**Báo Cáo Đề Tài**

**Chương 9: Lập Kế Hoạch Và Ước Tính**

|  |
| --- |
| Nhóm 13 |
| Vũ Đức Khôi – B19DCCN363 |
| Đinh Như Cương – B19DCCN075 |
| Ngô Văn Quyết – B19DCCN543 |

|  |
| --- |
|  |

|  |
| --- |
| Giảng Viên: Đặng Ngọc Hùng |

**Chương 9: Lập kế hoạch và ước tính**

**Mục Tiêu học tập**:

Sau khi nghiên cứu chương này chúng ta có thể:

* Giải thích tầm quan trọng của việc lập kế hoạch.
* Ước tính quy mô và chi phí xây dựng một sản phẩm phần mềm.
* Đánh giá cao tầm quan trọng của việc cập nhật và theo dõi các ước tính.
* Lập kế hoạch quản lý dự án phù hợp với tiêu chuẩn IEEE.

Những thách thức khi xây dựng một sản phẩm phần mềm không có giải pháp. Để kết hợp một sản phẩm phần mềm lớn với nhau cần thời gian và tài nguyên. Và, giống như bất kỳ công trình xây dựng lớn dự án nào khác, lập kế hoạch cẩn thận khi bắt đầu dự án có lẽ là yếu tố quan trọng nhất giúp phân biệt thành công và thất bại. Lập kế hoạch, giống như kiểm thử, phải tiếp tục trong suốt quá trình phát triển và bảo trì phần mềm. Bất chấp nhu cầu lập kế hoạch liên tục, những các hoạt động đạt đến đỉnh điểm sau khi các thông số kỹ thuật đã được vạch ra nhưng trước khi các hoạt động thiết kế bắt đầu. Tại thời điểm này trong quá trình, khoảng thời gian có ý nghĩa và ước tính chi phí là được tính toán và lập kế hoạch chi tiết để hoàn thành dự án.

Trong chương này sẽ chia thành 2 loại kế hoạch. Đó là kế hoạch tiến hành trong suốt dự án và kế hoạch kịch liệt (the intense planing, phải được thực hiện khi các thông số kỹ thuật đã được hoàn thành)

**9.1 Lập kế hoạch và quy trình phần mềm**

Thực sự, chúng tôi muốn lập kế hoạch cho toàn bộ dự án phần mềm ngay từ đầu của quá trình, và sau đó thực hiện theo kế hoạch đó cho đến khi phần mềm đích cuối cùng đã được chuyển giao cho khách hàng. Tuy nhiên, điều này là không thể, vì chúng tôi thiếu thông tin trong quy trình làm việc ban đầu để có thể vạch ra một kế hoạch có ý nghĩa cho dự án hoàn chỉnh. Ví dụ trong luồng công việc yêu cầu (requirements workflow), bất kỳ loại lập lịch kế hoạch nào cũng đều vô ích khi không đúng với những kế hoạch đặt ra của luồng công việc yêu cầu (requirements workflow).

Có sự khác biệt rất lớn giữa thông tin do nhà phát triển đưa ra khi kết thúc luồng công việc yêu cầu (requirements workflow) và vào cuối luồng công việc phân tích (analysis workflow), tương tự như sự khác biệt giữa bản phác thảo thô và bản thiết kế chi tiết. Tại thời điểm cuối cùng của luồng phần tích (requirements workflow), nhà phát triển (developers) sẽ nắm rõ được những thông tin về cái mà khách hàng cần. Ngược lại, tại cuối của luồng phân tích (analysis workflow), tại thởi điểm này, khách hàng sẽ xác nhận một tài liệu nêu chính xác những gì được xây dựng (có thể thay đổi so với những công việc đề ra ở luồng yêu cầu ( requirements workflow)), từ đó các nhà phát triển sẽ đánh gía chi tiết về hầu hết (nhưng không phải là tất cả) các khía cạnh của sản phẩm mục tiêu. **Đây là điểm sớm nhất trong quá trình làm sản phẩm để xác định được chính xác khoảng thời gian và ước tính chi phí.**

Tuy nhiên, trong một số tình huống, một tổ chức có thể được yêu cầu đưa ra thời hạn và ước tính chi phí trước khi các trích dẫn cụ thể có thể được lập. Trong trường hợp xấu nhất, khách hàng có thể nhấn mạnh vào một giá thầu trên cơ sở một hoặc hai giờ thảo luận sơ bộ.

|  |
| --- |
|  |
|  |

Hình 9.1 cho thấy làm thế nào vấn đề này có thể được. Dựa trên một mô hình trong [Boehm và cộng sự, 2000], nó mô tả phạm vi ước tính chi phí tương đối cho các quy trình công việc khác nhau của vòng đời. Ví dụ, giả sử rằng, khi một sản phẩm vượt qua kiểm tra chấp nhận ở cuối quy trình triển khai và được giao cho khách hàng, giá của nó là 1 triệu đô la. Nếu một ước tính chi phí có được thực hiện giữa chừng với quy trình công việc yêu cầu, có khả năng là nó sẽ được ở đâu đó trong phạm vi (0,25 triệu đô la, 4 triệu đô la), như thể hiện trong Hình 9.2. Tương tự, nếu ước tính chi phí đã được thực hiện giữa chừng trong quá trình phân tích, phạm vi khả năng ước tính sẽ giảm xuống (0,5 triệu đô la, 2 triệu đô la).

Hơn nữa, nếu ước tính chi phí đã được thực hiện vào cuối quy trình phân tích, nghĩa là, vào thời điểm thích hợp, kết quả có lẽ sẽ vẫn nằm trong phạm vi tương đối rộng (0,67 triệu đô la, 1,5 triệu đô la). Tất cả bốn điểm được đánh dấu trên các đường giới hạn trên và dưới trong Hình 9.2, có thang logarit trên trục tung. Mô hình này được gọi là **Cone Of Uncertainty** (hình nón của sự không chắc chắn).

Rõ ràng từ Hình 9.1 và 9.2 rằng ước tính chi phí không phải là một khoa học chính xác; lý do cho điều này được đưa ra trong Phần 9.2. Thời hạn hoặc ước tính chi phí quá sớm, tức là ước tính được thực hiện trước khi các báo cáo cụ thể được ký kết do khách hàng bật ra, có thể kém chính xác hơn đáng kể so với ước tính được đưa ra khi dữ liệu đầy đủ đã được tích lũy.

Bây giờ chúng ta kiểm tra các kỹ thuật để ước tính thời lượng và chi phí.

**9.2 Thời lượng và chi phí ước tính**

Ngân sách là một trong những phần phần không thể thiếu của bất kỳ kế hoạch quản lý dự án phần mềm nào. Trước khi thiết kế bắt đầu, khách hàng cần biết họ sẽ phải trả bao nhiêu cho sản phẩm.

Nếu nhóm phát triển đánh giá thấp chi phí thực tế, tổ chức phát triển có thể mất tiền vào dự án. Mặt khác, nếu nhóm phát triển đánh giá quá cao, thì khách hàng có thể quyết định rằng, trên cơ sở phân tích chi phí - lợi ích hoặc lợi tức đầu tư, không có ích gì khi sản phẩm được xây dựng. Ngoài ra, khách hàng có thể giao công việc cho một tổ chức phát triển khác có ước tính hợp lý hơn. Dù bằng cách nào, nó là rõ ràng rằng ước tính chi phí chính xác là rất quan trọng.

Trên thực tế, có **hai loại** **chi phí** liên quan đến việc phát triển phần mềm. Điều đầu tiên là **chi phí nội bộ, chi phí cho các viec phat trien**; thứ hai là chi **phí bên ngoài, giá mà khách hàng sẽ trả tiền**.

*Chi phí nội bộ:* bao gồm tiền lương của các nhóm phát triển, người quản lý và nhân viên hỗ trợ tham gia vào dự án; chi phí của phần cứng và phần mềm để phát triển sản phẩm; và chi phí chung như tiền thuê nhà, tiền điện nước và tiền lương của quản lí cấp cao.

Mặc dù giá thường dựa trên chi phí cộng với tỷ suất lợi nhuận, trong một số trường hợp, các yếu tố kinh tế và tâm lý là quan trọng. Các nhà phát triển tính toán ước lượng giá trị sau khi hoàn thành sản phẩm sau đó ra giá cho các khách hàng. Khách hàng cũng có những tâm lý khác nhau khi nhìn nhận giá thành được đưa ra. Khách hàng có thể huỷ hợp đồng mặc dù giá thành khá thấp so với những hợp đồng khác hoặc tương tự vì lý do họ nghĩ chất lượng sản phẩm cũng có thể thấp hơn so với các sản phẩm tương tự. Do đó, một nhóm phát triển có thể cố gắng đưa ra một giá thầu sẽ thấp hơn một chút, nhưng không đáng kể.

Một phần quan trọng khác của bất kỳ kế hoạch nào là ước tính thời gian của dự án. Khách hàng chắc chắn muốn biết khi nào sản phẩm hoàn thành sẽ được giao. Nếu sự phát triển tổ chức không thể giữ đúng lịch trình của mình, thì tổ chức mất uy tín, tại các điều khoản hình phạt tồi tệ nhất được trích dẫn. Nếu nhà phát triển đưa ra khoảng thời gian cao hơn nhiều so với mặt bằng thì cũng khó có thể tiếp cận được khách hàng.

**Ước tính chi phí chính xác và thời gian ước tính** quả thật là **không dễ dàng**. Có nhiều **yếu tố ảnh hướng** đến một trong hai chi phí và thời lượng. Một khó khăn lớn là **yếu tố con người**. Hơn 40 năm trước, Sackman và đồng nghiệp đã quan sát thấy sự khác biệt lên tới 28-1 giữa các cặp lập trình viên. Kết quả này cũng dễ dàng chấp nhận được bằng cách nói rằng các lập trình viên có kinh nghiệm luôn làm tốt hơn những người mới bắt đầu, nhưng Sackman và các đồng nghiệp của ông đã so sánh các cặp lập trình viên phù hợp. Ví dụ, họ quan sát thấy hai lập trình viên với 10 năm kinh nghiệm về các loại dự án tương tự và đo lường thời gian chúng thực hiện các tác vụ như mã hóa và gỡ lỗi. Sau đó, họ quan sát, chẳng hạn, hai người mới bắt đầu đã tham gia nghề nghiệp trong cùng một khoảng thời gian ngắn và có trình độ học vấn tương tự. So sánh hiệu suất kém nhất và tốt nhất, họ quan sát thấy sự khác biệt từ 6 - 1 về kích thước sản phẩm, 8 - 1 trong thời gian thực thi sản phẩm, 9 - 1 trong thời gian phát triển, 18 - 1 trong thời gian mã hóa và 28 - 1 trong thời gian gỡ lỗi. Một quan sát đặc biệt đáng báo động là màn trình diễn tốt nhất và tồi tệ nhất trên một sản phẩm của hai lập trình viên, mỗi người đã có 11 năm kinh nghiệm. Thậm chí khi các trường hợp tốt nhất và xấu nhất được loại bỏ khỏi mẫu của Sackman và cộng sự, sự khác biệt quan sát được vẫn theo thứ tự từ 5 – 1

🡺 **Trên cơ sở các kết quả này, rõ ràng, chúng tôi không thể hy vọng để ước tính chi phí hoặc thời lượng phần mềm với bất kỳ mức độ chính xác nào (trừ khi chúng tôi đã nêu chi tiết thông tin liên quan đến tất cả các kỹ năng của tất cả các nhân viên, điều này sẽ là bất thường nhất). Trong một dự án lớn, việc các đội lập trình viên có kinh nghiệm rất tốt (hoặc rất tệ) sẽ gây ra sự sai lệch rõ rệt so với lịch trình và ảnh hưởng đáng kể đến ngân sách.**

**Một trong các yếu tố khác ảnh** hưởng đến tiến trình và kinh phí đó là những sự việc bất ngờ xảy ra. Ví dụ như có một số nhân việc nghỉ việc tạm thời hoặc nghỉ hẳn , điều này sẽ làm cho công việc bị trì trệ. Mất thời gian và kinh phí để tìm nhân viên có chức năng tương tự…

Tuy nhiên chúng ta cũng có những cơ sở để ước lượng. Cơ bản của việc ước lượng chi phí đó là kích thước của một sản phẩm là như thế nào.

**9.2.1 Các thước đo về kích thước của một sản phẩm**

**Một thước đo chung về kích thước của một sản phẩm là số lượng dòng codes**. 2 đơn vị chung được sử dụng là: **Line Of Code (LOC)**, và **Thousand Delivered Sourse Instructions (KDSI)** (Những nguồn hướng dẫn được phân phối). Có nhiều vấn để liên quan đến việc sử dụng dòng mã.

* Tạo mã nguồn chỉ là một phần nhỏ trong tổng số nỗ lực phát triển phần mềm.Thời gian cần thiết cho các luồng yêu cầu, phân tích, thiết kế, triển khai và kiểm thử.
* Việc triển khai cùng một sản phẩm bằng 2 ngôn ngữ khác nhau dẫn đến các phiên có số dòng mã khác nhau. Ngoài ra với các ngôn ngữ phi thủ tục hay Lisp thì số lượng dòng mã không được định nghĩa.
* Thường thì không rõ ràng chính xác về cách đếm các dòng mã. Chỉ nên thực thi các dòng mã được đếm hoặc định nghĩa dữ liệu là tốt? Và những bình luận có nên được tính không?
* Thường thì không rõ ràng chính xác về cách đếm các dòng mã. Chỉ nên thực thi các dòng mã được đếm hoặc định nghĩa dữ liệu là tốt? Và những comment có nên được tính không? Nếu không, có một vấn đề xảy ra là các lập trình viên sẽ đưa vào tối thiểu các comment hay thậm trí là không để tránh mất thời gian nhưng nếu có tính thì các lập trình viên sẽ lạm dụng việc comment để tăng năng suất.
* Không phải tất cả các mã được triển khai đều được chuyển đến máy khách. Nó không phải là hiếm đối với một nửa, mã bao gồm các công cụ cần thiết để hỗ trợ nỗ lực phát triển.
* Giả sử rằng một nhà phát triển phần mềm sử dụng trình tạo mã, chẳng hạn như trình tạo báo cáo, trình tạo màn hình hoặc trình tạo giao diện người dùng đồ họa (GUI). Sau một vài phút của hoạt động thiết kế từ phía nhà phát triển, công cụ này có thể tạo ra hàng nghìn dòng mã.
* Số lượng dòng mã trong sản phẩm cuối cùng chỉ có thể được xác định khi sản phẩm hoàn thành. Do đó, ước tính chi phí dựa trên các dòng mã là nguy hiểm gấp đôi. Để bắt đầu quá trình ước tính, số dòng mã trong thành phẩm phải được ước tính. Sau đó, ước tính này được sử dụng để ước tính chi phí sản phẩm. Không chỉ có sự không chắc chắn trong mọi kỹ thuật chi phí, mà nếu đầu vào cho bản thân công cụ ước tính chi phí không chắc chắn cũng không chắc chắn (nghĩa là số dòng mã trong sản phẩm chưa được chế tạo), thì độ tin cậy của ước tính chi phí kết quả là không chắc là rất cao.

**Bởi vì số lượng dòng mã không đáng tin cậy**, các chỉ số khác phải được xem xét. Một cách tiếp cận thay thế để ước tính kích thước của sản phẩm là sử dụng các chỉ số dựa trên các đại lượng đo lường có thể được xác định sớm trong quá trình phần mềm. Ví dụ, van der Poel và Schach [1983] đã đưa ra **số liệu FFP** để **ước tính chi phí của các sản phẩm** xử lý dữ liệu quy mô trung bình. **3 yếu tố cấu thành lên FFP** là *File* (các tệp), *Flow* (Luồng), và *Process* (Quy Trình). File là một tập các dữ liệu liên quan đến sản phẩm. Flow (luồng) là giao diện dữ liệu giữa sản phẩm và môi trường, như là screen hay 1 report. Process là một thao tác logic hoặc số học được xác định về mặt chức năng dữ liệu; các ví dụ bao gồm sắp xếp, xác thực hoặc cập nhật. **Với số lượng tệp Fi, các luồng Fl, và xử lý Pr trong 1 sản phẩm, kích thước S và giá thành C được cho bởi công thức:**

|  |
| --- |
|  |

trong đó d là hằng số thay đổi giữa các tổ chức. **Hằng số d là thước đo efficiency (hiệu quả, năng suất) của quá trình phát triển phần mềm trong đó cơ quan**. Kích thước của một sản phẩm đơn giản là tổng số tệp, số lượng tệp và quy trình, mỗi số lượng có thể được xác định sau khi thiết kế kiến trúc hoàn thành. Các khi đó chi phí tỷ lệ với quy mô, **hằng số tỷ lệ d được xác định bởi bình phương nhỏ nhất phù hợp với dữ liệu chi phí liên quan đến các sản phẩm do tổ chức đó phát triển trước đó.**

Không giống như các chỉ số dựa trên số dòng mã, chi phí có thể được ước tính trước mã hóa bắt đầu. Tính hợp lệ và độ tin cậy của chỉ số **FFP** đã được chứng minh bằng cách sử dụng mẫu bao gồm một loạt các ứng dụng xử lý **dữ liệu quy mô trung bình**. Đáng tiếc, chỉ số này không bao giờ được mở rộng để bao gồm cơ sở dữ liệu, một thành phần thiết yếu của nhiều sản phẩm xử lý cơ sở dữ liệu.

Một số liệu tương tự, nhưng được phát triển độc lập, cho kích thước của sản phẩm đã được phát triển của Albrecht [1979] dựa trên các **điểm chức năng (Function Ponits**); Số liệu của Albrecht dựa trên số lượng mục đầu vào (input items) *Inp*, các mục đầu ra *Out*, yêu cẩu (Inquiries) *Inq*, các file chính *Maf,* và các giao diện *Inf.* Trong đó dạng đơn giản nhất của FP được cho bơi phương trình:

|  |
| --- |
|  |

Vì đây là thước đo kích thước của sản phẩm nên nó có thể được sử dụng để ước tính chi phí và ước tính năng suất.

Công thức (9.3) là phép tính đơn giản của phép tính ba bước. Ba bước cụ thể như sau:

Đầu tiên, điểm chức năng chưa điều chỉnh được tính.

1. Mỗi thành phần của sản phẩm –Inp, Out, Inq, Maf, Inf- phải được phân loại là đơn giản, trung bình hoặc là phức tạp (H 9.3)

|  |
| --- |
|  |
|  |

1. Mỗi thành phần được gán một số **điểm chức năng** tùy **thuộc vào cấp độ của nó**. Ví dụ như Inf trong bảng 9.3 ở mức Simple có điểm chức năng là 3, mức trung bình (Average) là 4…
2. Sau đó, các điểm chức năng được gán cho mỗi thành phần được tổng hợp lại, tạo ra các điểm chức năng chưa được điều chỉnh **(UFP).**

Bước 2, **hệ số phức tạp kỹ thuật (TCF)** được tính toán. Đây là thước đo ảnh hưởng của 14 yếu tố kỹ thuật được liệt kê trong hình 9.4. Mỗi yếu tố có thăng điểm từ 0 – 5 (0 là không hoặc không có, 5 là mức ảnh hưởng mạnh nhất). Tổng của 14 giá trị này cho ra **tổng mức độ ảnh hưởng (DI),**  lúc này TCF được tính bởi:

|  |
| --- |
|  |

Vì DI có giá trị nằm trong khoảng từ 0 -70 nên TCF có giá trị từ 0.65 – 1.35.

|  |
| --- |
|  |

Cuối cùng, FP được tính bởi công thức:

|  |
| --- |
|  |

Các thử nghiệm để tỷ lệ năng suất phần mềm đã cho thấy sự phù hợp hơn khi sử dụng các điểm chức năng so với sử dụng KDSI Ví dụ, Jones [1987] đã tuyên bố rằng ông đã quan sát thấy các lỗi vượt quá 800 phần trăm khi đếm KDSI, nhưng chỉ [nhấn mạnh thêm] 200 phần trăm trong việc đếm các điểm chức năng.

Để cho thấy sự vượt trội của các điểm chức năng () so với các dòng mã, Jones [1987] trích dẫn ví dụ được chỉ ra trong Hình 9.5

|  |
| --- |
|  |

**9.2.2 Các kỹ thuật ước tính chi phí**

Điều quan trọng là các nhà phát triển phần mềm phải làm hết sức có thể để có được ước tính chính xác về cả thời lượng dự án và chi phí dự án, đồng thời tính đến càng nhiều yếu tố có thể ảnh hưởng đến ước tính của họ càng tốt. Chúng bao gồm trình độ kỹ năng của nhân sự, mức độ phức tạp của dự án, quy mô của dự án (chi phí tăng theo quy mô nhưng nhiều hơn so với tuyến tính), sự quen thuộc của nhóm phát triển với khu vực ứng dụng, phần cứng mà sản phẩm sẽ trở thành chạy và tính khả dụng của các công cụ CASE. Một yếu tố khác là hiệu ứng thời hạn thời hạn càng ngắn, nỗ lực càng lớn và do đó, chi phí càng lớn. bài viết đưa ra các kỹ thuật ước tính chi phí.

**1. Phán đoán của chuyên gia bằng phép tương tự.**

Trong phần này, một số chuyên gia được tham khảo ý kiến. Một chuyên gia đưa ra ước tính bằng cách so sánh sản phẩm mục tiêu với các sản phẩm đã hoàn thành mà chuyên gia đã tham gia tích cực. Ví dụ: một chuyên gia có thể so sánh sản phẩm mục tiêu với một sản phẩm tương tự được phát triển cách đây 2 năm. Vì tổ chức đã quen với loại sản phẩm sẽ được phát triển, chuyên gia giảm thời gian và nỗ lực phát triển đi 15 phần trăm. Tuy nhiên, giao diện người dùng đồ họa hơi phức tạp; điều này làm tăng thời gian và nỗ lực lên 25 phần trăm. Cuối cùng, sản phẩm mục tiêu phải được phát triển bằng một ngôn ngữ mà hầu hết các thành viên trong nhóm không quen thuộc, do đó tăng thời gian lên 15 phần trăm và nỗ lực lên 20 phần trăm. Kết hợp ba số liệu này, chuyên gia quyết định rằng sản phẩm mục tiêu sẽ tốn nhiều thời gian hơn 25% và nỗ lực hơn 30% so với sản phẩm trước đó. Vì sản phẩm trước đó mất 12 tháng để hoàn thành và cần 100 người-tháng, sản phẩm mục tiêu ước tính mất 15 tháng và tiêu thụ 130 người-tháng.

Hai chuyên gia khác trong tổ chức so sánh hai sản phẩm giống nhau. Một người kết luận rằng sản phẩm mục tiêu sẽ mất 13,5 tháng và 140 tháng. Người còn lại đưa ra các số liệu của 16 tháng và 95 người-tháng. Làm thế nào để các dự đoán của ba chuyên gia này có thể được đối chiếu? Một kỹ thuật là kỹ thuật Delphi: Nó cho phép các chuyên gia đi đến thống nhất mà không cần họp nhóm, điều này có thể gây ra tác dụng phụ không mong muốn là một thành viên thuyết phục làm lung lay nhóm. Trong kỹ thuật này, các chuyên gia làm việc độc lập. Mỗi thứ tạo ra một ước tính và cơ sở lý luận cho ước tính đó. Sau đó, những ước tính và hợp lý này được phân phối cho tất cả các chuyên gia, những người hiện đưa ra ước tính thứ hai. Quá trình ước tính và phân phối này tiếp tục cho đến khi các chuyên gia có thể đồng ý trong phạm vi dung sai được chấp nhận.

Hãy nhớ lại rằng chuyên gia phần mềm đầu tiên của chúng tôi đã tuyên bố rằng việc sử dụng một ngôn ngữ không quen thuộc sẽ tăng thời gian lên 15 phần trăm và nỗ lực lên 20 phần trăm. Trừ khi chuyên gia có một số dữ liệu đã được xác thực mà từ đó có thể xác định được ảnh hưởng của từng sự khác biệt (khả năng rất khó xảy ra), các sai sót gây ra bởi những gì có thể được mô tả chỉ là phỏng đoán sẽ dẫn đến ước tính chi phí không chính xác một cách vô vọng. Ngoài ra, trừ khi các chuyên gia may mắn với tổng số thu hồi (hoặc đã lưu giữ hồ sơ chi tiết), những hồi tưởng của họ về các sản phẩm đã hoàn thành có thể không chính xác đến mức làm mất hiệu lực dự đoán của họ. Cuối cùng, các chuyên gia là con người và do đó, có những thành kiến có thể ảnh hưởng đến dự đoán của họ. Đồng thời, kết quả ước tính của một nhóm chuyên gia nên lấy kinh nghiệm tập thể của họ; nếu điều này đủ rộng, thì kết quả cũng có thể chính xác.

**2. Phương pháp tiếp cận từ dưới lên**

Một cách để cố gắng giảm thiểu sai sót do đánh giá tổng thể một sản phẩm là chia sản phẩm thành các thành phần nhỏ hơn. Ước tính về thời lượng và chi phí được thực hiện cho từng thành phần riêng biệt và được kết hợp để cung cấp một con số tổng thể. Cách tiếp cận từ dưới lên này có ưu điểm là ước tính chi phí cho một số thành phần nhỏ hơn thường nhanh hơn và chính xác hơn so với một thành phần lớn. Ngoài ra, quá trình ước tính có thể sẽ chi tiết hơn so với một sản phẩm lớn, nguyên khối. Điểm yếu của cách tiếp cận này là một sản phẩm nhiều hơn tổng các thành phần của nó. Với mô hình hướng đối tượng, sự độc lập của các lớp khác nhau giúp cho cách tiếp cận từ dưới lên. Tuy nhiên, sự tương tác giữa các đối tượng khác nhau trong sản phẩm làm phức tạp quá trình ước tính.

**3. Mô hình ước tính chi phí theo thuật toán**

Trong cách tiếp cận này, một số liệu, chẳng hạn như điểm chức năng hoặc số liệu FFP, được sử dụng làm đầu vào cho mô hình xác định giá thành sản phẩm. Công cụ ước tính tính toán giá trị của số liệu; ước tính thời lượng và chi phí sau đó có thể được tính toán bằng cách sử dụng mô hình. Nhìn bề ngoài, mô hình ước tính chi phí theo thuật toán vượt trội hơn so với ý kiến ​​của chuyên gia, bởi vì một chuyên gia về con người, như đã chỉ ra trước đây, có thể bị sai lệch và có thể bỏ qua một số khía cạnh của cả sản phẩm đã hoàn thành và sản phẩm mục tiêu. Ngược lại, một mô hình ước tính chi phí theo thuật toán là không thiên vị; mọi sản phẩm đều được xử lý theo cùng một cách. Điều nguy hiểm với một mô hình như vậy là các ước tính của nó chỉ tốt như các giả định cơ bản. Ví dụ, cơ bản của mô hình điểm chức năng là giả định rằng mọi khía cạnh của sản phẩm được thể hiện trong năm đại lượng ở phía bên phải của phương trình (9.3) và 14 yếu tố kỹ thuật. Một vấn đề nữa là thường cần một lượng đáng kể phán đoán chủ quan trong việc quyết định những giá trị nào cần gán cho các tham số của mô hình. Ví dụ, thường xuyên nó không rõ liệu hệ số kỹ thuật cụ thể c của mô hình điểm chức năng nên được xếp hạng 3 hay 4.

Nhiều mô hình ước tính chi phí theo thuật toán đã được đề xuất. Một số dựa trên các lý thuyết toán học về cách phần mềm được phát triển. Các mô hình khác dựa trên thống kê; số lượng lớn các dự án được nghiên cứu và xác định các quy tắc thực nghiệm từ dữ liệu. Các mô hình kết hợp kết hợp các phương trình toán học, mô hình thống kê và đánh giá của chuyên gia. Mô hình lai quan trọng nhất là Boehm’s COCOMO

*(COCOMO là một từ viết tắt được hình thành từ hai chữ cái đầu tiên của mỗi từ trong COnstructive COst MOdel)*

COCOMO thực sự là một chuỗi ba mô hình, từ mô hình xác thực vĩ mô xử lý sản phẩm một cách tổng thể đến mô hình xác thực vi mô xử lý sản phẩm một cách chi tiết. Trong phần này, mô tả được đưa ra về COCOMO trung gian, có mức độ phức tạp và chi tiết ở mức trung bình.

Thời gian phát triển máy tính sử dụng COCOMO trung gian được thực hiện trong hai giai đoạn. Đầu tiên, một ước tính sơ bộ về nỗ lực phát triển được cung cấp. Hai thông số phải được ước tính: chiều dài của sản phẩm trong KDSI và chế độ phát triển của sản phẩm, một số đo về mức độ khó khăn nội tại của việc phát triển sản phẩm đó. Có ba chế độ: organic(nhỏ và đơn giản), semidetached (kích thước trung bình) và embedded (phức tạp).

Từ hai tham số này, có thể tính được **nominal effort**. Ví dụ, nếu dự án được đánh giá là về cơ bản là đơn giản (organic), thì **nominal effort**(tính bằng người-tháng) được đưa ra bởi phương trình:

|  |
| --- |
|  |

(Các hằng số 3.2 và 1.05 là các giá trị phù hợp nhất với dữ liệu về các sản phẩm chế độ organic được Boehm sử dụng để phát triển COCOMO trung gian.)

Boehm cung cấp các nguyên tắc để giúp nhà phát triển xác định xem liệu thông số có thực sự nên được xếp hạng danh nghĩa hay xếp hạng thấp hơn hay cao hơn. Ví dụ, hãy xem xét lại hệ số phức tạp của mô-đun. Nếu các hoạt động điều khiển của mô-đun về cơ bản bao gồm một chuỗi các cấu trúc của lập trình có cấu trúc (chẳng hạn như if - then - else, do - while, case), thì độ phức tạp được đánh giá là rất thấp. Nếu các toán tử này được lồng vào nhau, thì xếp hạng sẽ thấp. Việc thêm các bảng quyết định và điều khiển liên mô-đun sẽ tăng xếp hạng lên danh nghĩa. Nếu các toán tử được lồng vào nhau nhiều, với các vị từ ghép và hàng đợi và ngăn xếp, thì xếp hạng sẽ cao. Sự hiện diện của mã hóa quay lại và đệ quy và xử lý ngắt ưu tiên cố định đẩy xếp hạng lên rất cao. Cuối cùng, lập lịch nhiều tài nguyên với các mức độ ưu tiên thay đổi động và kiểm soát mức vi mã đảm bảo rằng xếp hạng sẽ rất cao. Các xếp hạng này áp dụng cho các hoạt động kiểm soát. Một mô-đun cũng phải được đánh giá từ quan điểm của các hoạt động tính toán, hoạt động phụ thuộc vào thiết bị và hoạt động quản lý dữ liệu.

Để xem cách này hoạt động như thế nào, Boehm [1984] đưa ra ví dụ về phần mềm xử lý truyền thông dựa trên bộ vi xử lý cho một mạng chuyển tiền điện tử mới có độ tin cậy cao, với các yêu cầu về hiệu suất, lịch phát triển và giao diện. Sản phẩm này là mô tả của chế độ nhúng và ước tính có độ dài 10.000 hướng dẫn nguồn được phân phối (10 KDSI), vì vậy **norminal effort** được đưa ra bởi:

|  |
| --- |
|  |

(các hằng số 2,8 và 1,20 là các giá trị phù hợp nhất với dữ liệu trên các sản phẩm phức tạp.)

Bởi vì dự án được ước tính có độ dài 10 KDSI, nỗ lực danh nghĩa là:



Xếp hạng của các số nhân này và giá trị của chúng được cho trong Hình 9.7. Sử dụng các giá trị này, tích của các số nhân được tìm thấy là 1,35, do đó, nỗ lực ước tính cho dự án là:



|  |
| --- |
|  |
| Con số này sau đó được sử dụng trong các công thức bổ sung để xác định chi phí đô la, lịch trình phát triển, phân phối giai đoạn và hoạt động, chi phí máy tính, chi phí bảo trì hàng năm và các hạng mục liên quan khác.  COCOMO trung gian đã được xác nhận đối với một mẫu rộng gồm 63 dự án bao gồm nhiều lĩnh vực ứng dụng khác nhau. Kết quả của việc áp dụng COCOMO trung gian cho mẫu này là các giá trị thực tế nằm trong khoảng 20% so với dự đoán giá trị khoảng 68 phần trăm thời gian. Những nỗ lực để cải thiện độ chính xác này không có ý nghĩa gì vì trong hầu hết các tổ chức, dữ liệu đầu vào cho COCOMO trung gian thường chỉ chính xác trong khoảng 20%. Tuy nhiên, độ chính xác thu được bởi các nhà ước tính có kinh nghiệm đã đặt COCOMO trung gian vào vị trí tiên tiến của nghiên cứu ước tính chi phí trong suốt những năm 1980; không có kỹ thuật nào khác chính xác nhất quán.  Vấn đề chính của COCOMO trung gian là đầu vào quan trọng nhất của nó là số dòng mã trong sản phẩm mục tiêu. Nếu ước tính này không chính xác, thì mọi dự đoán của mô hình có thể không chính xác. Do có khả năng các dự đoán của COCOMO trung gian hoặc bất kỳ kỹ thuật ước lượng nào khác có thể không chính xác, ban quản lý phải theo dõi tất cả các dự đoán trong suốt quá trình phát triển phần mềm.  **9.2.4 COCOMO II**  COCOMO được đưa ra vào năm 1981. Vào thời điểm đó, mô hình vòng đời duy nhất được sử dụng là mô hình thác nước. Hầu hết phần mềm được chạy trên máy tính lớn. Về cơ bản, các công nghệ như máy khách-máy chủ và hướng đối tượng chưa được biết đến. Theo đó, COCOMO đã không kết hợp bất kỳ yếu tố nào trong số này. Tuy nhiên, khi các công nghệ mới hơn bắt đầu trở thành thực hành kỹ thuật phần mềm được chấp nhận, COCOMO bắt đầu trở nên kém chính xác hơn.  COCOMO II [Boehm và cộng sự, 2000] là một bản sửa đổi lớn của COCOMO năm 1981. COCOMO II có thể xử lý nhiều loại kỹ thuật kỹ thuật phần mềm hiện đại, bao gồm hướng đối tượng, các mô hình vòng đời khác nhau được mô tả trong Chương 2, tạo mẫu nhanh Phần 11.13), ngôn ngữ thế hệ thứ tư (Phần 15.2), sử dụng lại (Phần 8.1) và phần mềm COTS (Phần 1.11). COCOMO II vừa linh hoạt vừa tinh vi. Thật không may, để đạt được mục tiêu này, COCOMO II phức tạp hơn đáng kể so với COCOMO ban đầu. chỉ tổng quan về sự khác biệt chính giữa COCOMO II và trung gian COCOMO được đưa ra ở đây.  Đầu tiên, COCOMO trung gian bao gồm một mô hình tổng thể dựa trên các dòng mã (KDSI). Mặt khác, COCOMO II bao gồm ba mô hình khác nhau. (Applicationcomposition model) Mô hình thành phần ứng dụng, dựa trên các điểm đối tượng (tương tự như các điểm chức năng), được áp dụng sớm nhất trong quy trình công việc, khi có sẵn kiến ​​thức tối thiểu về sản phẩm sẽ được xây dựng, khi có nhiều kiến ​​thức hơn, mô hình thiết kế ban đầu được sử dụng năng (early design model), mô hình này dựa trên các điểm chức, Cuối cùng, khi các nhà phát triển có thông tin tối đa, mô hình hậu kiến ​​trúc sẽ được sử dụng(postarchitecture model). Mô hình này sử dụng các điểm chức năng hoặc các dòng mã (KDSI).  Đầu ra từ COCOMO trung gian là ước tính chi phí và quy mô; đầu ra từ mỗi mô hình trong số ba mô hình của COCOMO II là một loạt các ước tính về chi phí và kích thước. Theo đó, nếu ước tính có khả năng nhất của nỗ lực là E, thì mô hình thành phần ứng dụng trả về phạm vi (0,50E, 2.0E) và mô hình hậu kiến ​​trúc trả về phạm vi (0,80E, 1,25E). Điều này phản ánh mức độ chính xác ngày càng tăng của tiến trình của các mô hình COCOMO II.  Điểm khác biệt thứ hai nằm ở mô hình nỗ lực cơ bản của COCOMO:   |  | | --- | |  |   trong đó a và b là hằng số. Trong COCOMO trung gian, số mũ b nhận ba giá trị khác nhau, tùy thuộc vào chế độ của sản phẩm được xây dựng là organic (b = 1,05), semidetached (b = 1,12) hay embedded (b = 1,20). Trong COCOMO II, giá trị của b thay đổi trong khoảng từ 1,01 đến 1,26, tùy thuộc vào nhiều tham số của mô hình. Chúng bao gồm sự quen thuộc với các sản phẩm thuộc loại đó, mức độ thành thục của quy trình (Phần 3.13), mức độ giải quyết rủi ro (Phần 2.7) và mức độ hợp tác nhóm (Phần 4.1).  Sự khác biệt thứ ba là giả định liên quan đến việc tái sử dụng. COCOMO trung gian giả định rằng số tiền tiết kiệm được do tái sử dụng tỷ lệ thuận với số lượng tái sử dụng. COCOMO II tính đến rằng những thay đổi nhỏ đối với phần mềm được sử dụng lại sẽ phát sinh chi phí lớn không tương xứng (vì mã đã được hiểu chi tiết cho dù chỉ một thay đổi nhỏ và chi phí thử nghiệm mô-đun modifi ed là tương đối lớn).  Thứ tư, hiện có 17 trình điều khiển chi phí nhân, thay vì 15 trong COCOMO trung gian. Bảy trong số các yếu tố chi phí là yếu tố mới, chẳng hạn như khả năng tái sử dụng bắt buộc đối với các sản phẩm trong tương lai, luân chuyển nhân sự hàng năm và liệu sản phẩm có đang được phát triển tại nhiều địa điểm hay không.  COCOMO II đã được hiệu chỉnh bằng cách sử dụng 83 dự án từ nhiều lĩnh vực khác nhau. Mô hình vẫn còn quá mới để có nhiều kết quả về độ chính xác của nó và đặc biệt là mức độ cải tiến của nó so với người tiền nhiệm của nó, COCOMO ban đầu (1981).  **9.2.5 Thời lượng theo dõi và ước tính chi phí**  Trong khi sản phẩm đang được phát triển, nỗ lực phát triển thực tế phải liên tục được so sánh với các dự đoán. Ví dụ: giả sử rằng số liệu ước tính được các nhà phát triển phần mềm sử dụng dự đoán rằng thời gian (**Duration**) của quy trình phân tích sẽ kéo dài 3 tháng và yêu cầu 7 người/tháng. Tuy nhiên, 4 tháng đã trôi qua và 10 người/tháng đã được sử dụng, nhưng các thông tin cụ thể vẫn chưa hoàn thành. Những sai lệch kiểu này có thể coi là một cảnh báo sớm rằng có điều gì đó không ổn và cần phải thực hiện hành động sửa chữa. Vấn đề có thể là do kích thước của sản phẩm bị đánh giá thấp nghiêm trọng hoặc nhóm phát triển không đủ năng lực như người ta vẫn tưởng. Bất kể lý do là gì, sẽ có thời gian và chi phí vượt quá nghiêm trọng, và ban giám đốc phải có hành động thích hợp để giảm thiểu các ảnh hưởng.  Việc theo dõi cẩn thận các dự đoán phải được thực hiện trong suốt quá trình phát triển, bất kể các kỹ thuật mà các dự đoán đã được thực hiện. Sai lệch có thể là do các chỉ số dự đoán kém, phát triển phần mềm không hiệu quả, kết hợp cả hai hoặc một số lý do khác. Điều quan trọng là phát hiện sớm những sai lệch và có hành động sửa chữa ngay lập tức. Ngoài ra, điều cần thiết là phải liên tục cập nhật các dự đoán dưới ánh sáng của thông tin bổ sung khi nó có sẵn.  **9.3 Các thành phần của kế hoạch quản lý dự án phần mềm**  Kế hoạch quản lý dự án phần mềm có **ba thành phần chính**: *công việc phải thực hiện*, *nguồn lực để thực hiện* nó và *số tiền chi trả* cho tất cả. Trong phần này, ba thành phần của kế hoạch sẽ được thảo luận.  **Phát triển phần mềm cần có nguồn lực (resources). Các nguồn lực chính cần có là những người sẽ phát triển phần mềm, phần cứng chạy phần mềm và phần mềm hỗ trợ như hệ điều hành, trình soạn thảo văn bản và phần mềm điều khiển phiên bản.**  Việc sử dụng các nguồn lực như nhân sự thay đổi theo thời gian. Norden [1958] đã chỉ ra rằng đối với các dự án lớn, **phân bố Rayleigh** là một xấp xỉ tốt của cách mà tiêu thụ tài nguyên (Rc), thay đổi theo thời gian (t), nghĩa là:   |  | | --- | |  |   Tham số k là hằng số thời gian mà mức tiêu thụ ở mức cao nhất và e = 2,71828. Một đường cong Rayleigh điển hình được thể hiện trong Hình 9.8.   |  | | --- | | Mức tiêu thụ tài nguyên bắt đầu nhỏ, tăng nhanh đến đỉnh, sau đó giảm với tốc độ chậm hơn. Putnam [1978] đã nghiên cứu khả năng ứng dụng các kết quả của Norden vào việc phát triển phần mềm và nhận thấy rằng nhân sự và mức tiêu thụ tài nguyên khác được mô hình hóa với độ chính xác ở một mức độ nào đó bằng phân phối Rayleigh. |   Thực tế là nhu cầu nguồn lực phụ thuộc vào thời gian không chỉ áp dụng cho nhân sự mà còn áp dụng cho thời gian sử dụng máy tính, phần mềm hỗ trợ, phần cứng máy tính, cơ sở vật chất và thậm chí cả việc đi lại. Do đó, kế hoạch quản lý dự án phần mềm là một hàm của thời gian.  Công việc phải làm được chia thành hai loại. **Đầu tiên** là công việc tiếp tục trong suốt dự án và không liên quan đến bất kỳ quy trình phát triển phần mềm cụ thể nào. Công việc như vậy được gọi là một chức năng của dự án (**Project Function**) Ví dụ như quản lý dự án và kiểm soát chất lượng. Thứ hai là công việc liên quan đến quy trình làm việc cụ thể trong quá trình phát triển sản phẩm; công việc đó được gọi là một hoạt động (**activity**) hoặc một nhiệm vụ (**task**). Một hoạt động (**activity**) là một đơn vị công việc chính có ngày bắt đầu và ngày kết thúc chính xác, tiêu tốn tài nguyên, chẳng hạn như thời gian sử dụng máy tính và tạo ra các sản phẩm công việc, chẳng hạn như ngân sách, tài liệu thiết kế, lịch trình, mã nguồn hoặc hướng dẫn sử dụng. Ngược lại, một hoạt động bao gồm một tập hợp các nhiệm vụ (**task**), một nhiệm vụ là đơn vị công việc nhỏ nhất chịu trách nhiệm giải trình của cấp quản lý. **Do đó, có ba loại công việc trong kế hoạch quản lý dự án phần mềm**: các chức năng dự án (**Project Function**) được thực hiện trong suốt dự án, các hoạt động (**Activity**) (đơn vị công việc chính) và nhiệm vụ (**Task**) (đơn vị công việc nhỏ).  Trên thực tế, có nhiều thứ hơn đối với một sản phẩm làm việc chứ không chỉ đơn thuần là bản thân sản phẩm đó. Một gói công việc xác định không chỉ sản phẩm công việc mà còn xác định các yêu cầu của nhân viên, thời gian, nguồn lực, tên của cá nhân chịu trách nhiệm và tiêu chí chấp nhận cho sản phẩm công việc. Tiền của khóa học là một thành phần quan trọng của kế hoạch. Một ngân sách chi tiết phải được vạch ra và phân bổ tiền, như một chức năng của thời gian, cho các chức năng và hoạt động của dự án.  Vấn đề làm thế nào để lập một kế hoạch sản xuất phần mềm.  **9.4 Khung kế hoạch quản lý dự án phần mềm**  Có nhiều cách để lập một kế hoạch quản lý dự án. Một trong những tiêu chuẩn tốt nhất là IEEE Standard 1058 [1998]. Các thành phần của kế hoạch được thể hiện trong Hình 9.9.   1. Tiêu chuẩn được đưa ra bởi đại diện của nhiều tổ chức lớn liên quan đến phát triển phần mềm. 2. Kế hoạch quản lý dự án IEEE được thiết kế để sử dụng với tất cả các loại sản phẩm phần mềm. Nó không áp đặt một mô hình vòng đời cụ thể hoặc quy định một phương pháp luận cụ thể. 3. Khung kế hoạch quản lý dự án IEEE hỗ trợ cải tiến quy trình. Ví dụ, nhiều phần của khung quy định các lĩnh vực quy trình chính của CMM (Phần 3.13) chẳng hạn như quản lý và đo lường kết hợp. 4. Khung kế hoạch quản lý dự án IEEE là lý tưởng cho Quy trình hợp nhất. Ví dụ, một phần của kế hoạch được dành cho kiểm soát các yêu cầu và một phần khác cho quản lý rủi ro, cả hai khía cạnh trung tâm của Quy trình hợp nhất.     **9.5 Kế hoạch quản lý dự án phần mềm IEEE (SPMP)**  Bản thân khung kế hoạch quản lý dự án phần mềm (**SPMP**) của IEEE hiện nay đã được mô tả chi tiết.Các thuật ngữ được sử dụng:  1 Overview.1.1 Project summary.  1.1.1 Purpose, scope, and objectives.  Một mô tả ngắn gọn được đưa ra về mục đích và phạm vi của sản phẩm phần mềm sẽ được phân phối, cũng như các mục tiêu của dự án. Nhu cầu kinh doanh được bao gồm trong tiểu mục này.  1.1.2 Assumptions and constraints (Các giả định và ràng buộc)  Bất kỳ giả định nào làm cơ sở cho dự án đều được nêu ở đây, cùng với các ràng buộc, chẳng hạn như ngày giao hàng, ngân sách, nguồn lực và hiện vật sẽ được sử dụng lại.  1.1.3 Project deliverables (**Các sản phẩm của dự án**)  Tất cả các mặt hàng sẽ được giao cho khách hàng được liệt kê ở đây, cùng với ngày giao hàng.  1.1.4 Schedule and budget summary (Lịch trình và tóm tắt ngân sách.)  Lịch trình tổng thể được trình bày ở đây, cùng với ngân sách tổng thể.  1.2 Evolution of the project management plan.( Diễn biến của kế hoạch quản lý dự án.)  Không có kế hoạch nào có thể được đúc bằng bê tông.  Kế hoạch quản lý dự án, giống như bất kỳ kế hoạch nào khác, yêu cầu cập nhật liên tục theo kinh nghiệm và sự thay đổi trong cả tổ chức khách hàng và tổ chức phát triển phần mềm. Trong phần này, các thủ tục và cơ chế chính thức để thay đổi kế hoạch được mô tả, bao gồm cơ chế đặt chính kế hoạch quản lý dự án dưới sự kiểm soát cấu hình.  2 Reference materials.( Tài liệu tham khảo.)  Tất cả các tài liệu được tham chiếu trong kế hoạch quản lý dự án được liệt kê ở đây.  3 Definitions and acronyms (Định nghĩa và từ viết tắt)  Thông tin này đảm bảo rằng kế hoạch quản lý dự án sẽ được mọi người hiểu theo cách giống nhau.  4 Project organization. (Tổ chức dự án.)  4.1 External interfaces (Các giao diện bên ngoài)  Không có dự án nào được xây dựng trong môi trường chân không. Các thành viên của dự án phải tương tác với tổ chức khách hàng và các thành viên khác trong tổ chức của họ. Ngoài ra, các nhà thầu phụ có thể tham gia vào một dự án lớn. Ranh giới hành chính và quản lý giữa dự án và các đơn vị khác phải được xác định.  4.2 Internal structure.( Cấu trúc bên trong)  Trong phần này, cấu trúc của chính tổ chức phát triển được mô tả. Ví dụ, nhiều tổ chức phát triển phần mềm được chia thành hai loại nhóm: nhóm phát triển hoạt động trên một dự án duy nhất và nhóm hỗ trợ cung cấp các chức năng hỗ trợ, chẳng hạn như quản lý cấu hình và đảm bảo chất lượng, trên cơ sở toàn tổ chức. Ranh giới hành chính và quản lý giữa nhóm dự án và các nhóm hỗ trợ cũng phải được xác định rõ ràng.  4.3 Roles and responsibilities (Vai trò và trách nhiệm)  Đối với từng chức năng của dự án, chẳng hạn như đảm bảo chất lượng, và đối với từng hoạt động, chẳng hạn như thử nghiệm sản phẩm, cá nhân chịu trách nhiệm phải được xác định.  5 Managerial process plans (Kế hoạch quy trình quản lý)  5.1 Start-up plan (Kế hoạch khởi nghiệp)  5.1.1 Estimation plan (Kế hoạch ước tính)  Các kỹ thuật được sử dụng để ước tính thời gian và chi phí của dự án được liệt kê ở đây, cũng như cách các ước tính này được theo dõi và, nếu cần, được sửa đổi trong khi dự án đang thực hiện.  5.1.2 Staffing plan (Kế hoạch nhân sự)  Số lượng và loại nhân sự cần thiết được liệt kê cùng nhau với thời lượng mà chúng cần thiết  5.1.3 Resource acquisition plan (Kế hoạch mua lại nguồn lực.)  Cách thức có được các tài nguyên cần thiết, bao gồm phần cứng, phần mềm, hợp đồng dịch vụ và dịch vụ quản trị, được đưa ra ở đây.  5.1.4 Project staff training plan (Kế hoạch đào tạo nhân viên dự án)  Tất cả các khóa đào tạo cần thiết để hoàn thành dự án thành công được liệt kê trong tiểu mục này.  5.2 Work plan.  5.2.1 Work activities.( Hoạt động công việc)  Trong tiểu mục này, các hoạt động công việc được chỉ rõ, xuống cấp độ nhiệm vụ nếu thích hợp.  5.2.2 Schedule allocation.( Phân bổ lịch trình )  Nói chung, các gói công việc phụ thuộc lẫn nhau và phụ thuộc nhiều hơn vào các sự kiện bên ngoài. Ví dụ: quy trình thực hiện tuân theo quy trình công việc thiết kế và trước khi thử nghiệm sản phẩm. Trong tiểu mục này, các phụ thuộc liên quan được chỉ định  5.2.3 Resource allocation (Phân bổ nguồn lực)  Các nguồn lực khác nhau được liệt kê trước đây được phân bổ cho các chức năng, hoạt động và nhiệm vụ thích hợp của dự án.  5.2.4 Budget allocation.( Phân bổ ngân sách)  Trong tiểu mục này, ngân sách tổng thể được chia nhỏ ở các cấp chức năng, hoạt động và nhiệm vụ của dự án.  5.3 Control plan.( Kế hoạch kiểm soát)  5.3.1 Requirements control plan(Kế hoạch kiểm soát các yêu cầu)  Như được mô tả trong Phần B của cuốn sách này, trong khi một sản phẩm phần mềm đang được phát triển, các yêu cầu thường xuyên thay đổi. Các cơ chế giám sát và kiểm soát các thay đổi đối với các yêu cầu được đưa ra trong phần này.  5.3.2 Schedule control plan (Kế hoạch kiểm soát lịch trình)  Trong tiểu mục này, các cơ chế để đo lường tiến độ được liệt kê, cùng với mô tả về các hành động cần thực hiện nếu tiến độ thực tế chậm hơn tiến độ dự kiến.  5.3.3 Budget control plan (Kế hoạch kiểm soát ngân sách)  Điều quan trọng là chi tiêu không được vượt quá số tiền dự trù. Các cơ chế kiểm soát để giám sát khi chi phí thực tế vượt quá chi phí dự trù, cũng như các hành động cần thực hiện nếu điều này xảy ra, được mô tả trong tiểu mục này.  5.3.4 Quality control plan (Kế hoạch kiểm soát chất lượng)  Các cách thức mà chất lượng được đo lường và kiểm soát được mô tả trong tiểu mục này.  5.3.5 Reporting plan(Kế hoạch báo cáo)  Để giám sát các yêu cầu, tiến độ, ngân sách và chất lượng, cần phải có cơ chế báo cáo. Các cơ chế này được mô tả trong tiểu mục này.  5.3.6 Metrics collection plan (Kế hoạch thu thập số liệu)  Như đã giải thích trong Phần 5.5, không thể quản lý quá trình phát triển mà không đo lường các chỉ số liên quan. Các chỉ số sẽ được thu thập được liệt kê trong tiểu mục này.  5.4 Risk management plan (Kế hoạch quản lý rủi ro)  Rủi ro phải được xác định, ưu tiên, giảm thiểu và theo dõi. Tất cả các khía cạnh của quản lý rủi ro được mô tả trong phần này.  5.5 Project close-out plan (Kế hoạch kết thúc dự án)  Các hành động cần thực hiện sau khi dự án hoàn thành, bao gồm phân công lại nhân viên và lưu trữ hiện vật, được trình bày ở đây.  6 Technical process plans (Kế hoạch quy trình kỹ thuật)  6.1 Process model (Mô hình quy trình)  Trong phần này, mô tả chi tiết được đưa ra về vòng đời mô hình sẽ được sử dụng.  6.2 Methods, tools, and techniques (Phương pháp, công cụ và kỹ thuật)  Các phương pháp luận phát triển và ngôn ngữ lập trình sẽ được sử dụng được mô tả ở đây  6.3 Infrastructure plan (Kế hoạch cơ sở hạ tầng)  Các khía cạnh kỹ thuật của phần cứng và phần mềm được mô tả chi tiết trong phần này. Các hạng mục cần được đề cập bao gồm hệ thống máy tính (phần cứng, hệ điều hành, mạng và phần mềm) được sử dụng để phát triển sản phẩm phần mềm, cũng như các hệ thống máy tính mục tiêu mà sản phẩm phần mềm sẽ được chạy và các công cụ CASE sẽ được sử dụng .  6.4 Product acceptance plan (Kế hoạch nghiệm thu sản phẩm)  Để đảm bảo rằng sản phẩm phần mềm đã hoàn thành sẽ vượt qua kiểm tra chấp nhận của nó, các tiêu chí chấp nhận phải được soạn thảo, khách hàng phải đồng ý với các tiêu chí bằng văn bản và các nhà phát triển sau đó phải đảm bảo rằng các tiêu chí này thực sự được đáp ứng. Cách thức mà ba giai đoạn này của quá trình chấp nhận sẽ được thực hiện được mô tả trong phần này.  7 Supporting process plans (Các kế hoạch quy trình hỗ trợ)  7.1 Configuration management plan (Kế hoạch quản lý cấu hình)  Trong phần này, mô tả chi tiết được đưa ra về các phương tiện mà tất cả các hiện vật được đưa vào quản lý cấu hình.  7.2 Testing plan (Kế hoạch kiểm tra)  Kiểm thử, giống như tất cả các khía cạnh khác của phát triển phần mềm, cần lập kế hoạch cẩn thận  7.3 Documentation plan (Kế hoạch tài liệu)  Phần mô tả tài liệu các loại, có được giao cho khách hàng khi kết thúc dự án hay không, được bao gồm trong phần này.  7.4 Quality assurance plan (Kế hoạch đảm bảo chất lượng)  Phần này bao gồm tất cả các khía cạnh của đảm bảo chất lượng, bao gồm kiểm tra, tiêu chuẩn và đánh giá.  7.5 Reviews and audits plan (Kế hoạch đánh giá và kiểm tra)  Chi tiết về cách thức tiến hành đánh giá được trình bày trong phần này.  7.6 Problem resolution plan (Kế hoạch giải quyết vấn đề)  Trong quá trình phát triển một sản phẩm phần mềm, tất cả các vấn đề đều có thể phát sinh. Ví dụ, việc xem xét thiết kế có thể làm sáng tỏ một lỗi nghiêm trọng trong quy trình phân tích đòi hỏi những thay đổi lớn đối với hầu hết tất cả các hiện vật đã được hoàn thành. Trong phần này, cách các vấn đề như vậy được xử lý được mô tả.  7.7 Subcontractor management plan (Kế hoạch quản lý nhà thầu phụ)  Phần này có thể áp dụng khi các nhà thầu phụ cung cấp các sản phẩm công việc nhất định. Sau đó, cách tiếp cận để lựa chọn và quản lý nhà thầu phụ sẽ xuất hiện ở đây.  7.8 Process improvement plan (Kế hoạch cải tiến quy trình)  Các chiến lược cải tiến quy trình được bao gồm trong phần này.  8 Additional plans (kế hoạch bổ sung)  Đối với một số dự án nhất định, các thành phần bổ sung có thể cần xuất hiện trong kế hoạch. Về khuôn khổ IEEE, chúng xuất hiện ở cuối kế hoạch. Các thành phần bổ sung có thể bao gồm kế hoạch bảo mật, kế hoạch an toàn, kế hoạch chuyển đổi dữ liệu, kế hoạch cài đặt và kế hoạch bảo trì sau phân phối của dự án phần mềm.  **9.6 lập kế hoạch kiểm tra (planning test)**  Một thành phần của SPMP thường bị bỏ qua là lập kế hoạch kiểm tra. Giống như mọi hoạt động khác của phát triển phần mềm, kiểm thử phải được lập kế hoạch. SPMP phải bao gồm các tài nguyên để kiểm tra và lịch trình chi tiết phải chỉ ra rõ ràng việc kiểm tra sẽ được thực hiện trong mỗi quy trình làm việc  Nếu không có một kế hoạch thử nghiệm, một dự án có thể trở nên tồi tệ theo một số cách. Ví dụ, trong quá trình thử nghiệm sản phẩm (Phần 3.7.4), nhóm SQA phải kiểm tra xem mọi khía cạnh của tài liệu đặc điểm kỹ thuật, khi được khách hàng ký tên, đã được triển khai trong sản phẩm đã hoàn thành chưa. Một cách tốt để hỗ trợ nhóm SQA trong nhiệm vụ này là yêu cầu sự phát triển phải được theo dõi (Phần 3.7). Có nghĩa là, phải có khả năng kết nối mỗi câu lệnh trong tài liệu đặc tả với một phần của thiết kế và mỗi phần của thiết kế phải được phản ánh rõ ràng trong mã. Một kỹ thuật để đạt được điều này là đánh số từng câu lệnh trong tài liệu đặc tả và đảm bảo rằng những con số này được lấy lại trong cả thiết kế và mã kết quả. Tuy nhiên, nếu kế hoạch kiểm tra không chỉ rõ rằng việc này phải được thực hiện, thì rất khó có khả năng các tạo tác phân tích, thiết kế và mã sẽ được dán nhãn thích hợp. Do đó, khi kiểm tra sản phẩm cuối cùng được thực hiện, nhóm SQA sẽ rất khó để xác định rằng sản phẩm là sự triển khai hoàn chỉnh của các cation cụ thể. Trên thực tế, việc truy xuất nguồn gốc nên bắt đầu từ các yêu cầu; mỗi câu lệnh trong phần tạo tác yêu cầu (hoặc mỗi phần của nguyên mẫu nhanh) phải được kết nối với một phần của phần tạo tác phân tích.  Một khía cạnh khác của việc kiểm tra là danh sách chi tiết các lỗi được phát hiện trong quá trình kiểm tra. Giả sử rằng một nhóm đang kiểm tra các đặc trưng của một sản phẩm. Danh sách các lỗi được sử dụng theo hai cách. Đầu tiên, số liệu thống kê lỗi từ cuộc kiểm tra này phải được so sánh với số liệu thống kê lỗi trung bình tích lũy từ các cuộc kiểm tra thông số kỹ thuật trước đó. Sai lệch so với các định mức trước đây cho thấy có vấn đề trong dự án. Thứ hai, thống kê lỗi từ việc kiểm tra cation đặc trưng hiện tại phải được chuyển sang kiểm tra thiết kế và mã của sản phẩm. Rốt cuộc, nếu có một số lượng lớn các lỗi của một loại cụ thể, có thể không phải tất cả chúng đều được phát hiện trong quá trình kiểm tra các thông số kỹ thuật và việc kiểm tra thiết kế và mã cung cấp thêm cơ hội để xác định bất kỳ lỗi nào còn lại của loại này. Tuy nhiên, trừ khi kế hoạch kiểm tra quy định rằng các chi tiết của tất cả các lỗi phải được ghi lại cẩn thận, nếu không nhiệm vụ này sẽ không được thực hiện.  Một cách quan trọng để kiểm tra mô-đun mã được gọi là kiểm thử hộp đen (Phần 15.11), trong đó mã được thực thi với các trường hợp kiểm thử dựa trên các cation cụ thể. Các thành viên của nhóm SQA đọc qua các trích dẫn cụ thể và vẽ các trường hợp thử nghiệm để kiểm tra xem mã có tuân theo tài liệu cation cụ thể hay không. Thời điểm tốt nhất để tạo ra các trường hợp kiểm thử hộp đen là vào cuối quy trình phân tích, khi các chi tiết của tài liệu cụ thể vẫn còn mới trong tâm trí của các thành viên của nhóm SQA đã kiểm tra chúng. Tuy nhiên, trừ khi kế hoạch kiểm tra quy định rõ ràng rằng các trường hợp thử nghiệm hộp đen sẽ được chọn vào thời điểm này, nếu không thì chỉ một vài trường hợp thử nghiệm hộp đen sẽ được ném cùng nhau sau này. Đó là, một giới hạn số lượng các trường hợp thử nghiệm sẽ được lắp ráp nhanh chóng chỉ khi áp lực bắt đầu tăng từ nhóm lập trình để nhóm SQA phê duyệt các mô-đun của họ để chúng có thể được tích hợp vào toàn bộ sản phẩm. Kết quả là, chất lượng của sản phẩm nói chung bị ảnh hưởng.  Do đó, mọi kế hoạch thử nghiệm phải chỉ rõ thử nghiệm nào sẽ được thực hiện, khi nào nó được thực hiện và cách thức thực hiện. Kế hoạch kiểm tra như vậy là một phần thiết yếu của phần 7.2 của SPMP. Nếu không có nó, chất lượng của sản phẩm tổng thể chắc chắn sẽ bị ảnh hưởng.  **9.7 Lập kế hoạch các dự án hướng đối tượng**  Việc sử dụng mô hình hướng đối tượng dẫn đến một sản phẩm bao gồm một số các thành phần độc lập nhỏ hơn, cụ thể là các lớp. Điều này làm cho việc lập kế hoạch dễ dàng hơn đáng kể, trong đó ước tính chi phí và thời lượng có thể được tính toán dễ dàng và chính xác hơn cho các đơn vị nhỏ hơn. Tất nhiên, các ước tính phải tính đến việc một sản phẩm không chỉ là tổng các bộ phận của nó. Các thành phần riêng biệt không hoàn toàn độc lập; chúng có thể gọi lẫn nhau, và những hiệu ứng này không được bỏ qua.  **Các kỹ thuật ước tính chi phí và thời lượng được mô tả trong chương này có áp dụng được cho mô hình hướng đối tượng không?** COCOMO II được thiết kế để xử lý công nghệ phần mềm hiện đại, bao gồm cả hướng đối tượng, nhưng còn các số liệu trước đó như điểm chức năng và COCOMO trung gian thì sao? Trong trường hợp COCOMO trung gian, cần có những thay đổi nhỏ đối với một số nhân chi phí. Ngoài ra, các công cụ ước lượng của mô hình cổ điển dường như hoạt động khá tốt trên các dự án hướng đối tượng - với điều kiện là không được sử dụng lại.  Tái sử dụng đi vào mô hình hướng đối tượng theo hai cách: tái sử dụng các thành phần hiện có trong quá trình phát triển và sản xuất có chủ ý (trong dự án hiện tại) các thành phần sẽ được sử dụng lại trong các sản phẩm trong tương lai. Cả hai hình thức tái sử dụng đều ảnh hưởng đến quá trình lập dự toán. Sử dụng lại trong thời gian phát triển rõ ràng làm giảm chi phí và thời gian. Các công thức đã được công bố cho thấy sự tiết kiệm như một chức năng của việc tái sử dụng này,nhưng những kết quả này liên quan đến mô hình cổ điển. Hiện tại, không có thông tin về chi phí và thời lượng thay đổi như thế nào khi việc sử dụng lại được sử dụng trong quá trình phát triển sản phẩm hướng đối tượng.  Bây giờ chúng ta chuyển sang mục tiêu tái sử dụng các phần của dự án hiện tại. Có thể mất khoảng ba lần để thiết kế, triển khai, kiểm tra và ghi lại thành phần có thể tái sử dụng như một thành phần không thể sử dụng tương tự. Ước tính chi phí và thời lượng phải được sửa đổi để kết hợp lao động bổ sung này và SPMP nói chung phải được điều chỉnh để kết hợp hiệu quả của nỗ lực tái sử dụng. Do đó, hai hoạt động tái sử dụng hoạt động ngược chiều nhau. Việc tái sử dụng các thành phần hiện có làm giảm nỗ lực tổng thể trong việc phát triển một sản phẩm hướng đối tượng, trong khi việc thiết kế các thành phần để tái sử dụng trong các sản phẩm trong tương lai làm tăng nỗ lực. Người ta mong đợi rằng, về lâu dài, khoản tiết kiệm do việc tái sử dụng các lớp học sẽ lớn hơn chi phí phát triển ban đầu, và đã có một số bằng chứng ủng hộ điều này.  9.8 Training Requirements (Yêu cầu đào tạo)  Khi chủ đề đào tạo được đưa ra trong các cuộc thảo luận với khách hàng, câu trả lời phổ biến là, "Chúng tôi không cần phải lo lắng về việc đào tạo cho đến khi sản phẩm hoàn thiện, sau đó chúng tôi có thể đào tạo người dùng." Đây là một nhận xét hơi đáng tiếc, ngụ ý rằng nó chỉ có người dùng yêu cầu đào tạo. Trên thực tế, các thành viên của nhóm phát triển cũng có thể cần đào tạo, bắt đầu từ đào tạo về lập kế hoạch và ước lượng phần mềm. Khi các kỹ thuật phát triển phần mềm mới, chẳng hạn như kỹ thuật thiết kế mới hoặc quy trình kiểm thử, được sử dụng, thì mọi thành viên của nhóm sử dụng kỹ thuật mới phải được đào tạo.  Việc giới thiệu mô hình hướng đối tượng có những hệ quả đào tạo chính. Việc giới thiệu các công cụ phần cứng hoặc phần mềm như máy trạm hoặc môi trường tích hợp (xem Phần 15.24.2) cũng cần được đào tạo. Lập trình viên có thể cần được đào tạo về hệ điều hành của máy được sử dụng để phát triển sản phẩm cũng như ngôn ngữ thực thi. Việc đào tạo chuẩn bị tài liệu thường xuyên bị bỏ qua, bằng chứng là chất lượng kém của rất nhiều tài liệu. Các nhà khai thác máy tính chắc chắn yêu cầu một số loại đào tạo để có thể chạy sản phẩm mới; họ cũng có thể yêu cầu đào tạo thêm nếu phần cứng mới được sử dụng.  Việc đào tạo bắt buộc có thể đạt được theo một số cách. Cách dễ nhất và ít gây gián đoạn nhất là đào tạo nội bộ, bởi các nhân viên hoặc chuyên gia tư vấn. Nhiều công ty cung cấp nhiều khóa đào tạo khác nhau, và các trường cao đẳng thường cung cấp các khóa đào tạo vào buổi tối.  Các khóa học dựa trên World Wide Web là một lựa chọn thay thế khác.  Khi nhu cầu đào tạo đã được xác định và lập kế hoạch đào tạo, kế hoạch này phải được đưa vào SPMP.  9.9 Documentation Standards  Sự phát triển của một sản phẩm phần mềm đi kèm với rất nhiều tài liệu. Jones nhận thấy rằng 28 trang tài liệu được tạo trên mỗi 1000 hướng dẫn (KDSI) cho một sản phẩm thương mại nội bộ của IBM có kích thước khoảng 50 KDSI và khoảng 66 trang trên mỗi KDSI cho một sản phẩm phần mềm thương mại có cùng kích thước. Hệ điều hành IMS / 360 Phiên bản 2.3 có kích thước khoảng 166 KDSI và 157 trang tài liệu trên mỗi KDSI đã được tạo ra. Tài liệu thuộc nhiều loại khác nhau, bao gồm lập kế hoạch, kiểm soát, tài chính và kỹ thuật [Jones, 1986a]. Ngoài các loại tài liệu này, bản thân mã nguồn cũng là một dạng tài liệu; nhận xét trong mã tạo thành tài liệu bổ sung.  Một phần đáng kể của nỗ lực phát triển phần mềm được tiếp thu bởi tài liệu. Một cuộc khảo sát trên 63 dự án phát triển và 25 dự án bảo trì sau giao hàng cho thấy cứ 100 giờ dành cho các hoạt động liên quan đến mã thì 150 giờ dành cho các hoạt động liên quan đến tài liệu [Boehm, 1981]. Đối với các sản phẩm TRW lớn, tỷ lệ thời gian dành cho các hoạt động liên quan đến tài liệu đã tăng lên 200 giờ trên 100 giờ mã hóa [Boehm et al., 1984].  Các tiêu chuẩn là cần thiết cho mọi loại tài liệu. Ví dụ, tính thống nhất trong tài liệu thiết kế làm giảm sự hiểu lầm giữa các thành viên trong nhóm và hỗ trợ nhóm SQA. Mặc dù nhân viên mới phải được đào tạo về các tiêu chuẩn tài liệu, nhưng không cần đào tạo thêm khi nhân viên hiện tại chuyển từ dự án này sang dự án khác trong tổ chức. Từ quan điểm của bảo trì sau giao hàng, các tiêu chuẩn mã hóa thống nhất hỗ trợ các lập trình viên bảo trì hiểu được mã nguồn. Tiêu chuẩn hóa thậm chí còn quan trọng hơn đối với hướng dẫn sử dụng, bởi vì chúng phải được đọc bởi nhiều cá nhân, một số ít là chuyên gia máy tính. IEEE đã phát triển một tiêu chuẩn cho hướng dẫn sử dụng (IEEE Standard 1063 for Software User Documentation).  Là một phần của quá trình lập kế hoạch, các tiêu chuẩn phải được thiết lập cho tất cả các tài liệu được tạo ra trong quá trình sản xuất phần mềm. Các tiêu chuẩn này được kết hợp trong SPMP.  Khi một tiêu chuẩn hiện có được sử dụng, chẳng hạn như Tiêu chuẩn ANSI / IEEE cho Tài liệu Kiểm tra Phần mềm [ANSI / IEEE 829, 1991], tiêu chuẩn được liệt kê trong phần 2 của PMP (tài liệu tham khảo). Nếu một tiêu chuẩn được viết đặc biệt cho nỗ lực phát triển, sau đó nó xuất hiện trong phần 6.2 (phương pháp, công cụ và kỹ thuật).  Tài liệu là một khía cạnh thiết yếu của nỗ lực sản xuất phần mềm. Theo một nghĩa rất thực tế, sản phẩm là tài liệu, bởi vì không có tài liệu thì sản phẩm không thể được duy trì. Lập kế hoạch cho nỗ lực tài liệu đến từng chi tiết, và sau đó đảm bảo rằng kế hoạch được tuân thủ, là một thành phần quan trọng của quá trình sản xuất phần mềm thành công.  9.10 CASE Tools for Planning and Estimating (CASE Công cụ lập kế hoạch và ước tính)  Một số công cụ có sẵn để tự động hóa COCOMO trung gian và COCOMO II. Để tăng tốc độ tính toán khi giá trị của một tham số được sửa đổi, một số triển khai COCOMO trung gian đã được triển khai trong các ngôn ngữ bảng tính như Lotus 1-2-3 và Excel. Để phát triển và cập nhật kế hoạch, một trình xử lý văn bản là điều cần thiết.  Các công cụ thông tin quản lý cũng rất hữu ích cho việc lập kế hoạch. Ví dụ, giả sử rằng một tổ chức phần mềm lớn có 150 lập trình viên. Một công cụ lập kế hoạch có thể giúp các nhà lập kế hoạch theo dõi những người lập trình nào đã được giao cho các nhiệm vụ cụ thể và những công việc nào có sẵn cho dự án hiện tại  Các loại thông tin quản lý tổng quát hơn cũng là cần thiết. Một số công cụ quản lý có sẵn trên thị trường có thể được sử dụng để hỗ trợ quá trình lập kế hoạch và ước tính cũng như để giám sát toàn bộ quá trình phát triển. Chúng bao gồm MacProject và Microsoft Project.  9.11 Testing the Software Project Management Plan (Kiểm tra Kế hoạch Quản lý Dự án Phần mềm)  Như đã chỉ ra ở đầu chương này, một lỗi trong kế hoạch quản lý dự án phần mềm có thể có những tác động nghiêm trọng về mặt tài chính đối với các nhà phát triển. Điều quan trọng là tổ chức phát triển không được đánh giá quá cao hoặc đánh giá thấp chi phí của dự án hoặc thời gian của nó. Vì lý do này, toàn bộ SPMP phải được nhóm SQA kiểm tra trước khi đưa ra các ước tính cho khách hàng. Cách tốt nhất để kiểm tra kế hoạch là kiểm tra kế hoạch.  Nhóm kiểm tra kế hoạch phải xem xét SPMP một cách chi tiết, đặc biệt chú ý đến ước tính chi phí và thời gian. Để giảm thiểu rủi ro hơn nữa, bất kể số liệu được sử dụng là gì, ước tính thời lượng và chi phí phải được tính toán độc lập bởi một thành viên của nhóm SQA ngay sau khi các thành viên của nhóm lập kế hoạch xác định ước tính của họ. |
|  |