phẩm hay ý tưởng, thì tính thẩm mỹ có thể có nhiều điều cần quan tâm cho sự thành công cũng như thiết kế kĩ thuật.

Những đặc trưng tổng quát này áp dụng cho mọi ứng dụng Web nhưng với mức độ ảnh hưởng khác nhau. Các loại ứng dụng sau đây là thường hay gặp nhất trong công trình về Web:

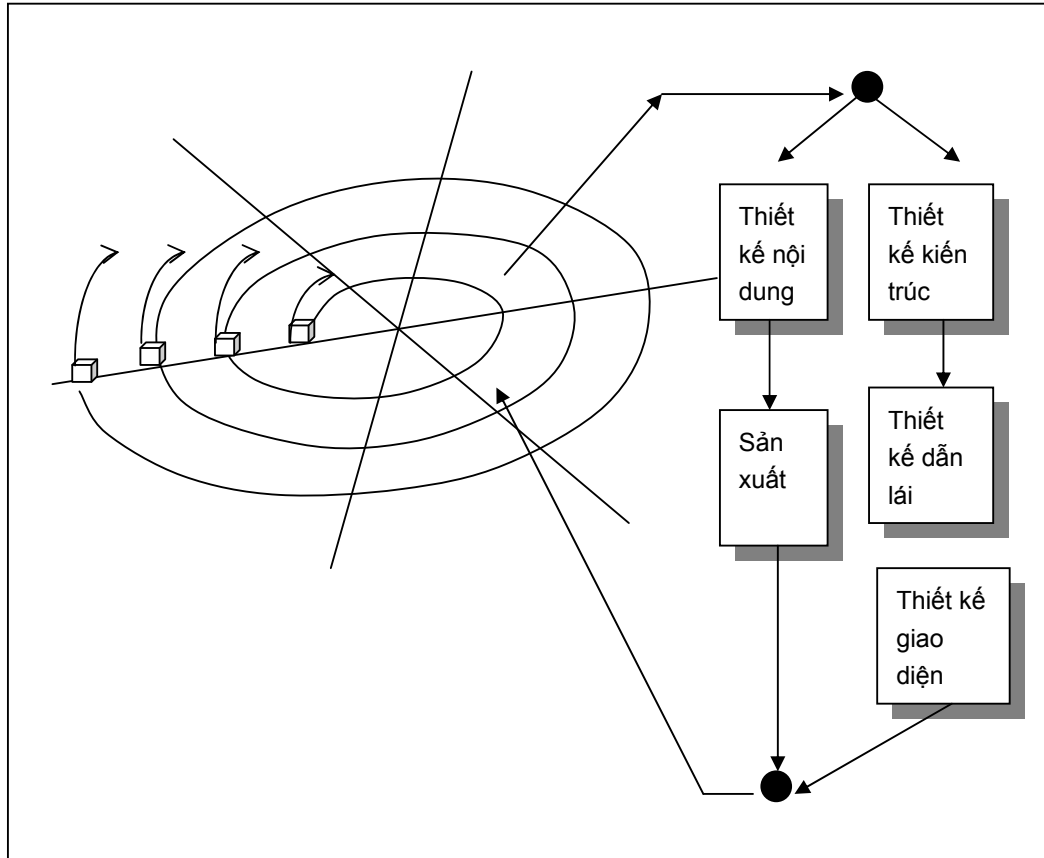
Các đặc trưng của hệ thống và ứng dụng dựa trên Web có ảnh hưởng sâu sắc tới việc xử lí kĩ nghệ Web.

VI.2 Một khuôn khổ cho kĩ nghệ Web

Khi ứng dụng Web tiến hoá từ các nguồn thông tin tĩnh, hướng nội dung sang môi trường ứng dụng hướng người dùng, động, thì nhu cầu áp dụng việc quản lí và các nguyên lí kĩ nghệ vững chãi trở thành quan trọng. Để thực hiện điều này, cần phát triển một khuôn khổ kĩ nghệ Web bao gồm một mô hình tiến trình hiệu quả, được đưa vào bằng các hoạt động khuôn khổ và nhiệm vụ kĩ nghệ. Mô hình tiến trình cho kĩ nghệ Web được gợi ý trong Hình 6.1.

Tiến trình kĩ nghệ Web bắt đầu với việc *phát biểu* - một hoạt động nhận diện ra các mục tiêu và mục đích của ứng dụng Web và thiết lập phạm vi cho lần tăng đầu tiên. *Lập kế hoạch* ước lượng chi phí dự án tổng thể, ước lượng rủi ro liên kết với nỗ lực phát triển, và xác định lịch phát triển dần cho bước tăng ứng dụng Web ban đầu, với một lịch kết tập thô cho các bước tăng sau. *Phân tích* thiết lập nên các yêu cầu kĩ

thuật cho ứng dụng Web và nhận diện các khoản mục nội dung sẽ được tổ hợp vào. Các yêu cầu về thiết kế đồ hoạ (thẩm mỹ) cũng được xác định.



H 6.1 Mô hình tiến trình kỹ nghệ Web

Hoạt động *chế tạo* tổ hợp hai nhiệm vụ song song được minh hoạ ở phía bên phải của Hình 6.1. *Thiết kế nội dung* và *sản xuất* là các nhiệm vụ do các thành viên không kĩ thuật của tổ kỹ nghệ Web phụ trách. Ý định của các nhiệm vụ này là thiết kế, sản xuất và/hoặc kiểm được mọi nội dung văn bản, đồ hoạ, âm thanh và video sẽ được tích hợp vào trong ứng dụng Web. Đồng thời, một tập các nhiệm vụ thiết kế kĩ thuật cũng được tiến hành.

Sinh trang là một hoạt động xây dựng dùng rất nhiều công cụ tự động cho việc tạo ứng dụng Web. Nội dung được xác định trong hoạt động chế tạo được gộp với các thiết kế kiến trúc, dẫn lái và giao diện để tạo ra các trang Web thực hiện được trong

HTML, XML và các ngôn ngữ hướng tiến trình khác (như Java). Việc tích hợp với cấu phần phần giữa (như CORBA, DCOM hay Javabeans) cũng được thực hiện trong hoạt động này. *Kiểm thử* cho phép luyện tập việc dẫn lái ứng dụng Web; cố gắng phát hiện ra lỗi trong các tiểu dụng, mẫu...; và giúp đảm bảo rằng ứng dụng Web sẽ vận hành đúng trong các môi trường khác nhau (như với các trình duyệt khác nhau).

Mỗi bước tăng được tạo ra khi một phần của tiến trình kỹ nghệ Web được kiểm điểm trong *đánh giá của khách hàng*. Đây là điểm mà tại đó những thay đổi được yêu cầu (việc mở rộng phạm vi xuất hiện). Những thay đổi này được tích hợp vào trong con đường liếp qua luồng tiến trình tăng dần.

VI.3 Phát biểu/phân tích hệ thống dựa trên Web

Phát biểu và phân tích hệ thống và ứng dụng dựa trên Web biểu thị cho một dãy các hoạt động kỹ nghệ Web bắt đầu với việc nhận diện ra các mục tiêu tổng thể cho ứng dụng Web và kết thúc với việc phát triển một mô hình phân tích hay đặc tả yêu cầu cho hệ thống. Việc phát biểu cho phép khách hàng và người phát triển thiết lập nên một tập các mục tiêu và mục đích chung cho việc xây dựng ứng dụng Web. Nó cũng nhận diện phạm vi của nỗ lực phát triển và cung cấp một phương tiện để xác định kết quả thành công. Phân tích là hoạt động kỹ thuật nhận diện các yêu cầu dữ liệu, chức năng và hành vi cho ứng dụng Web.

V.3.1 Phát biểu

Powell gợi ý một tập các câu hỏi nên được trả lời vào lúc bắt đầu bước phát biểu:

- Động cơ chính cho ứng dụng Web là gì?
- Tại sao ứng dụng Web lại được cần tới?
- Ai sẽ dùng ứng dụng Web?

Câu trả lời cho từng câu hỏi đơn giản này nên được phát biểu cô đọng nhất có thể được.

Một khi tất cả các mục tiêu ứng dụng và thông tin đã được nhận diện, thì một lược sử người dùng được xây dựng nên. Lược sử người dùng thu tóm các tính năng có liên quan tới người dùng tiềm năng kể cả nền tảng của họ, tri thức, sở thích và nhiều thứ nữa".

Một khi các mục tiêu và lược sử người dùng đã được phát triển, thì hoạt động phát biểu hội tụ vào việc phát biểu phạm vi cho ứng dụng Web. Trong nhiều trường hợp, các mục tiêu đã được phát triển lại được tích hợp vào trong phát biểu phạm vi. Tuy nhiên, bên cạnh đó, cũng có ích mà chỉ ra mức độ tích hợp được trông đợi từ ứng dụng Web. Tức là thường thì cần tích hợp hệ thống tin hiện có (như ứng dụng cơ sở dữ liệu hiện có) với đầu cuối dựa trên Web. Vấn đề tính nối được xem xét tới tại giai đoạn này.

VI.3. 2 Phân tích

Các khái niệm và nguyên lý đã được thảo luận cho phân tích yêu cầu phần mềm cũng áp dụng được mà không cần xem xét lại cho hoạt động phân tích kỹ nghệ Web. Phạm vi được xác định trong hoạt động phát biểu được trau chuốt thêm để tạo ra một mô hình phân tích đầy đủ cho ứng dụng Web. Bốn kiểu phân tích được tiến hành trong kỹ nghệ Web:

Phân tích nội dung. Toàn bộ phổ của nội dung cần được ứng dụng Web cung cấp sẽ được nhận diện. Nội dung bao gồm văn bản, đồ họa và ảnh, dữ liệu video và âm thanh. Việc mô hình hoá dữ liệu có thể được dùng để nhận diện và mô tả từng sự vật dữ liệu được dùng bên trong ứng dụng Web.

Phân tích tương tác. Cách thức mà theo đó người dùng tương tác với ứng dụng Web được mô tả chi tiết. Các trường hợp sử dụng có thể được phát triển để cung cấp các mô tả chi tiết về tương tác này.

Phân tích chức năng. Các kịch bản sử dụng (trường hợp sử dụng) được tạo ra như một phần của phân tích tương tác sẽ xác định các thao tác sẽ được áp dụng cho nội dung ứng dụng Web và kéo theo các chức năng xử lý khác. Tất cả các thao tác và chức năng được mô tả chi tiết.

Phân tích cấu hình. Môi trường và kết cấu nền mà theo đó ứng dụng Web thường trú được mô tả chi tiết. Ứng dụng Web có thể thường trú trên Internet, mạng nội bộ hay mạng ngoại bộ. Bên cạnh đó, kết cấu nền (chẳng hạn kết cấu nền cấu phần và mức độ theo đó cơ sở dữ liệu sẽ được dùng để sinh ra nội dung) cho ứng dụng Web nên được nhận diện tại giai đoạn này.

Mặc dầu đặc tả các yêu cầu chi tiết được khuyến cáo cho các ứng dụng Web cỡ lớn, những tài liệu như vậy là hiếm hoi. Có thể biện minh rằng sự tiến hoá liên tục của các yêu cầu Web có thể làm cho bất kì tài liệu nào cũng trở thành lạc hậu trước khi nó được hoàn thành. Mặc dầu điều này có thể đúng trong trường hợp cực đoan, cũng cần xác định một mô hình phân tích phục vụ làm nền tảng cho hoạt động thiết kế tiếp sau. Tối thiểu, thông tin được thu thập trong bốn nhiệm vụ phân tích trước nên được xem xét lại, sửa đổi như được yêu cầu, và rồi tổ chức thành một tài liệu mà có thể được truyền cho người thiết kế ứng dụng Web.

VI. 4 Thiết kế ứng dụng dựa trên web

Bản chất ngay tức khắc của ứng dụng dựa trên Web đi kèm với sức ép tiến hoá liên tục buộc người kĩ sư Web phải thiết lập một thiết kế giải quyết vấn đề nghiệp vụ ngay lập tức trong khi đồng thời phải xác định kiến trúc ứng dụng có khả năng tiến hoá nhanh chóng qua thời gian. Tất nhiên vấn đề là ở chỗ việc giải quyết vấn đề ngay tức khắc (nhanh chóng) có thể làm phát sinh những thoả hiệp ảnh hưởng tới khả năng của ứng dụng để tiến hoá theo thời gian. Đây là cái khó xử của người thiết kế.

Để thực hiện thiết kế dựa trên Web một cách có hiệu quả, người thiết kế Web nên làm việc để dùng lại bốn phần tử kĩ thuật:

Các nguyên lí và phương pháp thiết kế. Điều quan trọng cần lưu ý là các khái niệm và nguyên lí thiết kế được áp dụng cho cả mọi ứng dụng Web. Tính modul hiệu quả (biểu lộ tính cố kết cao và việc gắn thấp), che giấu thông tin, soạn thảo từng bước, và các trực cảm thiết kế phần mềm khác dẫn tới hệ thống và ứng dụng dựa trên Web dễ thích ứng, nâng cao, kiểm thử và sử dụng. Phương pháp thiết kế cho các hệ thống hướng sự vật được thảo luận trước đây có thể được dùng lại khi ứng dụng Web được tạo ra. Multimedia xác định các "sự vật" tương tác qua giao thức truyền thông tương tự lỏng lẻo với việc truyền thông báo. Trong thực tế kĩ pháp biểu đồ được đề nghị cho

UML có thể được thích ứng cho việc dùng trong các hoạt động thiết kế cho ứng dụng Web. Bên cạnh đó, nhiều phương pháp thiết kế multimedia đã được đề nghị.

Các qui tắc vàng. Các ứng dụng multimedia tương tác (ứng dụng Web) đã được xây dựng cho hơn thập kỷ qua. Qua thời gian đó, người thiết kế đã phát triển một tập các trực cảm thiết kế (*các qui tắc vàng*) mà có thể được áp dụng lại trong thiết kế ứng dụng mới.

Hình mẫu thiết kế. Như đã lưu ý, hình mẫu thiết kế là cách tiếp cận tổng quát để giải quyết một số vấn đề nhỏ có thể được thích ứng cho một phạm vi các vấn đề chuyên dụng rộng hơn. Trong hoàn cảnh của ứng dụng Web, các hình mẫu thiết kế có thể được áp dụng không chỉ cho các yếu tố chức năng của ứng dụng mà còn cho các tài liệu, đồ họa, và những điểm thẩm mỹ chung cho Website.

Tiêu bản. Tiêu bản có thể được dùng để cung cấp một khuôn khổ khung cho bất kì hình mẫu thiết kế hay tài liệu nào được dùng bên trong ứng dụng Web.

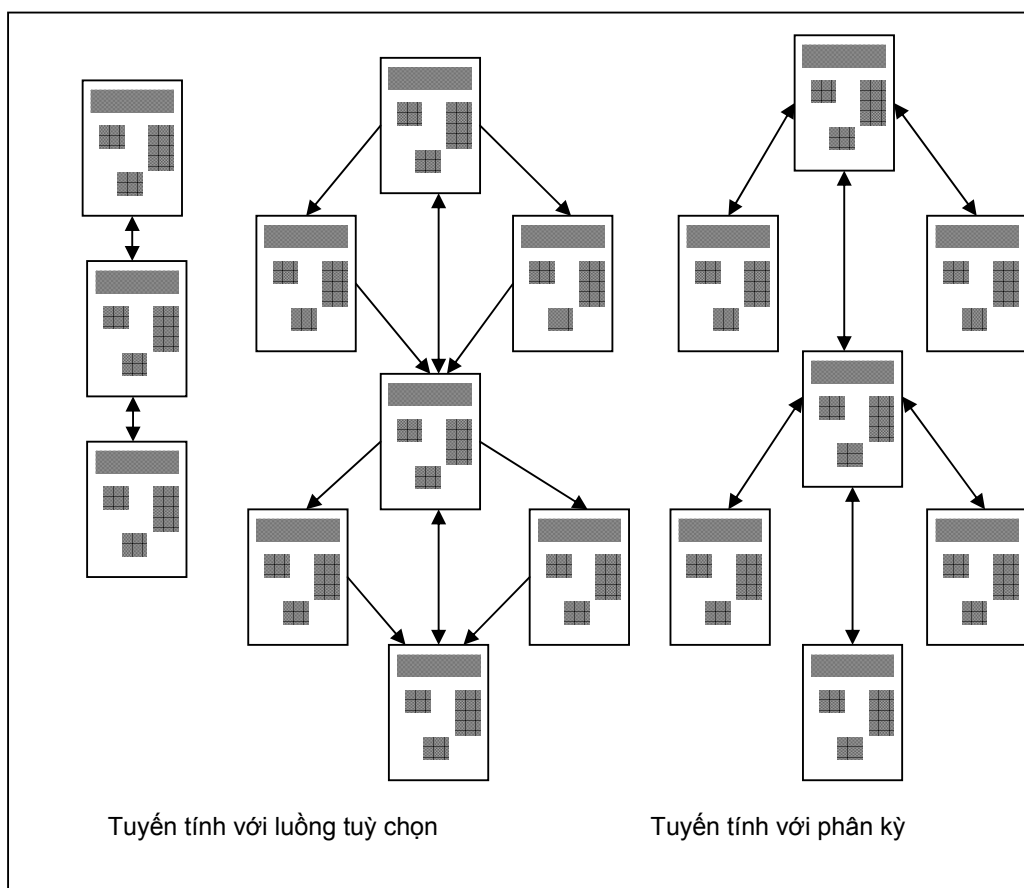
Mỗi một trong bốn phần tử thiết kế này sẽ được thảo luận chi tiết hơn trong các mục tiếp sau đây.

VI.4. 1 Thiết kế kiến trúc

Thiết kế kiến trúc cho hệ thống và ứng dụng dựa trên Web hội tụ vào việc xác định cấu trúc multimedia toàn thể của ứng dụng Web và ứng dụng các hình mẫu thiết kế và tiêu bản xây dựng để làm đầy cấu trúc (và đạt tới việc dùng lại). Một hoạt động song song, được gọi là thiết kế nội dung, dẫn ra cấu trúc toàn thể và cách bố trí chi tiết của nội dung thông tin sẽ được trình bày như một phần của ứng dụng Web.

Cấu trúc ứng dụng Web

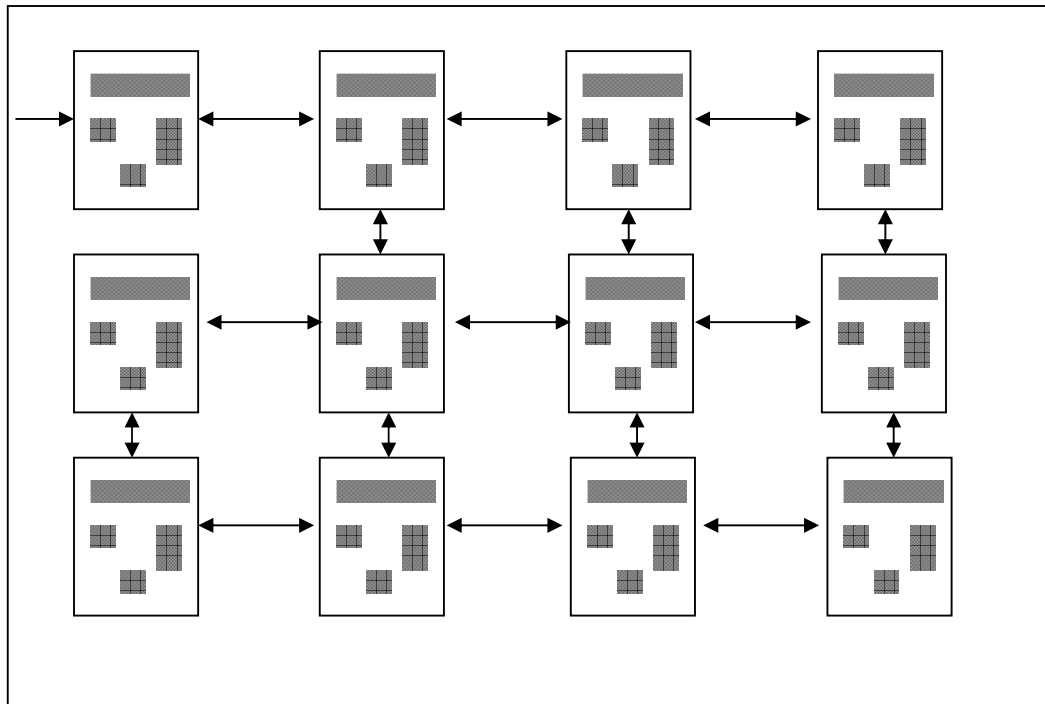
Cấu trúc kiến trúc tổng thể được gắn với các mục tiêu được thiết lập cho ứng dụng Web, nội dung định trình bày, người dùng sẽ viếng thăm, và triết lí dẫn lái đã được thiết lập. Người thiết kế kiến trúc có thể chọn từ bốn cấu trúc khác nhau khi phát triển thiết kế cho một ứng dụng Web điển hình.



H6.2 Cấu trúc tuyến tính

Cấu trúc tuyến tính (Hình 6.2) gặp phải khi một dãy các tương tác dự kiến trước được (với vài biến thể hay phân li) là khá thông dụng. Thí dụ cổ điển có thể là việc trình bày bài giảng trong đó các trang thông tin cùng các đồ họa có liên quan, các đoạn video hay âm thanh ngắn được trình bày chỉ sau khi thông tin cần thiết đã được trình bày. Trình tự của việc trình bày nội dung được định trước và nói chung là tuyến tính. Một thí dụ khác có thể là một trình tự đưa vào đơn mua sản phẩm trong đó thông tin riêng phải được xác định theo một trật tự đặc biệt. Trong những trường hợp như vậy, các cấu trúc được vẽ trên Hình 6.2 là thích hợp. Khi nội dung và xử lý trở nên phức tạp hơn, thì luồng tuyến tính thuần túy được vẽ bên trái của hình vẽ nhường bước cho cấu trúc tuyến tính phức tạp hơn mà trong đó nội dung khác có thể được gọi tới hay một sự phân tán để thu được nội dung phụ (cấu trúc được vẽ ở bên phải của Hình 6.2) sẽ xuất hiện.

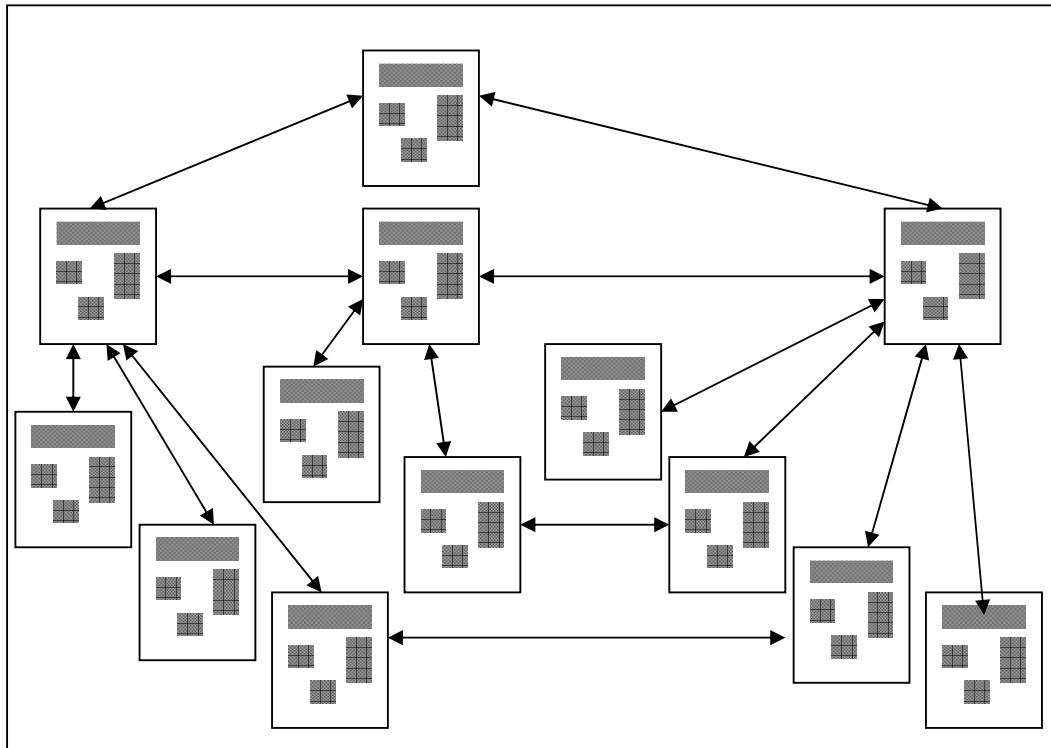
Cấu trúc lưới (Hình 6.3) là tùy chọn kiến trúc mà có thể được áp dụng khi nội dung ứng dụng Web có thể được tổ chức theo loại theo hai hay nhiều chiều. Chẳng hạn, ta xét tình huống trong đó một công ty thương mại điện tử bán gậy chơi golf. Chiều ngang của lưới biểu diễn cho kiểu gậy được bán (như gỗ, sắt, nôm, gậy ngắn) . Chiều đứng biểu diễn cho đề nghị do các nhà chế tạo gậy chơi golf cung cấp. Do đó, người dùng có thể đi qua lưới theo chiều ngang để tìm cột gậy ngắn và rồi theo chiều đứng để xem xét đề nghị do các nhà chế tạo bán gậy ngắn. Kiến trúc ứng dụng Web này là có ích chỉ khi gặp nội dung đều đặn cao độ.



H 6.3 Cấu trúc lưới

Cấu trúc cấp bậc (Hình 6 .4) không hoài nghi gì nữa là kiến trúc ứng dụng Web thông dụng nhất. Không giống như cấp bậc phần mềm được phân hoạch cố vũ cho luồng điều khiển chỉ theo nhánh đứng của cấp bậc, cấu trúc cấp bậc của ứng dụng Web có thể được thiết kế theo cách thức tạo khả năng (thông qua việc phân nhánh siêu văn bản) luồng điều khiển theo chiều ngang, đi qua phân nhánh chiều đứng của cấu trúc. Do đó, nội dung được trình bày trên nhánh bên trái nhất của cấp bậc có thể có móc nối siêu văn bản dẫn tới nội dung tồn tại ở nhánh giữa hay nhánh bên phải của

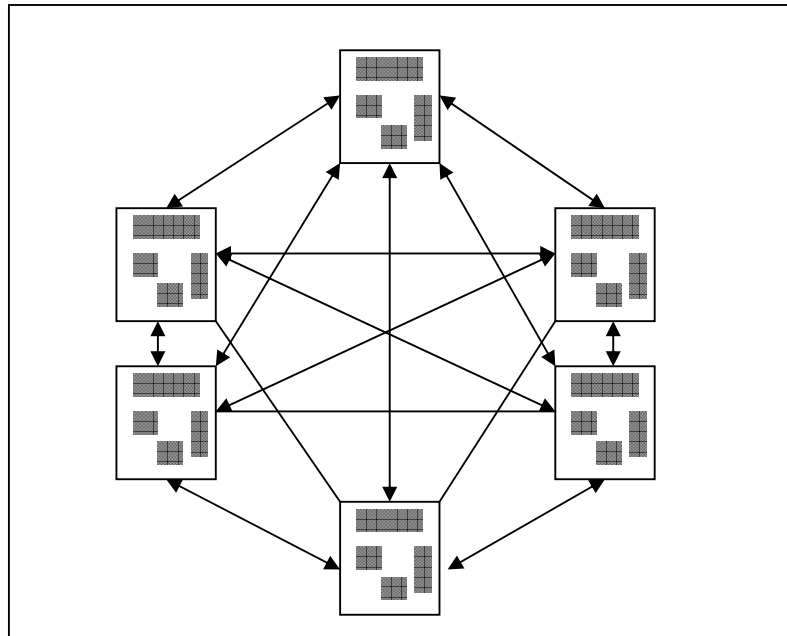
cấu trúc. Tuy nhiên cũng nên lưu ý rằng mặc dầu việc phân nhánh đó cho phép đi nhanh chóng qua nội dung ứng dụng Web, nó có thể dẫn tới lẫn lộn về phần người dùng.



H6.4 Cấu trúc cấp bậc

Cấu trúc nối mạng hay "Web thuần túy" (Hình 6.4) là tương tự theo nhiều cách với kiến trúc tiến hoá cho hệ thống hướng sự vật. Các cấu phần kiến trúc (trong trường hợp này, các trang Web) được thiết kế sao cho chúng có thể truyền điều khiển (qua móc nối siêu văn bản) thực sự tới mọi cấu phần khác trong hệ thống. Cách tiếp cận này cho phép sự linh hoạt di chuyển đáng kể, nhưng đồng thời, có thể gây lẫn lộn cho người dùng.

Cấu trúc kiến trúc được thảo luận trong đoạn trước có thể được tổ hợp để tạo nên cấu trúc hợp thành. Kiến trúc tổng thể của ứng dụng Web có thể là có cấp bậc, nhưng một phần của cấu trúc này có thể để lộ đặc trưng tuyến tính, trong khi phần khác của kiến trúc lại có thể được nối mạng. Mục tiêu cho người thiết kế kiến trúc là sánh đúng cấu trúc của ứng dụng Web với nội dung được trình bày và xử lý được tiến hành.



H6.5 Cấu trúc nối mạng hay "Web thuần túy "

Hình mẫu thiết kế

Như chúng ta đã lưu ý trong cuốn sách này, hình mẫu thiết kế là cách tiếp cận tổng quát cho việc giải quyết vấn đề nhỏ nào đó mà có thể được thích ứng cho một phạm vi rộng các vấn đề đặc thù. Trong hoàn cảnh hệ thống và ứng dụng dựa trên Web, các hình mẫu thiết kế có thể được áp dụng tại mức kiến trúc, mức cấu phần, và tại mức siêu văn bản (dẫn lái).

Khi chức năng xử lý dữ liệu được yêu cầu bên trong ứng dụng Web, thì các hình mẫu thiết kế mức kiến trúc và mức cấu phần có thể áp dụng được. Hình mẫu thiết kế mức siêu văn bản hội tụ vào thiết kế các tính năng dẫn lái cho phép người dùng đi qua nội dung ứng dụng Web theo cách dễ dàng. Trong số nhiều hình mẫu thiết kế siêu văn bản được đề nghị có:

Vòng - một hình mẫu đưa người dùng về nút nội dung đã thăm trước đó.

Vòng Web - hình mẫu cài đặt một "vòng lớn nối toàn bộ siêu văn bản thành một tua các chủ đề".

Đường viền- hình mẫu xuất hiện khi vòng chạm lẫn nhau, cho phép việc dẫn lái đi qua các đường được xác định bởi việc quay vòng.

Đối âm - hình mẫu bổ sung lời bình luận siêu văn bản, ngắt lời dán nội dung để cung cấp thông tin phụ hay góc nhìn.

Thế giới gương - nội dung được trình bày bằng việc dùng các mạch dẫn khác nhau, mỗi mạch với một quan điểm hay viễn cảnh khác nhau. Chẳng hạn, nội dung mô tả một máy tính cá nhân có thể cho phép người dùng chọn lời dẫn "kĩ thuật" hay "không kĩ thuật" mô tả về máy.

Sàng-hình mẫu hướng dẫn người dùng qua hàng loạt tùy chọn (quyết định) để hướng người dùng tới nội dung đặc biệt được chỉ ra bởi dãy các tùy chọn đã lựa hay các quyết định đã đưa ra.

Láng giềng - hình mẫu che phủ một khuôn khổ dẫn lái thống nhất qua tất cả các trang Web để cho phép người dùng có được hướng dẫn dẫn lái nhất quán bất kể tới vị trí bên trong ứng dụng Web.

Các hình mẫu thiết kế siêu văn bản này có thể được dùng lại khi nội dung được dịch thành một định dạng cho phép việc dẫn lái qua ứng dụng Web.

VI.4.2 Thiết kế dẫn lái

Một khi kiến trúc ứng dụng Web đã được thiết lập và các cấu phần (trang, bản viết, tiểu dụng và các chức năng xử lí khác) của kiến trúc đã được nhận diện, thì người thiết kế phải xác định các đường dẫn lái tạo khả năng cho người dùng truy nhập vào nội dung và dịch vụ ứng dụng Web. Để thực hiện điều này người thiết kế phải

(1) nhận diện ngữ nghĩa của việc dẫn lái cho những người dùng khác nhau

(2) Xác định cơ chế (cú pháp) để đạt tới việc dẫn lái

Ứng dụng Web lớn thường có nhiều vai người dùng khác nhau. Chẳng hạn, vai có thể là *khách thăm*, *khách hàng đã đăng ký*, hay *người dùng có đặc quyền*. Mỗi một trong các vai này cũng có thể được gắn với các mức truy nhập nội dung khác nhau và các dịch vụ khác nhau. *Khách thăm* có thể có quyền truy nhập chỉ vào nội dung hạn chế trong khi *khách hàng đã đăng kí* có thể có quyền truy nhập vào phạm vi thông tin và dịch vụ rộng hơn nhiều. Ngữ nghĩa của việc dẫn lái cho từng vai này sẽ khác nhau.

Người thiết kế ứng dụng Web tạo ra *đơn vị dẫn lái ngữ nghĩa* (SNU) cho từng mục tiêu được liên kết với từng vai người dùng. Chẳng hạn, khách hàng đã đăng kí có thể có sáu mục tiêu khác nhau, tất cả đều phát sinh trong việc truy nhập vào thông tin và dịch vụ khác nhau. Một SNU được tạo ra cho từng mục tiêu. Trong những giai đoạn khởi đầu của thiết kế dẫn lái, cấu trúc ứng dụng Web (kiến trúc và cấu phần) được truy nhập tới để xác định một hay nhiều WON (way of navigating) cho mỗi mục tiêu người dùng. Như đã lưu ý, WON nhận diện các nút dẫn lái (tức là các trang) và rồi móc nối việc dẫn lái có thể giữa chúng. WON được tổ chức trong các SNU.

Khi thiết kế tiến hành, các cơ chế cho từng móc nối dẫn lái được nhận diện. Trong nhiều tùy chọn có thể là các móc nối dựa trên văn bản, các biểu tượng, nút và chuyển mạch, các biểu đồ đồ họa. Người thiết kế phải chọn các móc nối dẫn lái thích hợp với nội dung và nhất quán với trực cảm dẫn tới thiết kế giao diện chất lượng cao.

Bên cạnh việc chọn cơ chế dẫn lái, người thiết kế cũng nên thiết lập những qui ước và trợ giúp dẫn lái thích hợp. Chẳng hạn, các biểu tượng và móc nối đồ họa lên có vẻ "bấm được" bằng việc làm xiên góc các cạnh để làm cho hình ảnh có vẻ ba chiều. Phản hồi âm thanh hay trực quan cũng nên được thiết kế để cung cấp cho người dùng một chỉ báo rằng tùy chọn dẫn lái đã được chọn. Với dẫn lái dựa trên văn bản, màu sắc nên để dùng để chỉ ra móc nối dẫn lái và để cung cấp chỉ báo về móc nối đã đi qua. Đây chỉ là một vài trong hàng chục qui ước thiết kế làm cho việc dẫn lái thành thân thiện người dùng. Bên cạnh các qui ước đó, các trợ giúp dẫn lái như bản đồ site, mục lục, chỉ số, cơ chế tìm kiếm và tiện nghi trợ giúp động cũng nên được thiết kế vào lúc này.

VI.4.3 Thiết kế giao diện

Các khái niệm, nguyên lí và phương pháp thiết kế giao diện tất cả đều áp dụng được cho thiết kế giao diện người dùng cho ứng dụng Web. Tuy nhiên, những đặc trưng đặc biệt của hệ thống và ứng dụng dựa trên Web đòi hỏi một số các xem xét phụ thêm.

Giao diện người dùng của ứng dụng Web là "ấn tượng đầu tiên" về nó. Bất kể tới giá trị của nội dung của nó, sự phức tạp trong khả năng xử lí và dịch vụ của nó, và cả ích lợi tổng thể của bản thân ứng dụng Web, một giao diện được thiết kế nghèo nàn sẽ

làm thất vọng người dùng tiềm năng và trong thực tế còn có thể làm cho người dùng bỏ đi đâu đó khác. Bởi vì khối lượng cực lớn các ứng dụng Web cạnh tranh đang có thực trong mọi dùng chủ đề, nên giao diện phải "bắt" ngay người dùng tiềm năng. Nielsen và Wagner gợi ý và hướng dẫn đơn giản dựa trên việc thiết kế lại của họ đối với phần lớn các ứng dụng Web:

- Lỗi nguồn phục vụ, cho dù là lỗi nhỏ, rất có thể làm cho người dùng bỏ vị trí Web và nhìn đâu đó tìm thông tin và dịch vụ.
- Tốc độ đọc trên màn hình máy tính phải chậm xấp xỉ 25 phần trăm hơn tốc độ đọc trên giấy in. Do đó, đừng buộc người dùng đọc khối lượng lớn văn bản, đặc biệt khi văn bản giải thích việc vận hành của ứng dụng Web hay trợ giúp dẫn lối.
- Tránh dấu hiệu "đang xây dựng" - chúng làm tăng hi vọng và gây ra những móc nối không cần thiết mà chắc chắn làm thất vọng.
- Người dùng ưa chuộng không cuộn màn hình. Thông tin quan trọng nên được đặt bên trong các chiều của cửa sổ duyệt điển hình.
- Menu dẫn lối và thanh tiêu đề nên được thiết kế nhất quán và nên có sẵn trên mọi trang có sẵn cho người dùng. Thiết kế không nên chỉ dựa vào các chức năng duyệt để trợ giúp cho dẫn lối.
- Tính thẩm mỹ không bao giờ nên che lấp tính chức năng. Chẳng hạn, một nút đơn giản có thể là tùy chọn dẫn lối còn tốt hơn một hình ảnh hay biểu tượng đẹp nhưng mơ hồ có nội dung không rõ ràng.
- Các tùy chọn dẫn lối nên hiển nhiên, ngay cả với người dùng tình cờ. Người dùng không nên bị bắt phải tìm trên màn hình để xác định cách móc nối tới nội dung hay dịch vụ khác.

Một giao diện thiết kế tốt làm cải thiện cảm nhận của người dùng về nội dung hay dịch vụ do Site cung cấp. Nó không nhất thiết phải hào nhoáng, nhưng nó bao giờ cũng nên được cấu trúc tốt và có vẻ mang tính công thái học. Thảo luận sâu sắc về giao diện người dùng ứng dụng Web ở ngoài phạm vi của cuốn giáo trình này.

VI.5 Kiểm thử ứng dụng dựa trên Web

Chúng ta đã lưu ý rằng kiểm thử là tiến trình luyện thử phần mềm với ý định tìm ra (và chung cuộc sửa) lỗi. Triết lí nền tảng này không thay đổi cho ứng dụng Web. Trong thực tế, bởi vì hệ thống và ứng dụng dựa trên Web nằm trên mạng và liên tác với nhiều hệ điều hành khác, các trình duyệt, các nền phần cứng và các giao thức truyền thông, nên việc tìm lỗi biểu thị một thách thức có ý nghĩa cho người kĩ sư Web.

Cách tiếp cận tới việc kiểm thử Web chấp nhận các nguyên lí cơ bản cho mọi kiểm thử Web và áp dụng các chiến lược và chiến thuật đã được khuyến cáo cho hệ thống hướng sự vật. Các bước sau đây tóm tắt cách tiếp cận này:

1 . Mô hình nội dung cho ứng dụng Web được xem xét để phát hiện ra lỗi.

Hoạt động "kiểm thử" này là tương tự theo nhiều khía cạnh với việc biên tập một tài liệu viết. Trong thực tế, một Web site lớn có thể thu nhận các dịch vụ biên tập chuyên nghiệp để phát hiện các lỗi in ấn, lỗi chính tả, lỗi nhất quán nội dung, lỗi biểu diễn đồ hoạ, và các lỗi tham chiếu chéo.

2. Mô hình thiết kế cho ứng dụng Web được xem xét để phát hiện lỗi dẫn lái. Các trường hợp sử dụng, được suy dẫn như một phần của hoạt động phân tích, cho phép người kĩ sư Web thực tập từng kịch bản sử dụng theo thiết kế kiến trúc và dẫn lái. Về bản chất, những kiểm thử không thực hiện này giúp phát hiện ra lỗi trong dẫn lái (như một trường hợp khi người dùng không thể đạt tới nút dẫn lái). Bên cạnh đó, các móc nối dẫn lái được xem xét để đảm bảo rằng chúng tương ứng với những dẫn lái đã đặc tả trong từng SNU cho từng nút người dùng.

3. Các cấu phần xử lí đã lựa và các trang Web được kiểm thử. Khi ứng dụng Web được xem xét, khái niệm về đơn vị thay đổi. Mỗi trang Web bao bọc nội dung, móc nối dẫn lái và phần tử xử lí (mẫu, chữ viết, tiểu dụng). Không phải bao giờ cũng có thể hay thực tế kiểm thử được từng đặc trưng này một cách riêng biệt. Trong nhiều trường hợp, đơn vị kiểm thử nhỏ nhất là trang Web. Không giống như việc kiểm thử đơn vị của phần mềm qui ước, có khuynh hướng hội tụ vào chi tiết thuật toán của modul và dữ liệu chảy qua giao diện modul, việc kiểm thử mức trang cho ứng dụng Web được hướng theo nội dung, xử lí và móc nối được bao bọc bởi trang Web.

4. Kiến trúc được xây dựng và kiểm thử tích hợp được tiến hành. Chiến lược cho việc kiểm thử tích hợp tuy thuộc kiến trúc đã được chọn cho ứng dụng Web. Nếu ứng dụng Web đã được thiết kế theo cấu trúc tuyến tính, tới hay cấp bậc đơn giản, thì có khả năng tích hợp các trang Web rất giống cách như chúng ta tích hợp các môđun cho phần mềm qui ước. Tuy nhiên, nếu kiến trúc cấp bậc hỗn hợp hay mạng (Web) được dùng, thì kiểm thử tích hợp lại tương tự với cách tiếp cận được dùng cho hệ thống hướng sự vật. Việc kiểm thử dựa trên mạch có thể được dùng để tích hợp tập các trang Web (một SNU có thể được dùng để xác định tập thích hợp) được yêu cầu để đáp ứng cho một biến cố người dùng. Mỗi mạch lại được tích hợp và kiểm thử riêng rẽ. Kiểm thử rà lại được áp dụng để đảm bảo rằng không hiệu quả phụ nào xuất hiện. Kiểm thử cụm tích hợp một tập các trang hợp tác (được xác định bằng việc xem xét các trường hợp sử dụng và SNU). Các trường hợp kiểm thử được suy dẫn để phát hiện lỗi trong sự hợp tác.

5. Ứng dụng Web đã lắp ráp được kiểm thử cho chức năng toàn thể và việc chuyển giao nội dung. Giống như việc làm hợp lệ qui ước, việc làm hợp lệ các hệ thống và ứng dụng dựa trên Web hội tụ vào những hành động người dùng thấy được và cái ra người dùng nhận biết được từ hệ thống. Để trợ giúp trong việc suy ra các kiểm thử hợp lệ, người kiểm thử phải dựa vào các trường hợp sử dụng. Trường hợp sử dụng cung cấp kịch bản có nhiều khả năng làm lộ ra lỗi trong yêu cầu tương tác với người sử dụng.

6. Ứng dụng Web được cài đặt trong nhiều cấu hình môi trường khác nhau và được kiểm thử cho tính tương hợp với từng cấu hình. Ma trận tham chiếu chéo xác định tất cả các hệ điều hành có thể có, các trình duyệt, các nền phần cứng và các giao thức truyền thông được tạo ra. Thế rồi việc kiểm thử được tiến hành để phát hiện lỗi liên kết với từng cấu hình có thể có.

7. Ứng dụng Web được kiểm thử bằng việc người dùng điều phối và kiểm soát. Một số đông người dùng bao gồm mọi vai người dùng có thể có được chọn ra. Ứng dụng Web được thực tập bởi những người dùng này và kết quả tương tác của họ với hệ thống được đánh giá về lỗi nội dung và việc dẫn lái, mối quan tâm sử dụng, mối quan tâm tương hợp, và tính tin cậy của ứng dụng Web cùng hiệu năng.

Bởi vì nhiều ứng dụng Web tiến hoá liên tục nên tiến trình kiểm thử là hoạt động tiếp diễn, do các nhân viên hỗ trợ Web tiến hành, những người dùng kiểm thử rà lại được suy dẫn ra từ các kiểm thử được phát triển khi ứng dụng Web lần đầu tiên được chế tạo

VI.6 Vấn đề quản lí

Với tính tức khắc của ứng dụng Web, điều hợp lí là hỏi: "Chúng ta có thực sự cần dành thời gian ra quản lí nỗ lực ứng dụng Web không? Chúng ta có nên cứ để cho ứng dụng Web tiến hoá một cách tự nhiên, với ít hay không có việc quản lí tương minh nào?" Một số người phát triển Web sẽ chọn ít hay không có việc quản lí nào, nhưng điều đó không làm cho họ là đúng!

Kĩ nghệ Web là một hoạt động kĩ thuật phức tạp. Nhiều người được tham dự vào đó, thường làm việc song song. Việc tổ hợp các nhiệm vụ kĩ thuật và phi kĩ thuật phải xuất hiện (đúng hạn và trong khuôn khổ ngân sách) để cung cấp ứng dụng Web chất lượng cao biểu thị một thách thức cho bất kì nhóm chuyên môn nào. Để tránh lẩn lộn, thất vọng, và thất bại, việc lập kế hoạch phải có, rủi ro phải được tính tới, lịch biểu phải được thiết lập và theo dõi, và kiểm soát phải được xác định. Những điều này là những hoạt động lõi mà chúng ta đã gọi là *quản lí dự án*.

VI. 6. 1 Tổ kĩ nghệ Web

Việc tạo ra ứng dụng Web thành công đòi hỏi một phạm vi rộng các kĩ năng. Tổ kĩ nghệ Web có thể được tổ chức rất giống cách tổ phần mềm được tổ chức. Tuy nhiên, người tham dự và vai trò của họ lại thường rất khác nhau. Trong số nhiều kĩ năng phải được phân bổ trong các thành viên tổ kĩ nghệ Web thì có kĩ nghệ phần mềm dựa trên cấu phần, mạng, thiết kế kiến trúc và dẫn lái, chuẩn/ngôn ngữ Internet, thiết kế giao diện con người, thiết kế đồ hoạ, bố trí nội dung, và kiểm thử ứng dụng Web. Các vai trò sau đây nên được phân bổ giữa các thành viên của tổ kĩ nghệ Web:

Người phát triển và người cung cấp nội dung. Bởi vì ứng dụng Web là hướng nội dung một cách nhất quán, nên vai trò thành viên tổ kĩ nghệ Web phải hội tụ vào

việc sinh ra và thu thập nội dung. Nhớ lại rằng nội dung trải ra trên một phạm vi rộng các sự vật dữ liệu, người phát triển và người cung cấp nội dung có thể xuất thân từ nhiều bối cảnh khác nhau (phi phần mềm). Chẳng hạn, nhân viên tiếp thị và bán hàng có thể cung cấp thông tin sản phẩm và hình ảnh đồ họa, người tạo môi giới có thể cung cấp video và âm thanh, người thiết kế đồ họa có thể cung cấp thiết kế bố trí và nội dung thẩm mỹ, người viết bài quảng cáo có thể cung cấp nội dung dựa trên văn bản. Bên cạnh đó, nhân viên nghiên cứu có thể được cần tới để tìm và định dạng nội dung bên ngoài để thay thế hay tham chiếu bên trong ứng dụng Web.

Người xuất bản Web. Nội dung đa dạng được những người phát triển và cung cấp nội dung sinh ra phải được tổ chức lại để đưa vào bên trong ứng dụng Web. Bên cạnh đó, ai đó phải hành động như người liên hệ giữa nhân viên kỹ thuật chế tạo ra ứng dụng Web và người cung cấp và phát triển nội dung phi kỹ thuật. Vai trò này được đáp ứng bởi người xuất bản Web, người phải hiểu cả nội dung và công nghệ ứng dụng Web kể cả HTML (hay việc mở rộng thể hệ tiếp của nó như XML), chức năng cơ sở dữ liệu, chữ viết, và việc dẫn lái website nói chung.

Kỹ sư Web. Kỹ sư Web trở nên được tham dự vào một phạm vi rộng các hoạt động trong việc phát triển ứng dụng Web kể cả khâu gọi yêu cầu; lập mô hình phân tích; thiết kế kiến trúc, dẫn lái và giao diện; cài đặt ứng dụng Web; và kiểm thử. Người kỹ sư Web cũng nên có hiểu biết vững chắc về công nghệ cấu phần, kiến trúc khách/nguồn phục vụ, HTML/XML, và công nghệ cơ sở dữ liệu cũng như tri thức làm việc về các khái niệm multimedia, nền phần cứng/phần mềm, an ninh mạng, và các vấn đề hỗ trợ Web site.

Chuyên viên hỗ trợ. Vai trò này được phân bổ cho người có trách nhiệm tiếp tục sự hỗ trợ ứng dụng Web. Bởi vì ứng dụng Web liên tục tiến hoá nên chuyên viên hỗ trợ chịu trách nhiệm cho việc sửa chữa, thích nghi và nâng cao site, kể cả cập nhật nội dung, cài đặt các thủ tục và mẫu mới, và thay đổi hình mẫu dẫn lái.

Người quản trị. Thường được gọi là Web master, ông chủ Web, người này có trách nhiệm vận hành hàng ngày ứng dụng Web, bao gồm

- Phát triển và cài đặt chính sách cho việc vận hành ứng dụng Web.
- Thiết lập các thủ tục hỗ trợ và phản hồi.

- Thực hiện các thủ tục an ninh và quyền truy nhập.
- Đo và phân tích lưu thông Web.
- Điều phối thay đổi các thủ tục kiểm soát.
- Điều phối các chuyên viên hỗ trợ

Người quản trị cũng có thể tham gia vào các hoạt động kĩ thuật do người kĩ sư Web và chuyên viên hỗ trợ thực hiện.

VI. 6. 2 Quản lý dự án

Về mặt lí thuyết hầu hết (nếu không nói là tất cả) các hoạt động quản lí dự án đã thảo luận trong các chương trước đều áp dụng được cho dự án kĩ nghệ Web. Nhưng trong thực hành, cách tiếp cận kĩ nghệ Web tới việc quản lí dự án là khác biệt đáng kể.

VI.6.2.1 Khởi đầu dự án

Cho dù việc khoán ngoài là chiến lược được chọn cho việc phát triển ứng dụng Web, một tổ chức phải thực hiện một số các. nhiệm vụ trước khi tìm nhà cung cấp khoán ngoài để làm công việc này:

1. Nhiều hoạt động phân tích nên được thực hiện nội bộ. Khách giả cho ứng dụng Web nên được nhận diện; những người bảo trợ nội bộ, người có thể có mối quan tâm tới ứng dụng Web được liệt kê ra; những mục tiêu tổng thể cho ứng dụng Web được xác định và xem xét; thông tin và dịch vụ được chuyển giao bởi ứng dụng Web được xác định; các Web site cạnh tranh được lưu ý tới; và tránh đo" định tính và định lượng về ứng dụng Web thành công được xác định. Thông tin này nên được làm tài liệu trong đặc tả sản phẩm.

2. Thiết kế đại thể cho ứng dụng Web nên được phát triển nội bộ. Hiển nhiên, người phát triển Web cỡ chuyên gia sẽ tạo ra một thiết kế đầy đủ, nhưng thời gian và chi phí có thể được tiết kiệm nếu góc nhìn chung về ứng dụng Web được vạch ra cho người làm khoán ngoài (điều này bao giờ cũng có thể được sửa đổi trong giai đoạn sơ bộ của dự án). Thiết kế phải bao hàm một chỉ báo về kiểu và khối lượng nội dung

được ứng dụng Web trình bày và kiểu xử lý tương tác (như các mẫu, đa vào đơn) cần được thực hiện. Thông tin này nên được bổ sung vào đặc tả sản phẩm.

3. *Một lịch dự án đại thể, không những chứa ngày tháng chuyển giao cuối cùng mà còn cả những ngày tháng cột mốc, phải được xây dựng ra.* Cột mốc nên được gắn vào phiên bản chuyển giao được của ứng dụng Web khi nó tiến hoá.

4. *Mức độ giám sát và tương tác bởi người hợp đồng với nhà cung cấp nên được định rõ.* Điều này nên bao hàm việc chỉ tên người liên hệ của nhà cung cấp và xác định trách nhiệm cùng chủ quyền của người liên hệ, xác định các điểm kiểm điểm chất lượng khi việc phát triển được thực hiện, và trách nhiệm của nhà cung cấp đối với truyền thông liên tổ chức.

VI.6.2.2 Lựa nhà cung cấp ứng cử viên

Trong những năm gần đây, hàng nghìn công ty “thiết kế Web” đã nổi lên để giúp các doanh nghiệp thiết lập sự hiện diện Web hay tham gia vào thương mại điện tử. Nhiều người đã trở thành lão luyện với tiến trình kỹ nghệ Web, nhưng nhiều người khác vẫn chỉ như một tài tử. Để tuyển người phát triển Web, người có hợp đồng phải siêng năng thực hiện :

(1) phỏng vấn khách hàng quá khứ để xác định tính chuyên nghiệp của người cung cấp Web, khả năng đáp ứng những cam kết lịch biểu và chi phí, và khả năng trao đổi có hiệu quả;

(2) xác định tên của kỹ sư Web chính của nhà cung cấp về những dự án quá khứ thành công (và về sau, hãy chắc chắn rằng người này có nghĩa vụ về mặt hợp đồng để được tham gia vào dự án của bạn); và (3) xem xét cẩn thận các mẫu công việc của nhà cung cấp tương tự về góc nhìn và cảm giác (và miền nghiệp vụ) với ứng dụng Web dự định ký hợp đồng. Thậm chí trước khi yêu cầu về giá được đưa ra, cuộc họp mặt đối mặt có thể cung cấp góc nhìn chủ chốt vào "sự thích hợp" giữa người có hợp đồng và nhà cung cấp.

VI.6.2.3 Thẩm định tính hợp lệ về giá và độ tin cậy của các ước lượng

Bởi vì còn tương đối ít dữ liệu lịch sử và phạm vi của ứng dụng Web còn hay thay đổi dễ sợ, nên việc ước lượng mang tính rủi ro cố hữu. Bởi lí do này, một số nhà cung cấp sẽ giữ lấy lẽ an toàn chủ yếu trong giá cho dự án. Điều này vừa là hiểu được và thích hợp.

VI.6.2.4 Mức độ quản lí dự án có thể trông đợi hay thực hiện

Tính hình thức liên kết với các nhiệm vụ quản lí dự án (được thực hiện bởi cả người cung cấp và người hợp đồng) tỉ lệ với kích thước, chi phí và độ phức tạp của ứng dụng Web. Với các dự án phức tạp, lớn, một lịch biểu dự án chi tiết xác định ra các nhiệm vụ công việc, các điểm kiểm tra SQA, các sản phẩm công việc đã làm, các điểm kiểm điểm của khách hàng, và những cột mốc chính nên được phát triển. Người cung cấp và người hợp đồng nên thẩm định các rủi ro cùng nhau và phát triển những kế hoạch để di chuyển, điều phối và quản lí những rủi ro có vẻ quan trọng. Các cơ chế cho việc đảm bảo chất lượng và kiểm soát thay đổi nên được xác định tường minh dưới dạng viết. Các phương pháp cho việc trao đổi hiệu quả giữa người hợp đồng và người cung cấp nên được thiết lập.

VI.6.2.5 Thẩm định lịch phát triển

Bởi vì lịch phát triển ứng dụng Web trải ra trong một thời kì tương đối ngắn (thường ít hơn một hay hai tháng), nên lịch phát triển nên có độ đóng cục cao. Tức là các nhiệm vụ công việc và những cột mốc nhỏ nên được lập lịch hàng ngày. Việc đóng cục thời gian này cho phép cả nhà cung cấp và người hợp đồng nhận ra việc trượt lịch trước khi nó đe dọa tới ngày tháng hoàn thành.

VI.6.2.6 Quản lí phạm vi

Rất có thể là phạm vi sẽ thay đổi khi dự án ứng dụng Web tiến triển, do vậy mô hình tiến trình kĩ nghệ Web nên tăng dần. Điều này cho phép tổ phát triển "làm đông cứng" phạm vi cho một lần tăng để cho ra ứng dụng Web sớm sửa. Lần tăng tiếp có thể đề cập tới những thay đổi phạm vi được gợi ý bởi việc duyệt xét lần tăng trước, nhưng một khi lần tăng thứ hai bắt đầu, thì phạm vi lại bị làm đông cứng tạm thời. Cách tiếp cận này tạo khả năng tổ kĩ nghệ Web làm việc không phải điều tiết luồng thay đổi liên tục nhưng vẫn nhận ra đặc trưng tiến hoá liên tục của hầu hết ứng dụng Web.

Những hướng dẫn này không có chủ ý hướng dẫn cho việc tạo ra ứng dụng Web giá thấp, đúng hạn. Tuy nhiên, chúng sẽ giúp cho cả người hợp đồng và nhà cung cấp khởi đầu công việc một cách trôi chảy với tối thiểu việc hiểu lầm.

CHƯƠNG V

SƠ LƯỢC VỀ UML (Unified Modeling Language)

I. Giới thiệu

Ngôn ngữ mô hình hoá thống nhất (UML) là một ngôn ngữ để:

- ♦ đặc tả
- ♦ trực quan hoá
- ♦ xây dựng
- ♦ làm tài liệu

cho các sản phẩm của một hệ thống phần mềm chuyên sâu.

UML là một ngôn ngữ đồ hoạ, hệ thống ký pháp đồ hoạ và văn bản của UML cho phép biểu diễn được đầy đủ thông tin của hệ thống.

UML bao gồm ba phần tử chính:

- ♦ Bảng từ vựng (các khối xây dựng cơ bản)
- ♦ Các quy tắc (chỉ ra các khối xây dựng có thể đặt bên nhau như thế nào)
- ♦ Các cơ chế chung

II. Bảng từ vựng của UML

Các khối xây dựng cơ bản hay bảng từ vựng của UML bao gồm:

- ♦ Các sự vật (things): là những trừu tượng hoá và cũng là những thành viên đầu tiên trong mô hình
- ♦ Các quan hệ (relationships): gắn kết các sự vật với nhau
- ♦ Các biểu đồ (diagrams): nhóm những sự vật có liên quan lại với nhau

II.1 Các sự vật

Các sự vật trong UML bao gồm:

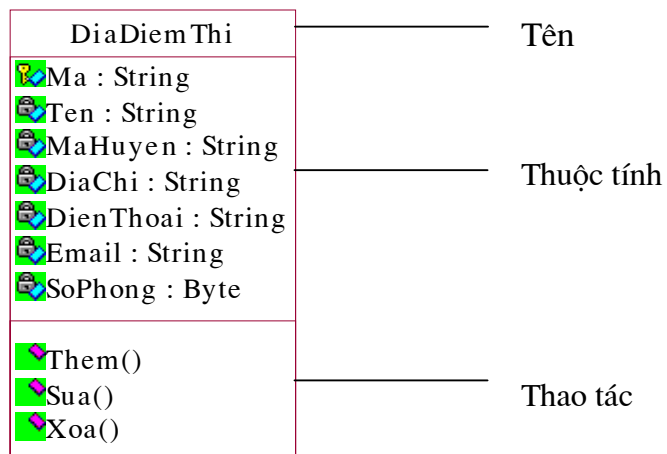
- ◆ Các sự vật cấu trúc (structural things)
- ◆ Các sự vật hành vi (behavioral things)
- ◆ Các sự vật nhóm (grouping things)
- ◆ Các sự vật giải thích (annotational things)

Các sự vật cấu trúc: là những phần hữu như tính của mô hình, biểu diễn các phần tử vật lý hay khái niệm.

Có 7 loại sự vật cấu trúc:

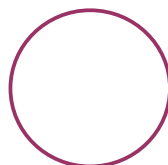
– *Lớp (class)*: mô tả một tập hợp các đối tượng có chung các thuộc tính và các hành vi. Một số mẫu thường gặp cho các lớp là: lớp biên (còn gọi là lớp giao diện), lớp thực thể, lớp điều khiển

Kí hiệu:



– *Giao diện (interface)*: mô tả một tập các thao tác đặc tả những dịch vụ của một lớp hay một thành phần, nhưng không phải là tập hợp các thực hiện thao tác.

Kí hiệu:



Giao diện

– *Lớp hoạt động (active class)*: là một lớp mà các đối tượng của nó sở hữu một hoặc một số tiến trình hoặc các xâu.

– *Sự cộng tác (collaboration)*: xác định một tác động bên trong, là một bộ các nguyên tắc và các phần tử cùng làm việc để cung cấp một hành vi hợp tác lớn hơn tổng của tất cả các phần tử. Sự cộng tác biểu diễn sự thực hiện các thành phần của hệ thống.

– *Ca sử dụng (use case)*: là một tập các chuỗi hành động mà hệ thống thực hiện để cho một kết quả là giá trị quan sát được đối với một tác nhân. Use Case thường được sử dụng để mô hình hoá các tiến trình công việc.

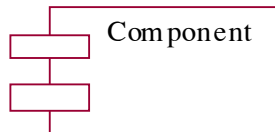
Kí hiệu:



Xoá địa điểm thi

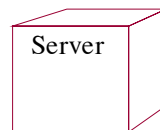
– *Thành phần (component)*: là một bộ phận vật lý có thể thay thế được của một hệ thống, cung cấp sự thực hiện một tập các giao diện.

Kí hiệu:



– *Nút (node)*: là một phần tử vật lý tồn tại trong thời gian thực hiện với khả năng tính toán hoặc sử dụng tài nguyên hệ thống.

Kí hiệu:



Ngoài 7 loại trên, còn có những biến thể như: tác nhân, tín hiệu và tiện ích (thuộc lớp), các tiến trình, các dãy (thuộc lớp hoạt động), các tài liệu, các file, các thư viện, các trang, các bảng (thuộc thành phần)

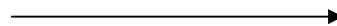
Các sự vật hành vi: là những bộ phận động trong các mô hình UML, biểu diễn những hành vi diễn ra trong không gian và thời gian

Có 2 loại sự vật hành vi chính:

– *Sự tương tác (interaction)*: là hành vi bao gồm tập các thông điệp trao đổi giữa các đối tượng trong một khung cảnh cụ thể để thực hiện một mục tiêu xác định.

Kí hiệu của thông điệp:

Thông điệp



Biểu diễn giá trị trả lại: thông điệp gửi tới một đối tượng có thể yêu cầu một giá trị trả lại cho đối tượng gửi. Giá trị trả lại được đặt trước thông điệp với một biến giá trị trả lại và toán tử (':=')

Biểu diễn lặp: một đối tượng có thể gửi các thông điệp tới các đối tượng khác lặp đi lặp lại. Điều này được biểu diễn bằng cách thêm vào trước thông điệp dấu (*).

– *Máy trạng thái (state machine)*: là một hành vi xác định một dãy các trạng thái liên tục mà một đối tượng hay một tương tác trải qua trong vòng đời tương ứng với các sự kiện.

Các sự vật nhóm: là những bộ phận tổ chức của UML, một mô hình có thể được phân rã thành các gói.

Có 1 loại sự vật nhóm đó là gói.

Gói (package): là một cơ chế để tổ chức các phần tử thành các nhóm, các sự vật cấu trúc, các sự vật hành vi và cả sự vật nhóm đều có thể được đặt trong một gói.

Các biến thể của gói là: *khung làm việc (framework)*, *mô hình (model)*, *hệ thống con (subsystem)*

Kí hiệu:

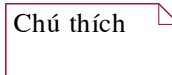


Các sự vật giải thích: là các chú thích của các mô hình UML, có thể dùng để mô tả, giải thích hay đánh dấu một phần tử bất kì trong mô hình.

Có 1 loại sự vật giải thích là chú thích

Chú thích (note): là một biểu tượng để ghi những hạn chế hay bình luận gắn với một phần tử hay một bộ phận các phần tử.

Kí hiệu:



II.2 Các quan hệ

Các quan hệ trong UML gồm có 4 loại:

Quan hệ phụ thuộc (dependency)

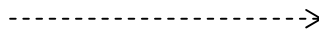
Quan hệ liên kết (association)

Quan hệ tổng quát hoá (generalization)

Quan hệ thực hiện (realization)

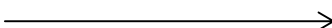
Quan hệ phụ thuộc: là mối *quan hệ ngữ nghĩa* giữa hai sự vật, trong đó sự thay đổi của một sự vật có thể tác động lên sự thay đổi ngữ nghĩa của sự vật kia.

Kí hiệu:



Quan hệ liên kết: là mối *quan hệ cấu trúc*, mô tả một tập các mối liên kết kết nối một số các đối tượng.

Kí hiệu:

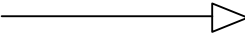


Các giá trị chỉ số lượng đối tượng mỗi bên tham gia vào liên kết có thể được ghi thêm tại mỗi đầu mút của liên kết. Các giá trị này có thể là:

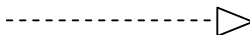
1 tương ứng với một

1..n	tương ứng với một hoặc nhiều
0..n	tương ứng với không hoặc nhiều
0..1	tương ứng với không hoặc một
m..n	chỉ một giới hạn cụ thể

Quan hệ tổng quát hoá: là một mối *quan hệ tổng quát hoá/cụ thể hoá*, trong đó các đối tượng cụ thể hoá thay thế được cho các đối tượng của phần tử được tổng quát hoá.

Kí hiệu: 

Quan hệ thực hiện: là mối quan hệ giữa phần đặc tả và phần triển khai của một lớp, giữa ca sử dụng và sự cộng tác để thực hiện nó.

Kí hiệu: 

II.3 Các biểu đồ

Biểu đồ là một biểu diễn đồ thị của một tập các phần tử, gồm các đỉnh (các sự vật) và các cạnh (các quan hệ). Biểu đồ là một thể hiện của hệ thống, được dùng để làm trực quan một hệ thống.

UML có 9 loại biểu đồ:

- Biểu đồ lớp (class diagram)
- Biểu đồ đối tượng (object diagram)
- Biểu đồ ca sử dụng (use Case diagram)
- Biểu đồ tuần tự (sequence diagram)
- Biểu đồ cộng tác (collaboration diagram)
- Biểu đồ trạng thái (statechart diagram)
- Biểu đồ hoạt động (activity diagram)
- Biểu đồ thành phần (component diagram)
- Biểu đồ triển khai (deployment diagram)

Biểu đồ lớp: biểu diễn một tập các lớp, các giao diện cùng với các sự cộng tác và các mối quan hệ của chúng. Đây là biểu đồ chung nhất trong mô hình hoá các hệ thống hướng đối tượng, nó hướng đến cấu trúc tĩnh của một hệ thống.

Một hệ thống có thể bao gồm một hoặc nhiều biểu đồ lớp nhưng không phải tất cả các lớp đều xuất hiện.

Biểu đồ ca sử dụng: biểu diễn một tập các Use Case, các tác nhân và mối quan hệ giữa chúng. Các biểu đồ này đặc biệt quan trọng trong việc tổ chức và mô hình hoá hành vi của hệ thống.

Biểu đồ tuần tự: biểu diễn một tương tác, gồm một tập các đối tượng và các mối quan hệ giữa chúng. Các mối quan hệ có thể bao gồm các thông điệp chuyển đi giữa các đối tượng. Biểu đồ tuần tự *nhấn mạnh đến trình tự thời gian* của các thông điệp được chuyển đi giữa các đối tượng.

Biểu đồ cộng tác: biểu diễn một tương tác, gồm một tập các đối tượng và các mối quan hệ giữa chúng. Các mối quan hệ có thể bao gồm các thông điệp chuyển đi giữa các đối tượng. Biểu đồ cộng tác *nhấn mạnh việc tổ chức cấu trúc* của các đối tượng gửi và nhận

III. Các quy tắc của UML

Các khối được xây dựng của UML không thể được ghép lại một cách ngẫu nhiên mà phải tuân theo một số quy tắc thì mới tạo được một mô hình tốt.

Một mô hình có cấu trúc tốt là mô hình nhất quán về mặt ngữ nghĩa và hoà hợp với tất cả các mô hình có liên quan.

III.1 Các quy tắc ngữ nghĩa

UML có các quy tắc ngữ nghĩa sau:

- Tên: để gọi các sự vật, các mối quan hệ và các biểu đồ.
- Phạm vi: ngữ cảnh cho ý nghĩa cụ thể của một tên gọi.
- Tính trực quan: tên của một sự vật có thể được cá sự vật khác nhìn thấy và sử dụng.
- Tính toàn vẹn: các sự vật liên kết với các sự vật khác một cách nhất quán và phù hợp.

- Tính thực thi: nói đến việc vận hành và mô phỏng một mô hình động

III.2 Các quy tắc xây dựng mô hình

Các mô hình được xây dựng theo nhiều cách, do đó không phải luôn đảm bảo cấu trúc hoàn chỉnh. Vì vậy, các mô hình xây dựng có thể:

- Được lược bớt: một số phần tử được giấu đi làm đơn giản khung nhìn
- Không đầy đủ: một số phần tử có thể bỏ qua
- Không chắc chắn: tính toàn vẹn của mô hình không đảm bảo

IV. Các cơ chế chung

UML có 4 cơ chế chung được áp dụng trong toàn bộ quá trình xây dựng mô hình, đó là:

- Đặc tả (specification)
- Bài trí (adornment)
- Phân hoạch chung (common divisions)
- Các cơ chế mở rộng (extensibility mechanism)

Đặc tả: UML mạnh hơn cả một ngôn ngữ đồ hoạ; đằng sau mỗi kí pháp đồ hoạ có một đặc tả cung cấp một mệnh đề văn bản theo cú pháp và ngữ nghĩa của khối đó.

Bài trí: trong UML hầu hết các phần tử có kí hiệu đồ hoạ duy nhất, cung cấp một sự thể hiện trực quan về các khía cạnh quan trọng nhất của phần tử đó.

Phân hoạch chung: trong mô hình hoá hệ thống hướng đối tượng, mọi sự vật thường được phân thành hai cặp: lớp/ đối tượng và giao diện/thực thi

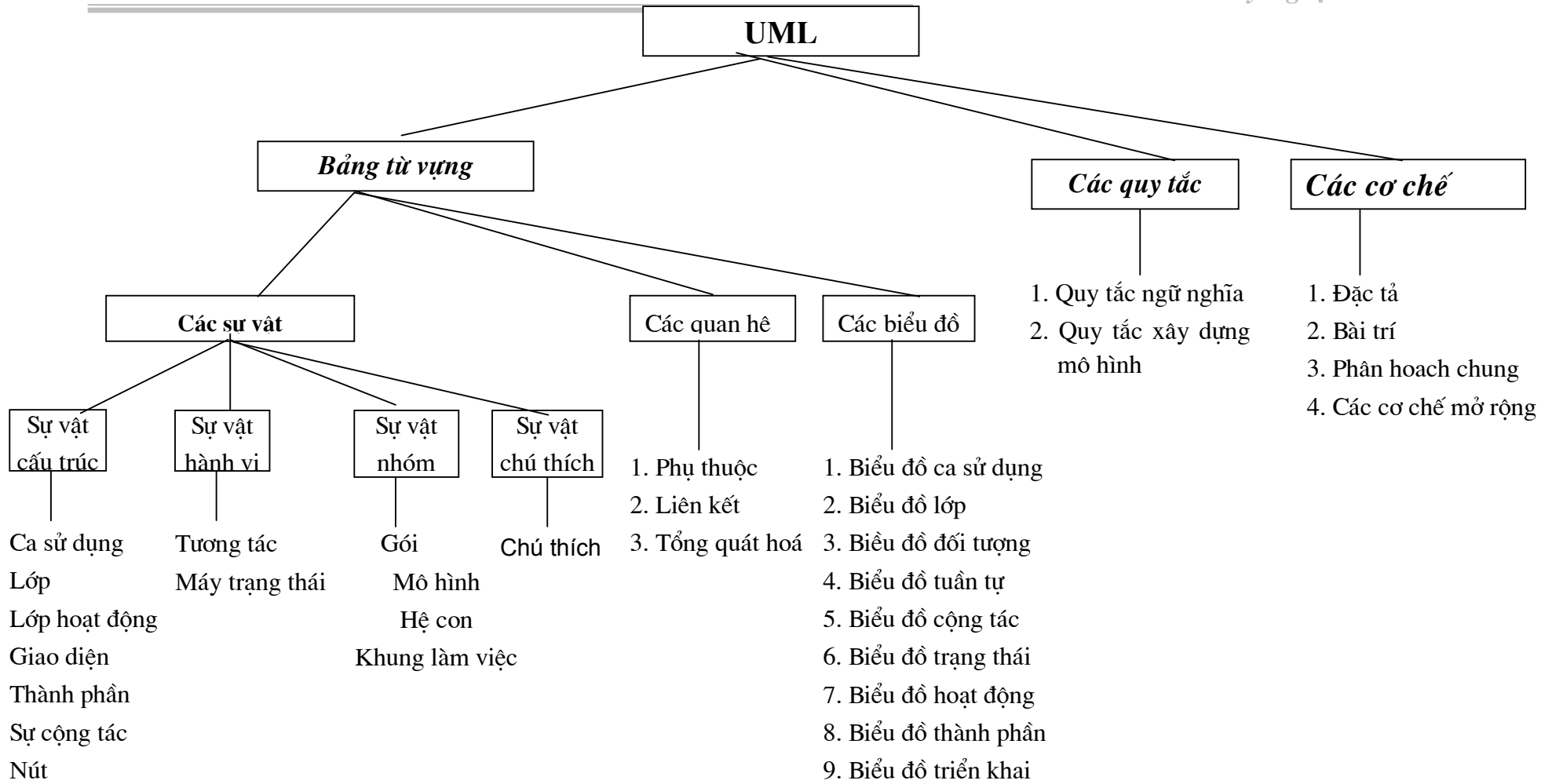
Các cơ chế mở rộng: UML là một ngôn ngữ mở, làm cho nó có khả năng mở rộng theo được cách quản lý. Các cơ chế có thể mở rộng là:

– *Sự rập khuôn (stereotype):* mở rộng từ vựng của UML, cho phép tạo các loại khối xây dựng mới được thừa kế từ các khối có sẵn. UML biểu diễn một đối tượng có sử dụng khuôn mẫu bằng cách đưa tên mô tả vào trong cặp ngoặc '<< >>'.
– *Giá trị gán nhãn (tagged value):* mở rộng các thuộc tính của một khối xây dựng trong UML, cho phép tạo thông tin mới trong đặc tả của phần tử.

– *Ràng buộc (constrain)*: mở rộng ngữ nghĩa của khối xây dựng trong UML, cho phép thêm các quy tắc hay sửa các quy tắc đã có.

Tóm tắt

Ngôn ngữ mô hình hóa thống nhất UML là một ngôn ngữ hỗ trợ phát triển phần mềm hiện đại, tiên tiến, đã được áp dụng nhiều trên thế giới. Nó gắn liền với ngôn ngữ lập trình hướng đối tượng, một ngôn ngữ đang thay thế dần các ngôn ngữ truyền thống khác bởi tính chặt chẽ và hiệu quả. UML giúp người phát triển xây dựng hệ thống từ lúc bắt đầu tìm hiểu cho tới lúc bàn giao sản phẩm phần mềm cho khách hàng, từ việc phân tích hệ thống, xây dựng tài liệu cho tới việc sinh mã...Hiện nay đã có những phần mềm hỗ trợ cho ngôn ngữ UML như Rational Rose dùng trên hệ điều hành Windows...



Bảng thành phần của UML

Phụ lục

THUẬT NGỮ SỬ DỤNG

đơn kéo xuống	pull-down menu	từ điển	dictionary
đa lập trình	multi programming	tệp	file
đặc tả	specification	số modul ra	fan - out
đặc trưng	characteristic	số modul vào	fan - in
đặc tính	characteristic	trạng thái	state
độ đo	metric	môi trường	environment
độ phân giải	resolution	mô phỏng	simulation
độ sâu	depth	phân rã	decomposition
đường dẫn	path	ước lượng thực nghiệm	empirical estimation
điểm chức năng	functional point	số dòng mã lệnh	LOC (line of code)
điều khiển, kiểm soát	Control	vạch ranh giới	base line
điều kiện	condition	biến ước lượng	estimation variable
đỉnh, nút	node	giá trị điều chỉnh độ phức tạp	complexity adjustment value
định danh	identify	giá trị hiệu năng được điều chỉnh	adjusted productivity value
ảnh	image	đặc tả hệ thống	system specification
bảo trì	maintenance	mô hình ước lượng	estimate model
bộ đệm	buffer	mô hình tài nguyên	resource model
bộ xử lý	processor	mô hình đơn biến tĩnh	static single-variable model
biên dịch	compilation	mô hình đa biến tĩnh	static multi-variable model
bước	step	mô hình đơn biến động	dynamic single-variable model
biểu đồ	diagram	mô hình đa biến động	dynamic multi-variable model

công cụ	tool	nhân tố điều chỉnh công sức	effort adjustment factor
công nghệ	technology	nhất quán	consistence
cơ sở dữ liệu	database	mạng nhiệm vụ	task network
cơ sở tri thức	knowledge database	phương pháp đường găng	critical path method
có cấu trúc	structured	thời gian biên	boundary time
cổ kết	cohension	công cụ tái công nghệ ngược	reverse reengineering tool
cột mốc	milestone	công nghệ hệ thống máy tính	computer system engineering
cấp bậc	hierachy	người phân tích hệ thống	system analysisist
cấu hình	configuration	đặc tả yêu cầu phần mềm	software requirement specification
cấu trúc	structure	kế hoạch dự án phần mềm	software project plan
chất lượng	quality	kế hoạch phát triển	development plan
chi phí	cost	phân tích yêu cầu phần cứng	hardware requirement analysis
chiều	dimension	đặc tả thiết kế	design specification
chiều rộng	width	bản kế hoạch và thủ tục kiểm thử	test plan and procedure
chỉ số	index	tương tác người máy	home-machine interaction
cú pháp	syntax	kỹ nghệ con người	human engineering
dữ liệu	data	kỹ nghệ cơ sở dữ liệu	data design
danh sách	list	thiết kế dữ liệu	data design
tái dụng	reusable	tài liệu quan niệm về hệ thống	system concept document
dự án	project	nghiên cứu khả thi	feasibility study
gián tiếp	indirect	khả thi kinh tế	economic feasibility
giải pháp	solution	khả thi kỹ thuật	technical feasibility
giai đoạn	stage	cấu trúc chương	program structure

		trình	
giao diện	interface	cấp bậc điều khiển	control hierachy
hàm, chức năng	function	kiến trúc phần mềm	software architecture
hành vi	behavior	tiêu bản	template
hướng sự vật	object-orient	xét duyệt đặc tả hệ thống	system specification review
hiệu năng	perfomance	tiêu chuẩn hiệu năng	performance criteria
hợp lệ	valid	ràng buộc	constrait
hệ chuyên gia	expert system	người điều khiển	facilitator
hệ thống	system	sự kiện	event
hoàn cảnh	context	miền thông tin	information domain
ước lượng, sự ước lượng	estimate, estimation	luồng thông tin	information flow
kho	store	kho dữ liệu	data store
khoản mục	item	nội dung thông tin	information content
khuôn cảnh	paradigm	cấu trúc thông tin	information structure
kiểm thử	test	cấu trúc dữ liệu	data structure
kết cấu	construct	phân hoạch	partition
kịch bản	scenario	hình thái, hình dạng	form
kỹ nghệ	enginneering	kịch bản	scenario
kỹ nghệ phần mềm	software engineering	đặc tả hình thức	formal specification
kỹ nghệ phần mềm có máy tính hỗ trợ	CASE	môi trường làm bản mẫu	prototype enviroment
kỹ sư phần mềm	software engineer	tác vụ	operation
kỹ thuật thể hệ 4	4GT	biểu đồ dòng dữ liệu	data flow diagram
làm bản mẫu	prototyping	đồ thị luồng dữ liệu	data flow graph
lặp	repeat	biến đổi	transform
lõi	core	cân bằng	balance
lập kế hoạch	planning	luồng sự kiện	event flow

lập lịch	schedule	tiến trình điều khiển	control process
lập trình	programming, coding	biểu đồ luồng điều khiển	control flow diagram
lệnh	instruction	đặc tả điều khiển	control
luồng	flow	điều kiện dữ liệu	data condition
máy tính	computer	biểu đồ chuyển trạng thái	state transition diagram
mô hình thác nước	waterfall model	mô hình hành vi	behaviour modeling
mảng	array	kế thừa	inheritance
mã hoá	coding	thao tác	operation
mạng	network	dịch vụ	service
mạng nơ-ron	neuron network	phương pháp	methods
móc nối, liên kết	link	xác định	identify
nâng cao	enhancement	đơn vị tổ chức	organizational unit
ngôn ngữ	language	thông báo	message
ngôn ngữ thủ tục	procedural language	hiển thị	display
ngữ nghĩa	semantic	cập nhật	update
người lập trình	programmer	phân lớp	classification
nhiều người dùng	multi-user	mô hình hoá dữ liệu	data modeling
phân tích	analysis	tên gọi, định danh	identifier
phân tích hệ thống	system analysis	qui tắc chuẩn hóa	normalization rule
phân tích rủi ro	risk analysis	mô hình quan hệ	relational model
phân tích yêu cầu	requirement analysis	biểu đồ thực thể	entity diagram
phương pháp	method	khai báo	declaration
phần cứng	hardware	môi trường đặc tả	specification environment
phần mềm	software	phân tích miền thông tin	information domain analysis
pha	phase	kiểm thử đơn vị	unit testing
phần mềm hệ thống	system software	kiểm thử tích hợp	intergrate testing
phần mềm nhúng	embedded software	tích hợp	intergrate

phi thủ tục	non-procedural	lưu đồ	flowchart
sơ đồ	shema	làm mịn	refinement
sự sửa đổi	modification	mô đun	module
song song	pararell	cấp bậc điều khiển	control hierachy
sự vật, đối tượng	object	cấu trúc chương trình	program structure
tăng lên	incremental	danh sách móc nối	linked list
khả năng tái sử dụng	reusability	biểu đồ thực thể quan hệ	entity-relationship diagram
tương tác	interactive	biên	boundary
tương tranh	concurence	chương trình gốc	source code
tài liệu	document	ngôn ngữ phi thủ tục	nonprocedural language
tổ hợp	combination	chương trình đích	object code
tầng, lớp	layer	kiểu dữ liệu	data type
thông báo, mẫu tin	message	đệ quy	recursion
thông dịch	interpretation	lớp	class
thành phần	component	lược đồ cấu trúc	structure chart
tham khảo	reference	lệnh	instruction
thời gian thực	real-time	biểu đồ sự vật	object diagram
trực quan	visibility	biểu đồ lớp	class diagram
thiết kế	design	biểu đồ modul	module diagram
thiết kế chi tiết	detail design	biểu đồ xử lý	process diagram
thiết kế sơ bộ	preliminary design	lược đồ	chart
thiết kế cấu trúc	architecture design	đa nhiệm	multitasking
họp xét duyệt thiết kế	design walkthrough	mô hình thiết kế	design modul
Họp xét duyệt kỹ thuật chính thức	formal technical review	mô hình người dùng	user model
thực thể	entity	phân rã chức năng	function decomposition
thượng cấp	superordinate	khoá	lock
thuộc cấp	surbordinate	hộp thư	mailbox

thuộc tính	attribute	sơ đồ hoạt động	activity chart
thích nghi	adaptation	sơ đồ trạng thái	state chart
quá trình, tiến trình	process	mốc	datum

TÀI LIỆU THAM KHẢO

1. *Kỹ nghệ phần mềm – Cách tiếp cận của người thực hành*, Roger S.Pressman, Tập 1, Người dịch: Ngô Trung Việt, NXB Giáo dục, Năm 1997
2. *Kỹ nghệ phần mềm – Cách tiếp cận của người thực hành*, Roger S.Pressman, Tập 2, Người dịch: Ngô Trung Việt, NXB Giáo dục, Năm 1997
3. *Kỹ nghệ phần mềm – Cách tiếp cận của người thực hành*, Roger S.Pressman, Tập 3, Người dịch: Ngô Trung Việt, NXB Giáo dục, Năm 1997
4. *Giáo trình Kỹ nghệ phần mềm*, Nguyễn Văn Vy, Đại học Quốc gia Hà Nội, Năm 2001
5. *Giáo trình Công nghệ phần mềm*, Đại học Khoa học Tự nhiên, Đại học Quốc gia Hà Nội, Năm 2000.
6. *Software Engineering – A Practitioner's Approach*, Roger S.Pressman, Third Edition, McGraw-Hill.Inc, 1992.
7. *The Unified Software Development Process*, Ivar Jacobson, Grandy Booch, James Rumbaugh.