







PHÁT TRIỂN VẬN HÀNH BẢO TRÌ PHẦN MỀM

ThS. NGUYỄN THỊ THANH TRÚC
Khoa Công Nghệ Phần Mềm

Nội dung (Chương 2)

- 
-  **Nền tảng của sự thay đổi phần mềm**
 -  **Mối liên quan kinh tế của việc cập nhật phần mềm**
 -  **Giải pháp tiềm năng đối với vấn đề bảo trì**
 -  **Thảo luận và làm bài tập**
 -  **Q&A**

Chương 2: NỀN TẢNG CỦA SỰ THAY ĐỔI PHẦN MỀM

- 2.1 Nền tảng của sự thay đổi phần mềm
- 2.2 Mối liên quan kinh tế của việc cập nhật phần mềm
- 2.3 Giải pháp tiềm năng đối với vấn đề bảo trì

Chương 2:

NỀN TẢNG CỦA SỰ THAY ĐỔI PHẦN MỀM

1. NỀN TẢNG SỰ THAY ĐỔI PHẦN MỀM

- Sự thay đổi phần mềm
- Phân loại sự thay đổi
 - ✓ Corrective Change (Thay đổi hiệu chỉnh)
 - ✓ Adaptive Change (Thay đổi thích nghi)

2. MỐI LIÊN HỆ KINH TẾ CỦA VIỆC CẬP NHẬT PHẦN MỀM

- Giới hạn đối với sự thay đổi
- Hạn chế tài nguyên
- Chất lượng của hệ thống tồn tại
- Chiến lược tổ chức
- Tính trì trệ (không thay đổi)

3. GIẢI PHÁP TIỀM NĂNG ĐỐI VỚI VẤN ĐỀ BẢO TRÌ

- cấp phát Ngân sách và Nỗ lực (Effort)
- Hoàn chỉnh thay thế hệ thống
- Bảo trì hệ thống tồn tại

2.1 NỀN TẢNG CỦA SỰ THAY ĐỔI PHẦN MỀM

❑ Sự thay đổi phần mềm

- Tiến hoá phần mềm

 - ✓ Loại phần mềm

 - ✓ Luật tiến hoá

❑ Phân loại những thay đổi

- Thay đổi hiệu chỉnh (Corrective Change)

- Thay đổi thích nghi (Adaptive Change)

- Thay đổi hoàn chỉnh (Perfective Change)

- Thay đổi ngăn ngừa (Preventive Change)

❑ Tầm quan trọng của việc phân loại

❑ Cho ví dụ cho mỗi loại?

Định nghĩa

□ E-type system

- the criteria for acceptability is based on its performance in a real world situation.

□ S-type system

- the criteria for acceptability is that it is correct relative to an absolute specification

□ On-going support

- a service offered to customers to assist their continuing use of a product

□ Ripple effect

- consequences of an action in one place, occurring elsewhere

Luật đầu tiên của Công nghệ phần mềm

“No matter where you are in the system life cycle, the system will change, and the desire to change it will persist throughout the life cycle”

Bersoff et al. (1980)

Nguồn của sự thay đổi

- ❑ Những điều kiện kinh doanh và thị trường mới gây ra thay đổi những yêu cầu sản phẩm và qui tắc nghiệp vụ (business rules)
- ❑ Khách hàng mới cần thay đổi nhu cầu dữ liệu, chức năng hay dịch vụ phân phối bởi hệ thống
- ❑ Tái tổ chức hay cắt giảm kinh doanh mà thay đổi ưu tiên và cấu trúc nhóm (team)
- ❑ Ràng buộc ngân sách và lịch trình gây ra tái định nghĩa hệ thống
- ❑ **HÀU HẾT SỰ THAY ĐỔI LÀ CÓ LÝ DO CHÍNH ĐÁNG**

Thay đổi hiệu chỉnh (Corrective Change)

- ❑ Thay đổi hiệu chỉnh ám chỉ đến cập nhật khởi nguồn từ dò lỗi trong phần mềm
- ❑ Dò lỗi có thể kết quả từ lỗi thiết kế, lỗi logic hay lỗi chương trình
 - **Lỗi thiết kế (Design)**
 - ✓ Thay đổi không chính xác, hoàn chỉnh, truyền thông sai lệch hay yêu cầu thay đổi hiểu nhầm
 - **Lỗi logic**
 - ✓ Kết quả từ kiểm thử không hợp lệ, dữ liệu thử, thực thi không đúng thiết kế, lỗi luồng logic
 - **Lỗi cài đặt (coding)**
 - ✓ Gây ra bởi thực thi không chính xác thiết kế chi tiết mức logic và sử dụng không chính xác logic của chương trình nguồn.

Thay đổi thích nghi (Adaptive change)

- ❑ Để thích nghi phần mềm với bất kỳ thay đổi môi trường
- ❑ Kết quả khởi nguồn từ loại bỏ software, hardware khác biệt, phần mềm nền, trình biên dịch, hệ điều hành
- ❑ Kết quả từ thay đổi máy ảo (virtual machine), thêm tiện ích cho hệ điều hành, lưu không gian đĩa, khôi phục sự cố lỗi
- ❑ Thay đổi hệ thống phần mềm liên quan khu vực (areas), ví dụ: giao dịch ngân hàng, kinh doanh

Thay đổi hoàn chỉnh (Perfective Change)

- ❑ Mở rộng những yêu cầu tồn tại của hệ thống
- ❑ Xuất phát từ thêm mới chức năng hay tình chế chức năng đã cung cấp, một số chức năng có thể bị bỏ đi.

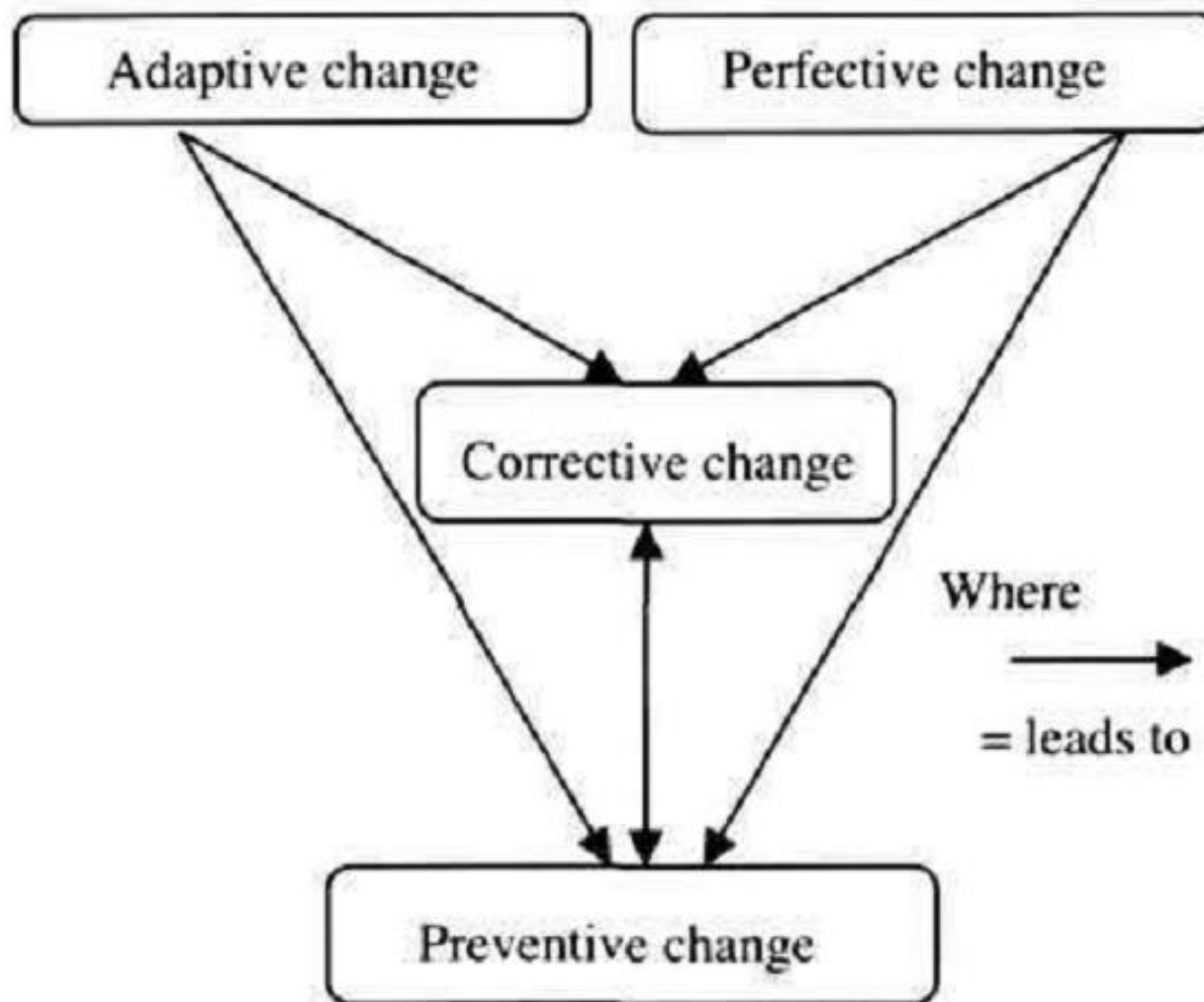
Thay đổi ngăn ngừa (Preventive change)

- ❑ Đề cập đến vấn đề làm hỏng (giảm giá trị) cấu trúc
- ❑ Ngăn chức năng lạ hay cải thiện khả năng bảo trì của phần mềm
- ❑ Gồm: tái cấu trúc mã nguồn, tối ưu mã nguồn và cập nhật sưu liệu

Tầm quan trọng của phân loại thay đổi phần mềm

- ❑ Về nguyên tắc, những hoạt động bảo trì được phân loại cách riêng lẻ
- ❑ Tuy nhiên trong thực tế, những thay đổi luôn quần vào nhau
- ❑ Một vài thay đổi đòi đáp ứng nhanh hơn những cái khác
- ❑ Việc hiểu bản chất của thay đổi được thực thi mang lại độ ưu tiên hiệu quả của những yêu cầu thay đổi.

Mối liên hệ giữa các loại thay đổi phần mềm



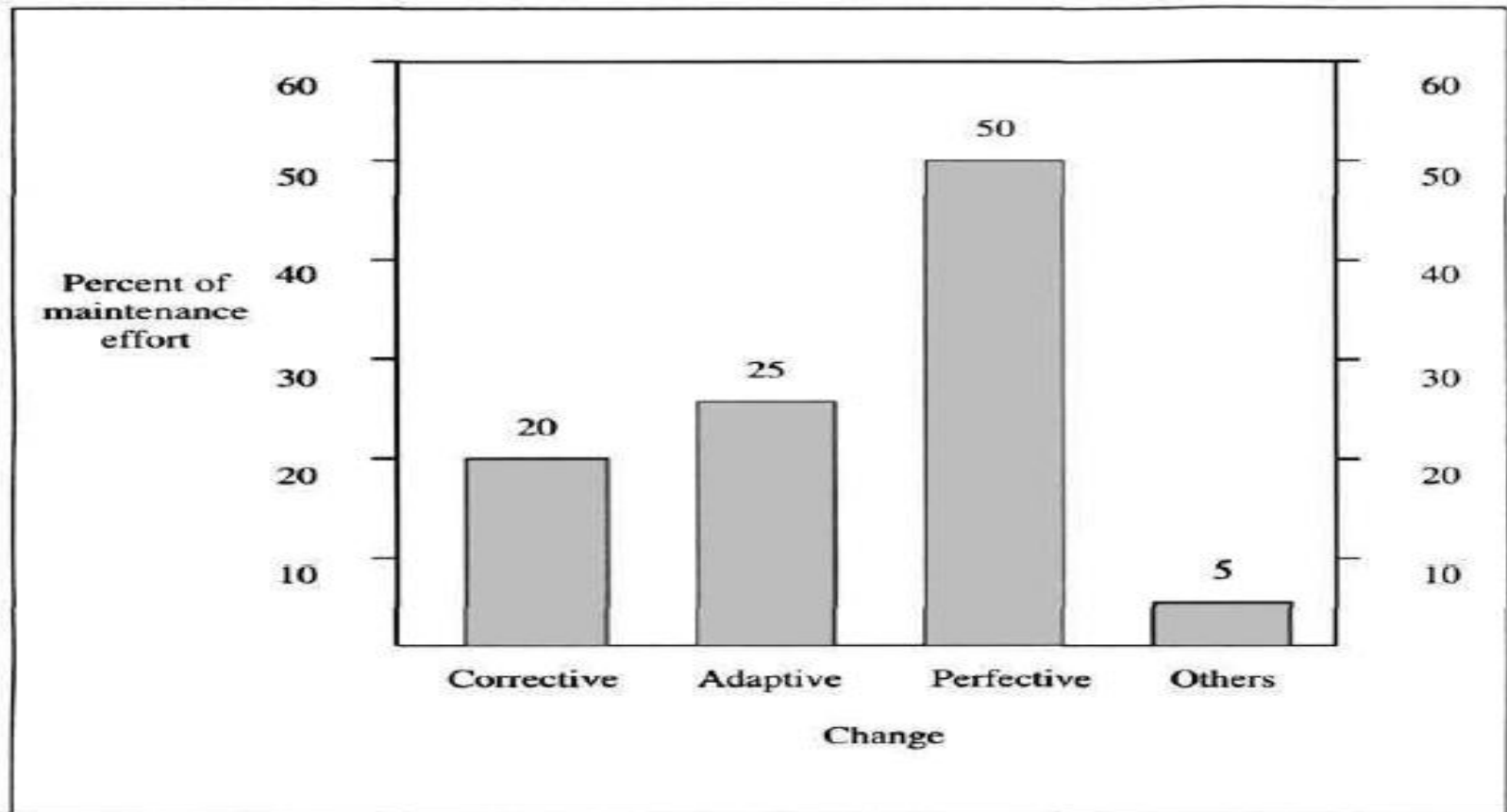


Figure 4.2 Expenditure on the different changes

Bảo trì và SDLC

- ❑ Qui trình phát triển phần mềm Waterfall, chúng ta có họp ở mỗi cuối qui trình và được bỏ qua trong mô tả qui trình
- ❑ Chu trình cải tiến hơn như Spiral Model, bảo trì phù hợp nhiều vị trí nổi bật
- ❑ Bảo trì vẫn liên quan đến khía cạnh của SDLC
- ❑ Ví dụ: Pressman không có chương cụ thể về Bảo Trì, Somerville có 15 trang trong 742 trang

Loại chương trình

❑ Chương trình S-type (“Specifiable”)

Source: Adapted from Lehman 1980, pp1061-1063

- Vấn đề được khẳng định hình thức và hoàn chỉnh
- Chấp nhận: Chương trình có chính xác như đặc tả của nó?
- Phần mềm này không tiến triển.

✓ **Thay đổi đặc tả định nghĩa vấn đề mới, như vậy một chương trình mới**

❑ Chương trình P-type (“Problem-solving”)

- Khẳng định không chính xác vấn đề thế giới thực
- Chấp nhận: Chương trình là giải pháp có thể chấp nhận đối với vấn đề?
- Phần mềm này xem như tiến triển liên tục

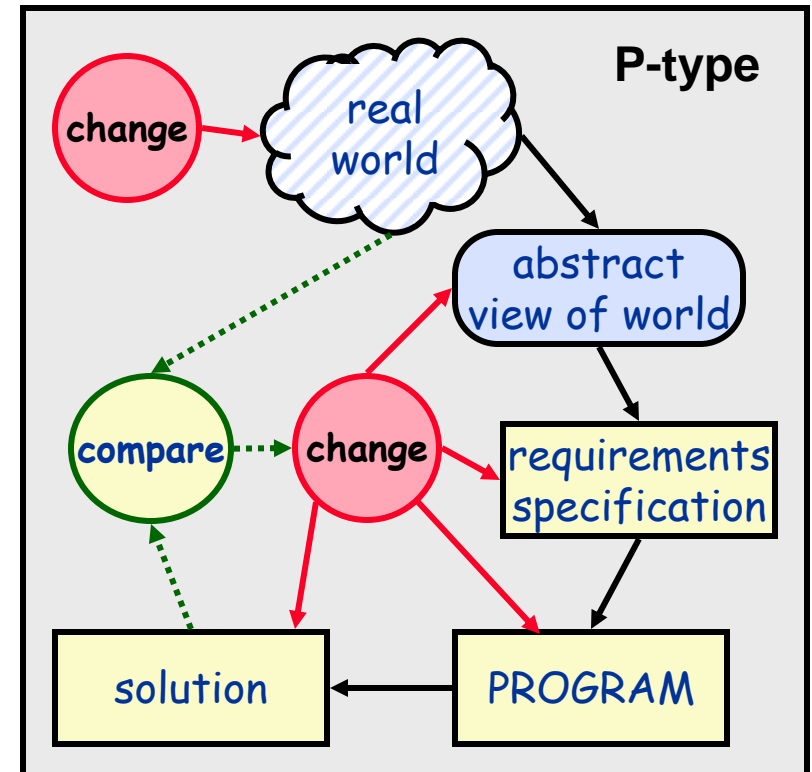
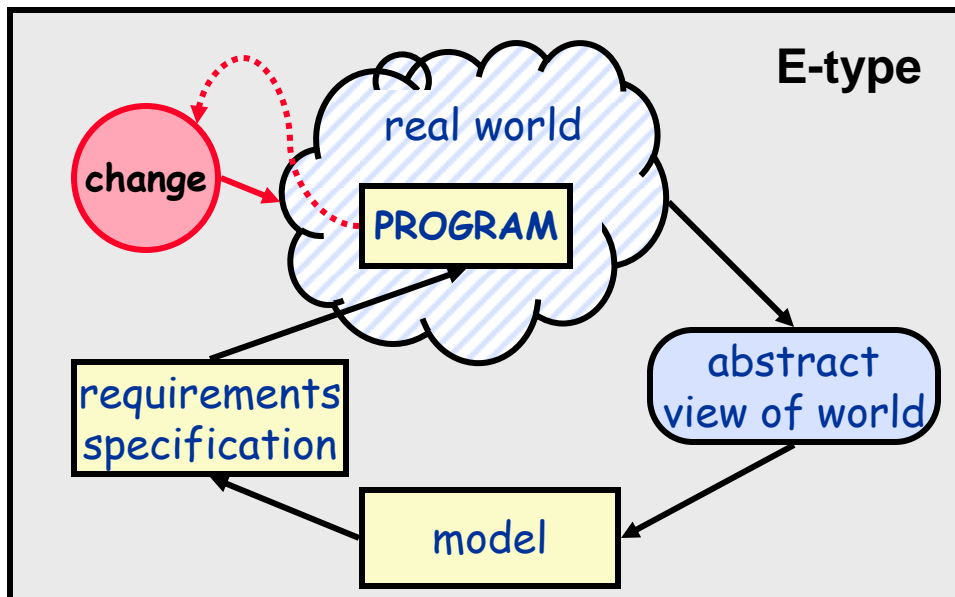
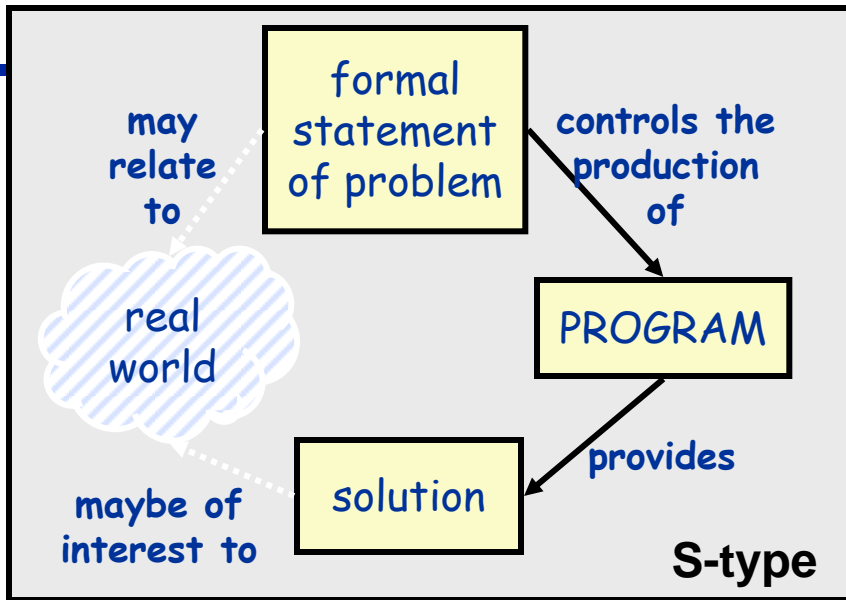
✓ **Bởi giải pháp không bao giờ hoàn hảo, và có thể cải tiến**

✓ **Bởi thế giới thực thay đổi và vấn đề thay đổi**

❑ Chương trình E-type (“Embedded”)

- Một hệ thống trở thành một phần thế giới nó được mô hình hoá
- Chấp nhận: phụ thuộc toàn bộ vào ý kiến và phán xét
- Phần mềm này vốn đã tiến hoá

✓ **Thay đổi trong phần mềm và thế giới tác động lẫn nhau**



Thảo luận: nêu ví dụ hệ thống phần mềm tương ứng 3 loại trên

Loại Bảo trì phần mềm

- ❑ Làm thế nào và tại sao chúng ta tốn khá nhiều thời gian và nỗ lực cho việc bảo trì?
- ❑ Bảo trì thì nhiều vấn đề hơn việc fix bug
- ❑ Phân chia thành ba loại chính
 - Corrective Maintenance (bảo trì hiệu chỉnh)
 - Adaptive Maintenance (Bảo trì thích nghi)
 - Perfective Maintenance (Bảo trì hoàn chỉnh)
- ❑ Ranh giới giữa các loại bảo trì khá mờ không rõ
- ❑ Chúng ta có thể định nghĩa các loại bảo trì khác – bảo trì ngăn ngừa

Các loại bảo trì phần mềm

Corrective Maintenance

fixing latent errors

includes temporary patches and
workarounds

Adaptive Maintenance

responding to external changes

changes in hardware platform
changes in support software

Perfective Maintenance

improving the as-delivered software

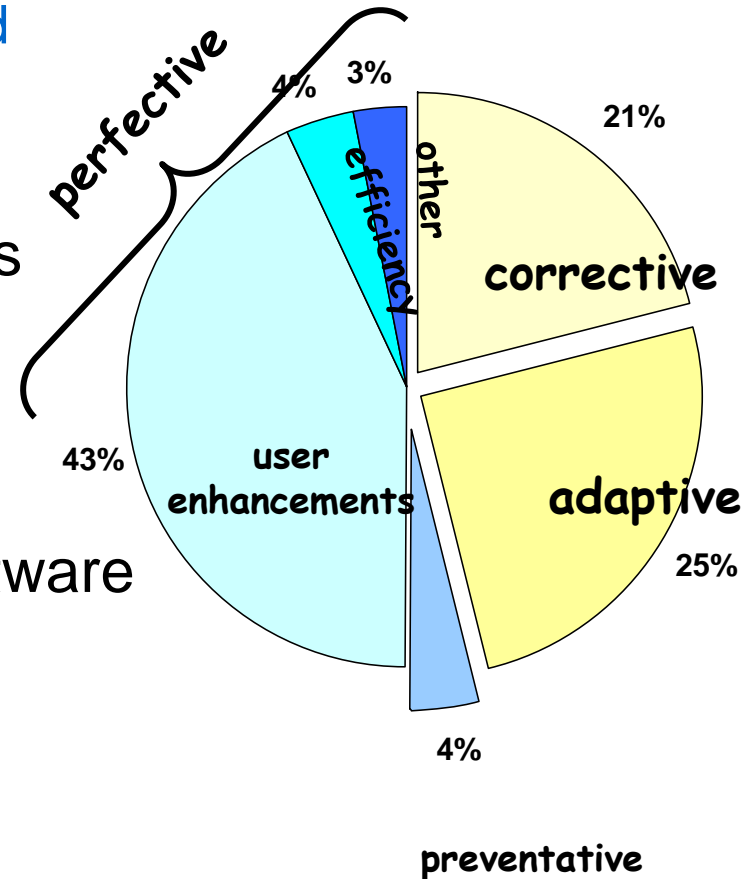
user enhancements
efficiency improvements

Preventative Maintenance

Improves (future) maintainability

Documenting, commenting, etc.

Source: Adapted from van Vliet, 1999, p449.



Bảo trì hiệu chỉnh (Corrective Maintenance)

- ❑ Tập trung vào fix lỗi
- ❑ Là qui trình có phản ứng lại
 - Lỗi và lỗi kết hợp nói chung cần được chính xác ngay lập tức hay trong tương lai gần
- ❑ Lỗi biến đổi theo chi phí để hiệu chỉnh:
 - Coding – thường tương đối rẻ
 - Design – Khá tốn kém khi chúng có thể đòi thay đổi vài thành phần chương trình
 - Requirements – Tốn kém nhất – có lẽ đòi tái thiết kế hệ thống mở rộng
- ❑ Thiết kế và Yêu cầu là nguồn chiếm khoảng 80% lỗi

Bảo trì hiệu chỉnh (2)

- ❑ Hiệu chỉnh lỗi có 20 đến 50% cơ hội đưa ra lỗi khác
- ❑ Nguyên nhân lỗi mới gồm :
 - Dấu vết tác động nơi mà sự thay đổi ở một nơi gây ra sự thay đổi vùng dường như không liên quan
 - Người sửa lỗi nói chung không phải là người viết code hay thiết kế hệ thống
- ❑ Hai loại bảo trì hiệu chỉnh
 - Sửa chữa khẩn cấp – thời gian ngắn- thường chương trình đơn, lỗi cần được sửa sớm như có thể
 - Sửa chữa theo lịch trình - lỗi không cần chú ý ngay, kiểm tra lại tất cả sửa chữa khẩn cấp

Ví dụ các thao tác yêu cầu thay đổi của một đặc tả chức năng

- ☐ Tạo yêu cầu mở.
- ☐ Khai báo file tác động
- ☐ Phê chuẩn file về thời gian và chi phí do chủ, người sử dụng ký.
- ☐ Hoàn thiện danh sách và kiểm soát về thay đổi của người điều hành dự án.
- ☐ File tài liệu liên quan đến thay đổi. Nếu tài liệu hoặc chương trình bị thay đổi, thì xác định ngày và các mục cập nhật đã hoàn thiện. Nếu các thủ tục hoặc thử nghiệm bị thay đổi, xác định các ngày mà việc sửa đổi xảy ra.
- ☐ Mẫu yêu cầu đóng file được chủ/người sử dụng thông qua.
- ☐ Tóm tắt các ngày tháng, quá trình và chi phí.

Bảo trì thích nghi (Adaptive Maintenance)

- ❑ Tiến hoá hệ thống nhằm đạt nhu cầu người dùng và doanh nghiệp
- ❑ Gây ra bởi
 - Nhu cầu nội bộ
 - Canh tranh bên ngoài
 - Những yêu cầu ngoài ví dụ thay đổi Luật
- ❑ Cần thiết đưa ra những yêu cầu mới cho hệ thống
- ❑ Như vậy chúng ta nên xử lý giống như phát triển mới theo hướng tiếp cận và phương pháp

Bảo trì hoàn chỉnh (Perfective Maintenance)

- ❑ Thành ngữ xưa “Nếu không hỏng, thì không sửa nó”
- ❑ Bảo trì hoàn chỉnh bỏ qua câu thành ngữ xưa
- ❑ Cải thiện chất lượng chương trình sẵn sàng hoạt động
- ❑ Mục tiêu đạt được
 - Giảm chi phí trong sử dụng hệ thống
 - Tăng khả năng bảo trì hệ thống
 - Gần hơn nhu cầu người dùng

Bảo trì hoàn chỉnh (2)

- ❑ Gồm tất cả nỗ lực để trau chuốt hay cải tiến chất lượng phần mềm và sơ liệu
- ❑ Important that the potential benefits of the perfective maintenance outweigh
 - Chi phí của bảo trì
 - Và chi phí cơ hội của cải tiến nơi khác hay dùng tài nguyên trong phát triển mới
- ❑ Như vậy, trước khi thực hiện bảo trì hoàn chỉnh, đầu tiên nên qua tiến trình phân tích
- ❑ Tuy nhiên, bảo trì hoàn chỉnh bé có thể những ảnh hưởng

Bảo trì ngăn ngừa (Preventative Maintenance)

- ❑ Có thể thấy như bảo trì hoàn chỉnh mức cơ bản hay một sự lựa chọn để bảo trì
- ❑ Được biết như Software Re-engineering
- ❑ Nắm giữ hệ thống và chuyển đổi cấu trúc hay chuyển đổi thành ngôn ngữ mới
- ❑ Hệ thống cũ bắt đầu như đặc tả cho hệ thống mới
- ❑ Phương pháp chung được biết như vỏ bọc mà toàn hệ thống được thay bởi **vỏ bọc OO** và được xử lý như đối tượng lớn

Bảo trì ngăn ngừa (Preventative Maintenance)

Xem xét các điểm sau

- ☐ Chi phí có thể lớn hơn 20 tới 40 lần chi phí cho phát triển ban đầu dòng lệnh đó.
- ☐ Thiết kế lại cấu trúc với sử dụng các khái niệm thiết kế hiện tại có thể làm cho việc bảo hành tương lai dễ dàng hơn.
- ☐ Bởi vì khuôn mẫu phần mềm đã tồn tại, năng suất phát triển chương trình chắc sẽ cao hơn mức trung bình nhiều.
- ☐ Người sử dụng bây giờ đã làm quen với phần mềm. Vì vậy, các đòi hỏi mới và hướng thay đổi có thể tìm ra dễ dàng hơn nhiều.
- ☐ Các công cụ CASE dành cho reverse engineering và re-engineering sẽ thực hiện tự động một số phần của công việc.
- ☐ Một cấu hình phần mềm sẽ tồn tại trên sự hoàn thành của bảo trì phòng ngừa.

Sự lựa chọn để bảo trì

- ❑ Đôi khi, bảo trì không thoả mãn chính nó
- ❑ Tái cấu trúc không hoàn chỉnh tích hợp với bảo trì thích nghi
 - Dùng để cải tiến có thứ tự với mỗi phiên bản hệ thống
- ❑ Hoàn chỉnh sắp xếp lại hoặc xem xét toàn bộ code tồn tại
 - Dùng hệ thống thiên về bảo trì mức độ cao

Chọn lựa để bảo trì (2)

❑ Hoàn chỉnh tái thiết kế và viết lại

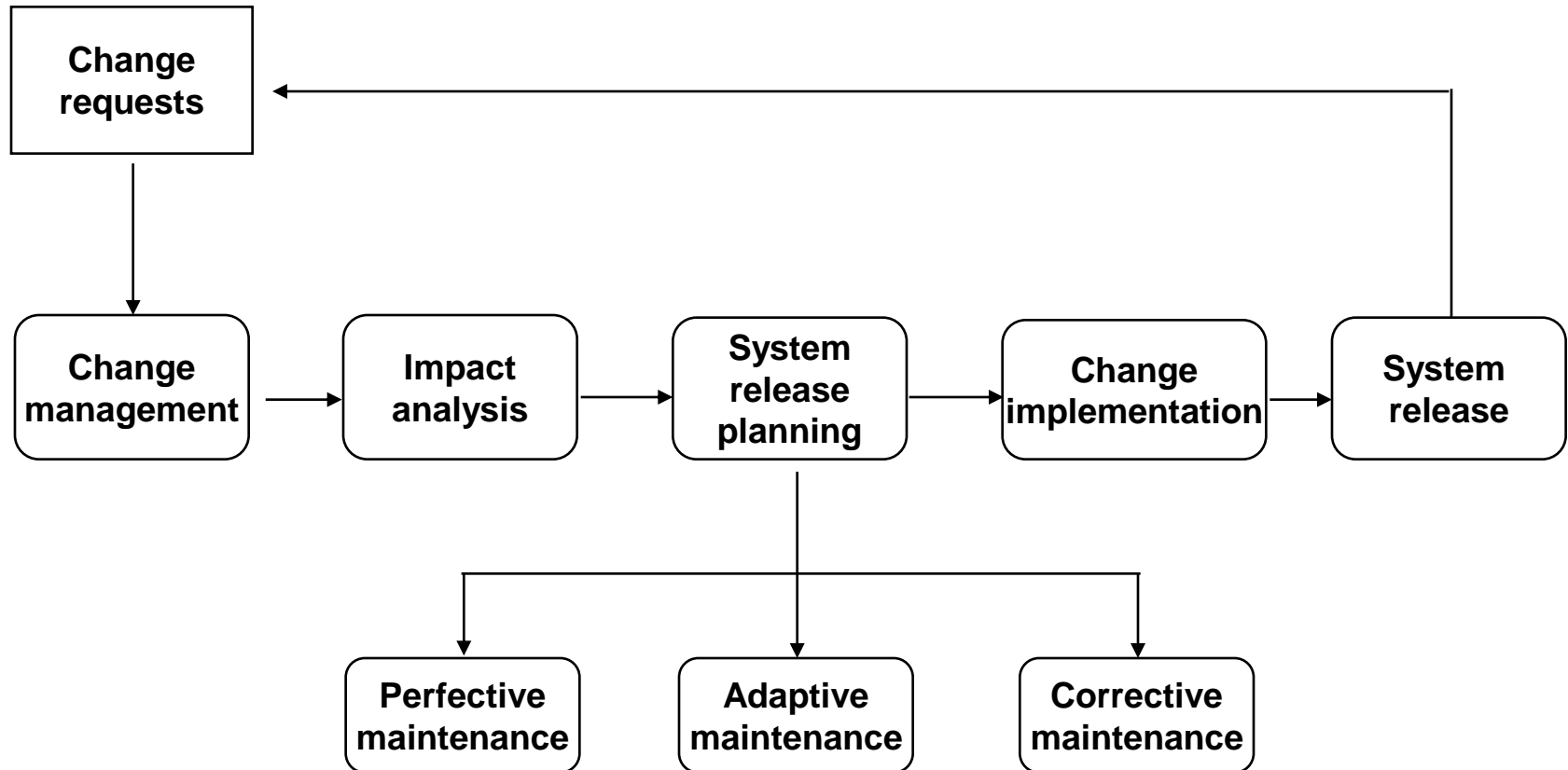
- Dùng khi nhiều hơn 20% chương trình phải thay đổi
- Dùng khi chương sẽ được nâng cấp cho công nghệ mới
- Không dùng khi thiết kế và chức năng của hệ thống không được biết

❑ Từ bỏ hệ thống và hoàn chỉnh tái phát triển

- Dùng khi chuyển qua công nghệ mới
- Dùng khi chi phí bảo trì phần mềm và phần cứng đạt đến chi phí của tái phát triển

Quy trình Bảo trì

Đây là quy trình lý tưởng, thường không đạt đến



Quy trình bảo trì (2)

❑ Quản lý sự thay đổi

- Nhận diện duy nhất, mô tả, lưu vết những trạng thái của tất cả yêu cầu

❑ Phân tích tác động

- Nhận diện tất cả hệ thống và sản phẩm tác động yêu cầu thay đổi
- Thực hiện ước tính tài nguyên cần thiết tác động thay đổi
- Phân tích lợi ích thay đổi

❑ Lập kế hoạch phiên bản (Release Planning)

- Thiết lập lịch biểu và nội dung của một phiên bản hệ thống
- Không muốn mỗi yêu cầu thay đổi được thực hiện khi chúng được xử lý

Quy trình Bảo trì phần mềm (3)

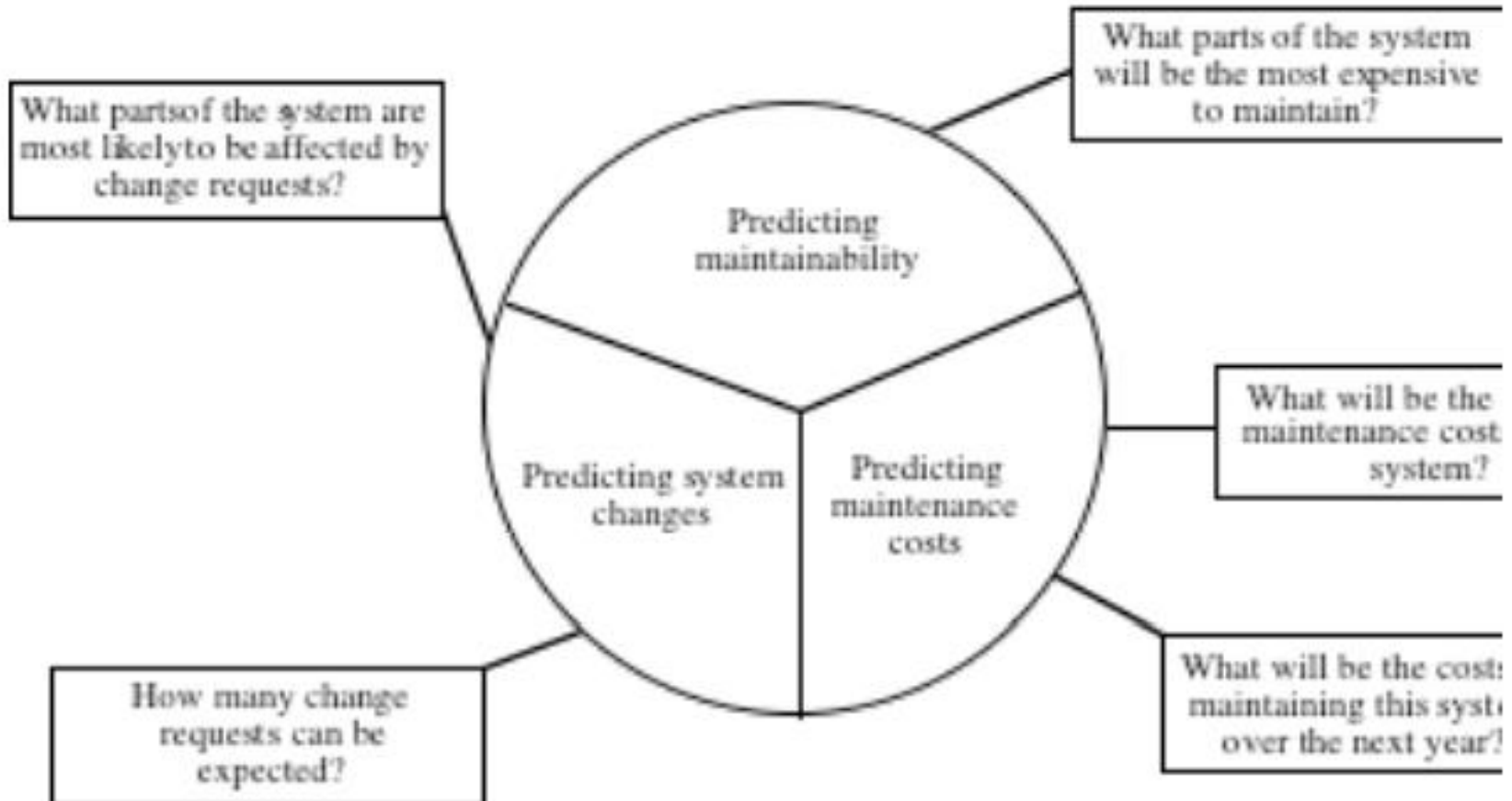
❑ Thực hiện sự thay đổi

- Thay đổi thiết kế
- Coding
- Kiểm thử Testing – phải thực hiện regression testing

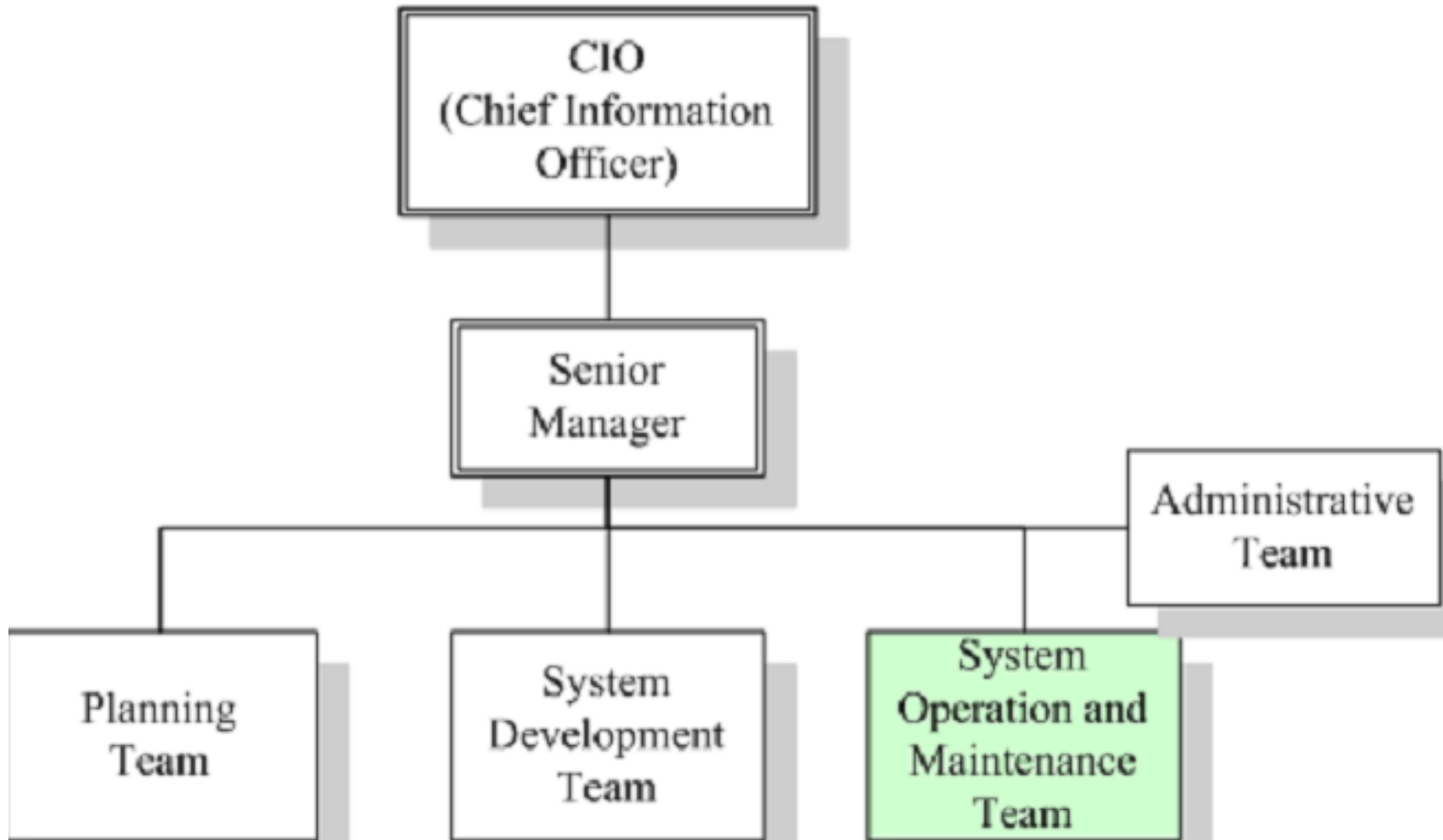
❑ Phiên bản hệ thống bao gồm

- Sơu liệu
- Phần mềm
- Huấn luyện
- Thay đổi phần cứng
- Chuyển đổi dữ liệu

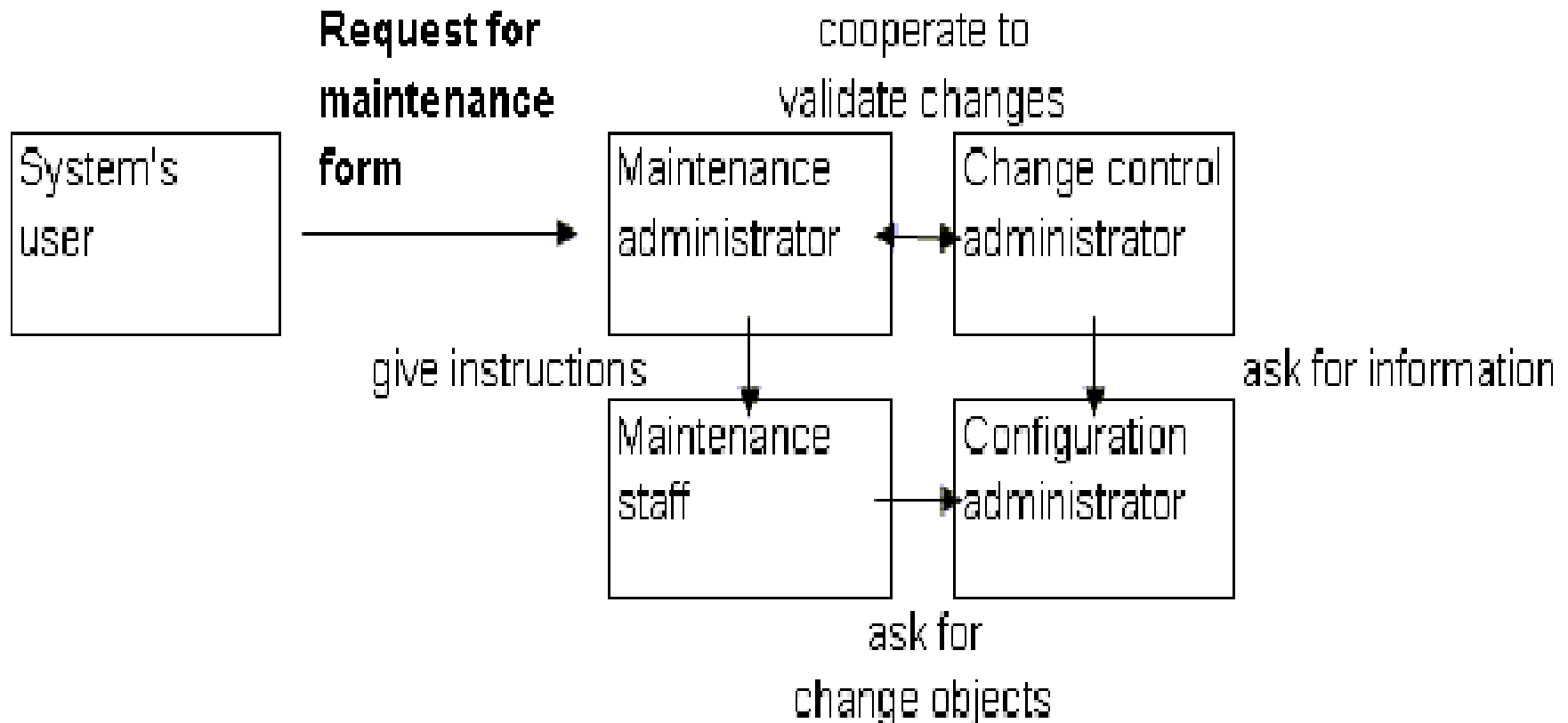
Dự đoán bảo trì (Maintenance Prediction)



Sơ đồ tổ chức bảo trì phần mềm



Vai trò và tổ chức bảo trì phần mềm



Mô hình quản lý sự thay đổi

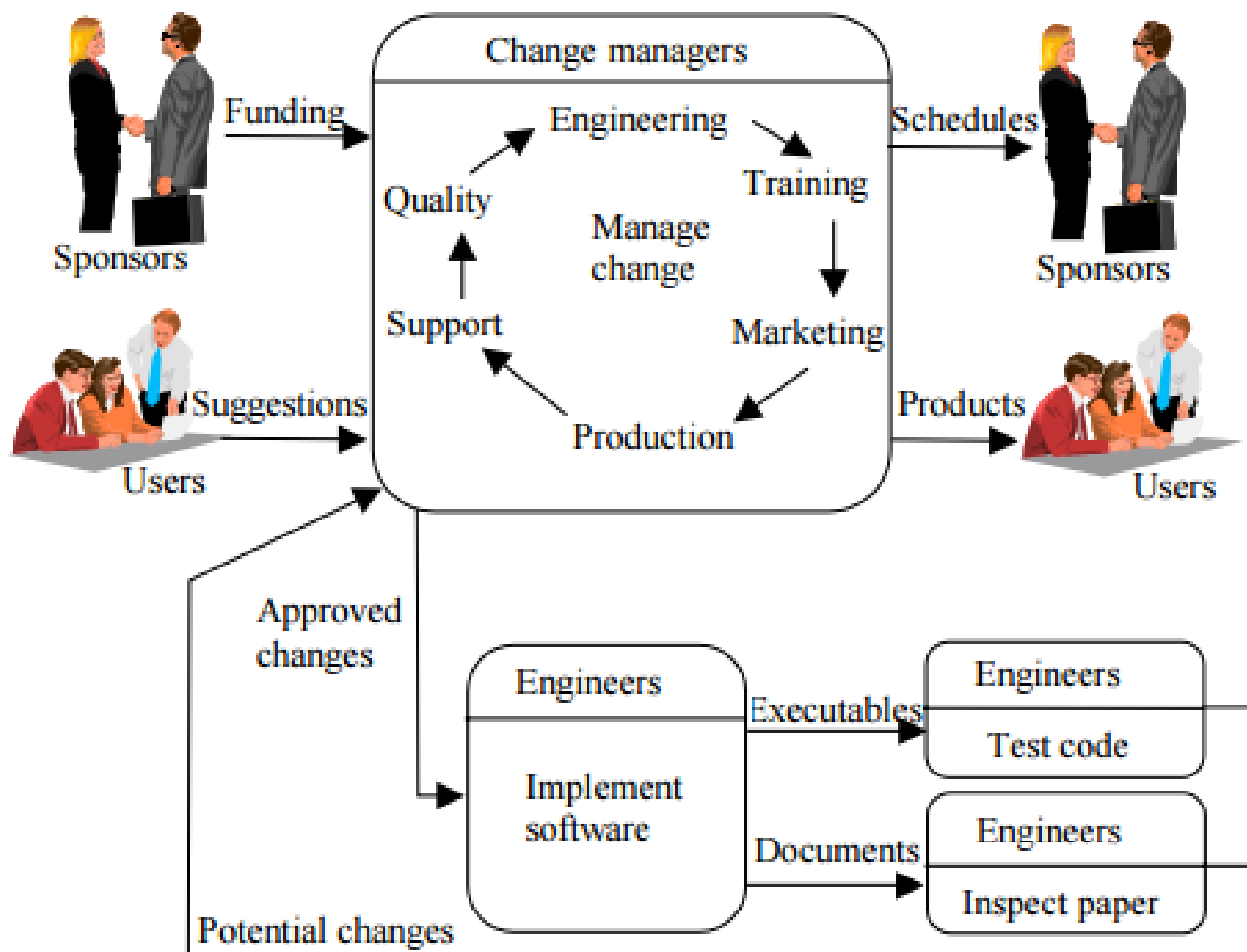


Figure 7. Olsen's change management model.

Mô hình quy trình sự thay đổi

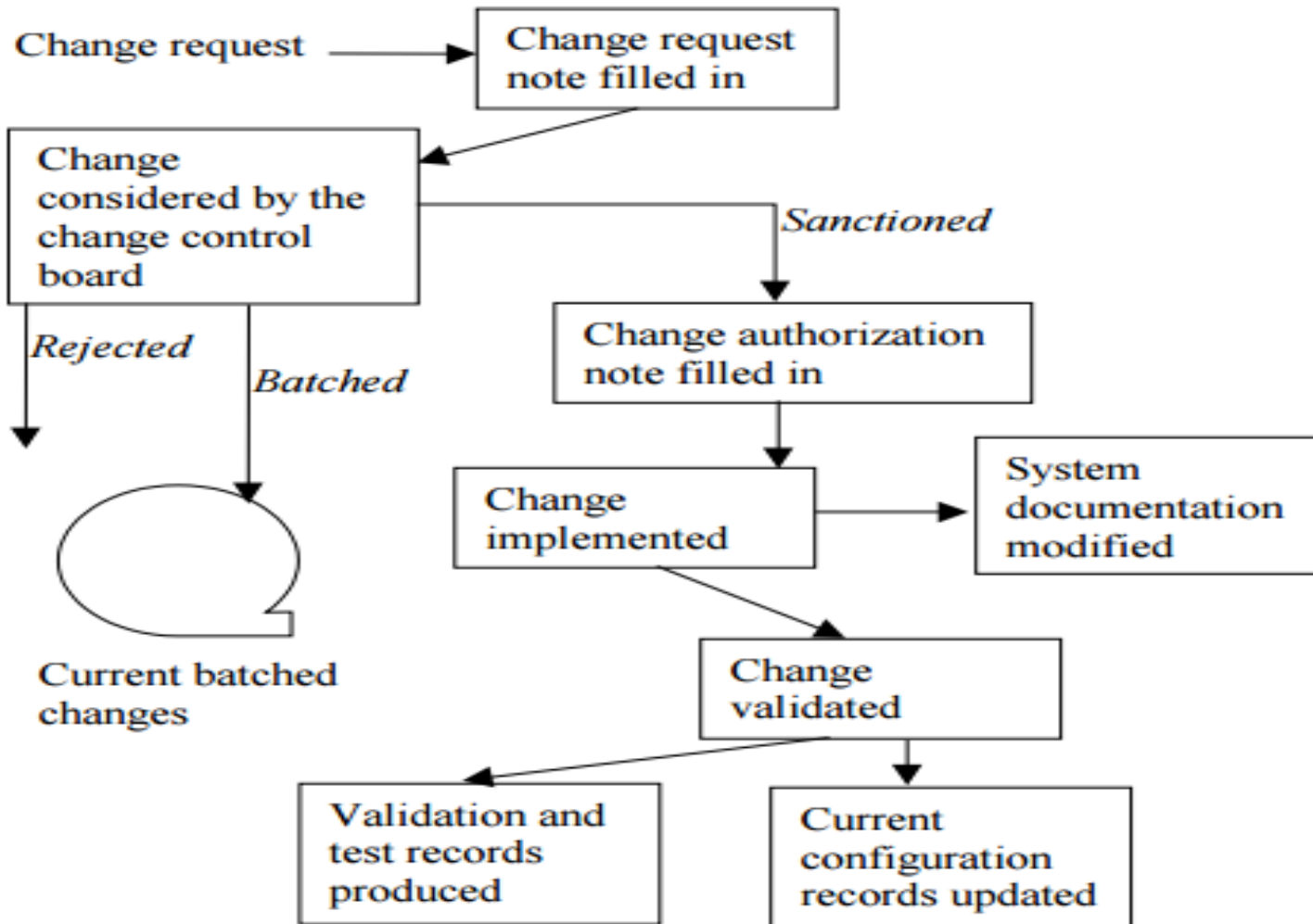


Figure 9. Ince's change process model (Ince 1994).

Mô hình hỗ trợ quy trình bảo trì ở một cty

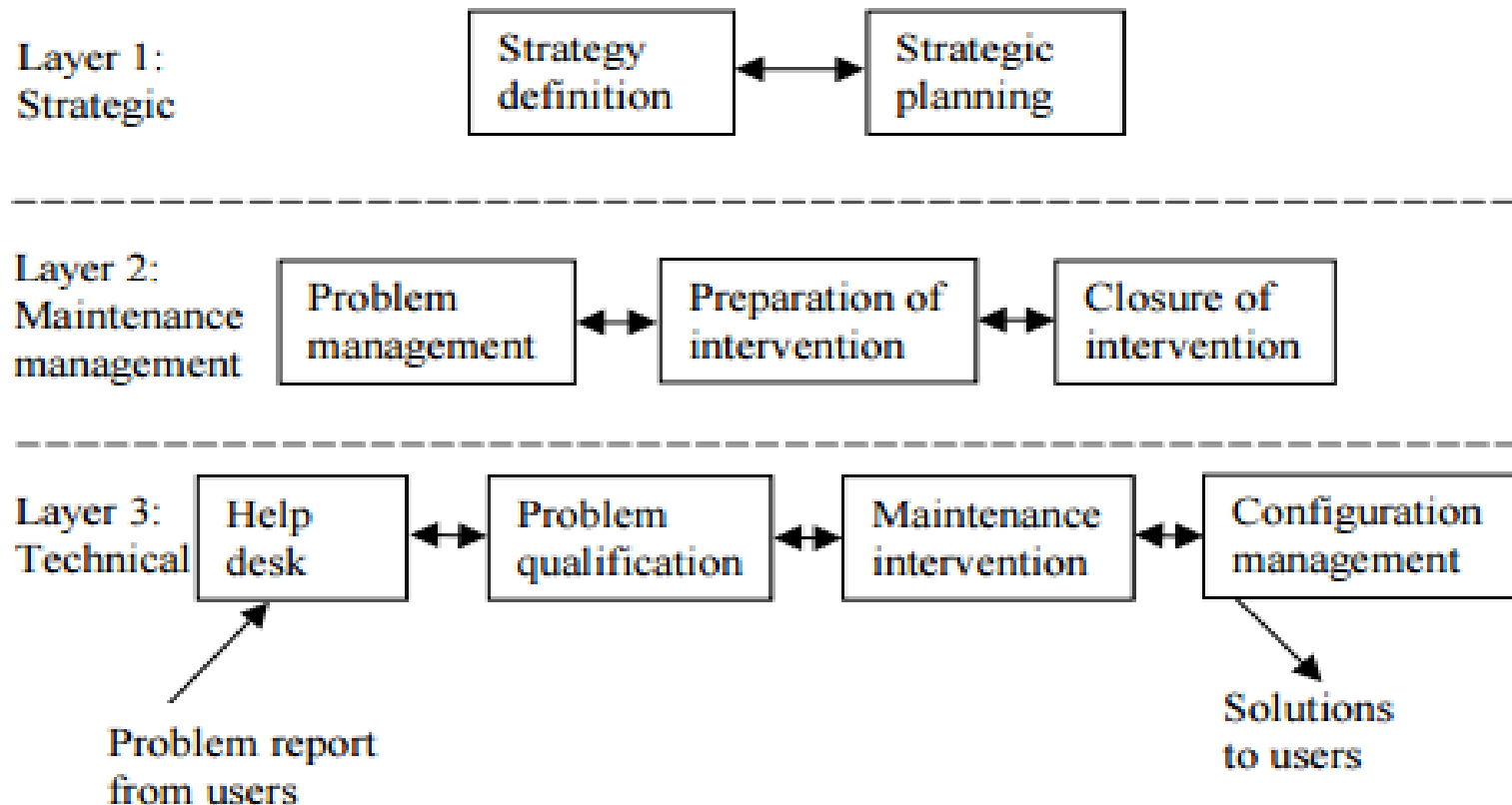


Figure 10. The AMES process model.

Mô hình quản lý thay đổi Spiral-like

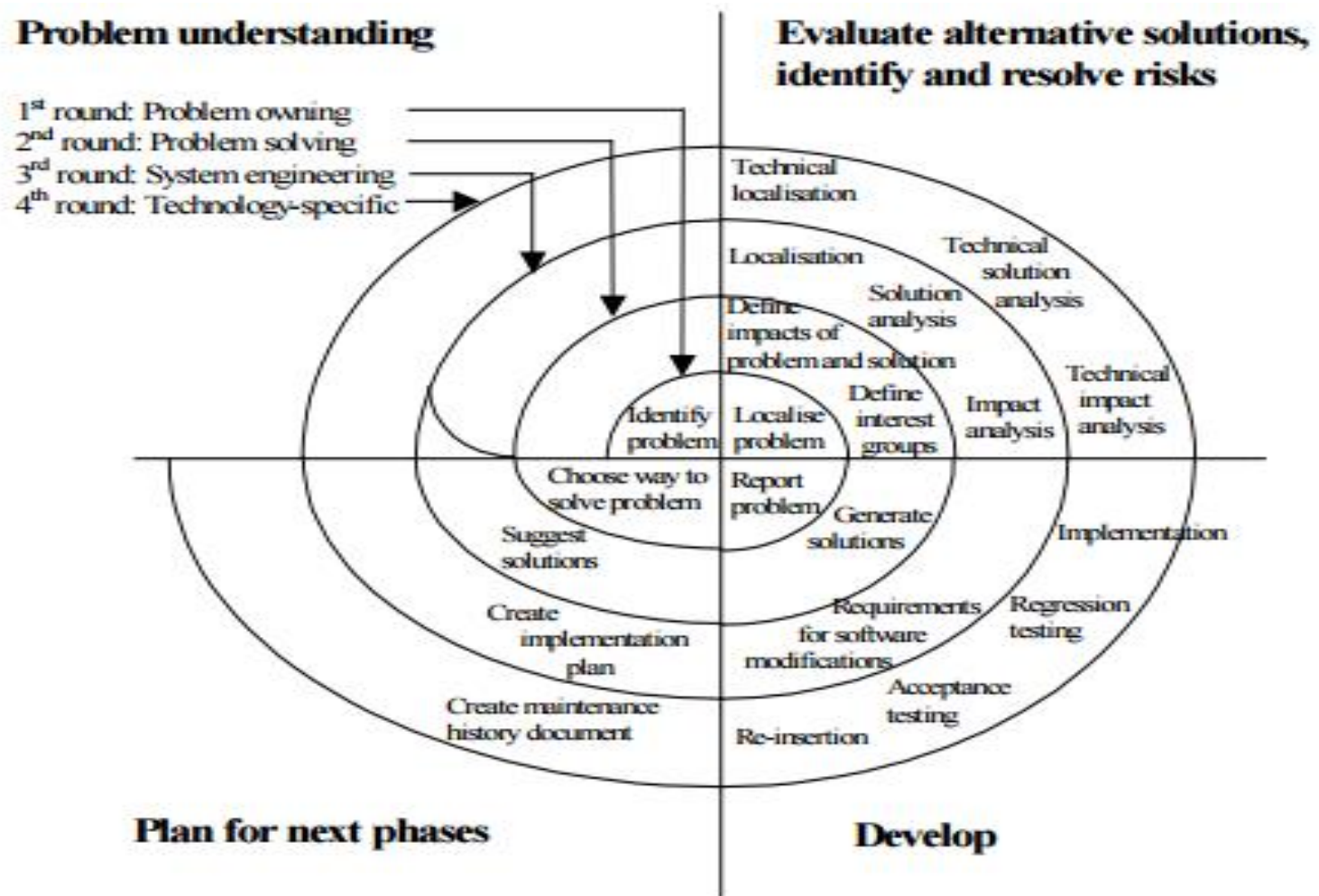


Figure 11. The spiral-like change management process.

OnGoing Support

❑ Truyền thông hiệu quả

- Thiết lập mối liên hệ tốt với khách hàng
- Hiểu rõ nhu cầu khách hàng
- Tránh ra quyết định trái ngược yêu cầu khách hàng

❑ Huấn luyện end-user

- Hỗ trợ người dùng dễ dàng hiểu, ex: phone online queries

❑ Cung cấp thông tin kinh doanh

- Thông tin chính xác để thể hỗ trợ ra quyết định chiến lược

❑ **Bài tập:** Thảo luận cho tình huống của nhóm để hoạt động OnGoing support hiệu quả, đề xuất công cụ giải pháp cụ thể hỗ trợ tối đa cho nhóm customer

Một số hiệu ứng lè (Ripple effect)

Sửa đổi phần mềm là công việc “nguy hiểm”, ta thường gặp ba loại chính của hiệu ứng lè

- ❑ **Hiệu ứng lè của việc thay đổi mã nguồn**
- ❑ **Hiệu ứng lè của việc thay đổi dữ liệu**
- ❑ **Hiệu ứng lè của việc thay đổi tài liệu**
- ❑ **Bài tập:** Thảo luận cho ví dụ minh họa ngữ cảnh các hiệu ứng lè như trên khi sửa lỗi phần mềm

Hiệu ứng lẻ của việc thay đổi mã nguồn

Việc sửa lỗi luôn dẫn đến các vấn đề phức tạp. Tập hợp các thay đổi sau có thể gây ra nhiều lỗi hơn

- ☐ Một chương trình con bị xóa hay thay đổi.
- ☐ Một dòng nhãn bị xóa hay thay đổi.
- ☐ Một biến bị xóa hay thay đổi.
- ☐ Các thay đổi để tăng khả năng thực hiện.
- ☐ Việc mở và đóng file bị thay đổi.
- ☐ Các phép toán logic bị thay đổi.
- ☐ Việc thay đổi thiết kế chuyển thành các thay đổi lớn về chương trình.
- ☐ Các thay đổi ảnh hưởng đến việc chạy thử các trường hợp biên.

Hiệu ứng lề của việc thay đổi dữ liệu

- ☐ Định nghĩa lại các hằng số cục bộ và hằng số địa phương.
- ☐ Định nghĩa lại cấu trúc bản ghi hay cấu trúc file.
- ☐ Tăng hoặc giảm kích thước một mảng.
- ☐ Thay đổi dữ liệu tổng thể.
- ☐ Định nghĩa lại các cờ điều khiển và các con trỏ.
- ☐ Xếp lại các tham số vào ra hay tham số của chương trình con.

Hạn chế: bằng tài liệu thiết kế mô tả cấu trúc dữ liệu và cung cấp một lời chỉ dẫn tham khảo đến từng phần tử dữ liệu, các bản ghi, các file và các cấu trúc khác của các module phần mềm

Hiệu ứng lè của việc thay đổi tài liệu

- ❑ thay đổi chương trình nguồn mà không thay đổi tài liệu thiết kế và tài liệu hướng dẫn sử dụng
- ❑ Hiệu ứng lè xảy ra trong các lần bảo trì sau đó, khi việc nghiên cứu không kỹ các tài liệu kỹ thuật, dẫn tới sự đánh giá sai về các đặc tính của phần mềm. Đối với người sử dụng, phần mềm tốt chỉ khi có tài liệu hướng dẫn sử dụng chúng.
- ❑ không được thay đổi thiết kế của phần mềm hoặc mã chương trình, mà chỉ cần chỉ ra sự thiếu rõ ràng trong tài liệu của người sử dụng

Các luật của Tính tiến hoá chương trình

❑ Continuing Change

Source: Adapted from Lehman 1980, pp1061-1063. See also, van Vliet, 1999, Pp59-62

- Any software *that reflects some external reality* undergoes continual change or becomes progressively less useful
 - ✓ The change process continues until it is judged more cost effective to replace the system entirely

❑ Increasing Complexity

- As software evolves, its *complexity* increases...
 - ✓ ...unless steps are taken to control it.

❑ Fundamental Law of Program Evolution

- Tiến hoá của phần mềm là qui định tự nó với hướng có thể xác định và không thay đổi theo thống kê

❑ Conservation of Organizational Stability

- Trong suốt hoạt động thực sự của hệ thống phần mềm, đầu ra công việc của một dự án phát triển là gần không đổi (xem như tài nguyên)

❑ Conservation of Familiarity

- Trong suốt hoạt động thực sự của một chương trình số lượng thay đổi trong những phiên bản kế tiếp gần không đổi

Lehman's laws

Law	Description
Continuing change	A program that is used in a real-world environment necessarily must change or become progressively less useful in that environment.
Increasing complexity	As an evolving program changes, its structure tends to become more complex. Extra resources must be devoted to preserving and simplifying the structure.
Large program evolution	Program evolution is a self-regulating process. System attributes such as size, time between releases and the number of reported errors is approximately invariant for each system release.
Organisational stability	Over a program's lifetime, its rate of development is approximately constant and independent of the resources devoted to system development.
Conservation of familiarity	Over the lifetime of a system, the incremental change in each release is approximately constant.
Continuing growth	The functionality offered by systems has to continually increase to maintain user satisfaction.
Declining quality	The quality of systems will appear to be declining unless they are adapted to changes in their operational environment.
Feedback system	Evolution processes incorporate multi-agent, multi-loop feedback systems and you have to treat them as feedback systems to achieve significant product improvement.

Khả năng áp dụng của các luật Lehman

- ❑ Luật Lehman dường như được áp dụng với hệ thống lớn được phát triển bởi tổ chức lớn.
- ❑ không rõ ràng các luật này được thay đổi như thế nào đối với
 - Những tổ chức nhỏ
 - Hệ thống độ lớn trung bình
 -

2.2 MỐI LIÊN HỆ KINH TẾ CỦA VIỆC CẬP NHẬT PHẦN MỀM

- ☐ Giới hạn đối với sự thay đổi
- ☐ Hạn chế tài nguyên
- ☐ Chất lượng của hệ thống tồn tại
- ☐ Chiến lược tổ chức
- ☐ Tính trì trệ (không thay đổi)

Chi phí bảo trì

❑ Một nghiên cứu đưa ra

- Định nghĩa yêu cầu 3%
- Thiết kế sơ bộ 3%
- Thiết kế mức chi tiết 5%
- Thực thi 7%
- Kiểm thử 15%
- Bảo trì 67%

❑ Nghiên cứu khác đưa ra ít nhất 50% nỗ lực chi cho bảo trì

❑ Nghiên cứu khá tìm thấy khoảng giữa 65% và 75% cho bảo trì

❑ Những hệ thống nhúng thời gian thực, chi phí bảo chỉ chiếm gấp 4 lần chi phí phát triển

Tại sao bảo trì tốn kém ?

- ☐ Hầu hết phần mềm có từ 10 and 15 năm
- ☐ Nhiều phần mềm chỉ tuổi nó được tạo bởi kích thước chương trình và không gian lưu trữ là những yếu tố quan trọng hơn
- ☐ Điều này dẫn đến không thay đổi thiết kế, code, và sơ liệu
- ☐ Bảo trì được thực hiện bởi đội ngũ không có kinh nghiệm và không quen với ứng dụng
- ☐ Người phát triển không thích bảo trì
- ☐ Thay đổi thường gây ra lỗi mới trong hệ thống
- ☐ Thay đổi hướng làm mạnh mún cấu trúc chương trình
- ☐ Thay đổi thường không được ghi sơ liệu

Yếu tố tác động đến chi phí bảo trì

❑ Tính độc lập module

- Khả năng thay đổi một phần hệ thống
- Thuận lợi tiềm năng của OO

❑ Ngôn ngữ lập trình

- Mức độ ngôn ngữ lập trình càng cao, càng bảo trì dễ hơn
- C – ngôn ngữ chữ viết :-)

❑ Kiểu chương trình

- Cách thức một chương trình được viết

❑ Đánh giá và kiểm thử chương trình

- Càng nhiều thời gian và nỗ lực chi cho đánh giá thiết kế và chương trình, lỗi càng ít và nhu cầu bảo trì hiệu chỉnh càng ít hơn

Yếu tố tác động đến chi phí bảo trì (2)

❑ Chất lượng sưu liệu chương trình

- Sưu liệu càng tốt, việc bảo trì càng dễ dàng

❑ Kỹ thuật quản lý cấu hình

- Lưu vết tất cả tài liệu hệ thống và đảm bảo chúng phù hợp thì chi phí chính của bảo trì, như vậy công cụ quản lý cấu hình (CM tools) và thực tế giảm chi phí này

❑ Phạm vi ứng dụng

- Càng ít hiểu phạm vi được hiểu kỹ, the less well-understood the domain, nhu cầu có thể xảy ra càng lớn cho bảo trì thích nghi như người dùng và người phát triển bắt đầu hiểu phạm vi

❑ Ổn định đội ngũ

- Chi phí bảo trì giảm nếu người phát triển phải bảo trì chính hệ thống của họ, thường qua chu kỳ sống của hệ thống
-

Yếu tố tác động đến chi phí bảo trì (3)

❑ Tuổi hệ thống

- Hệ thống càng cũ, càng nhiều mức độ manh mún và khó bảo trì
- Lôi cuốn đội ngũ người biết hệ thống cũ ngôn ngữ, DB, vận hành trở nên khó hơn và nhiều kinh nghiệm

❑ Phụ thuộc hệ thống và môi trường ngoài

- Sự phụ thuộc càng cao, càng nhiều bảo trì thích nghi được cần

❑ Độ ổn định phần cứng

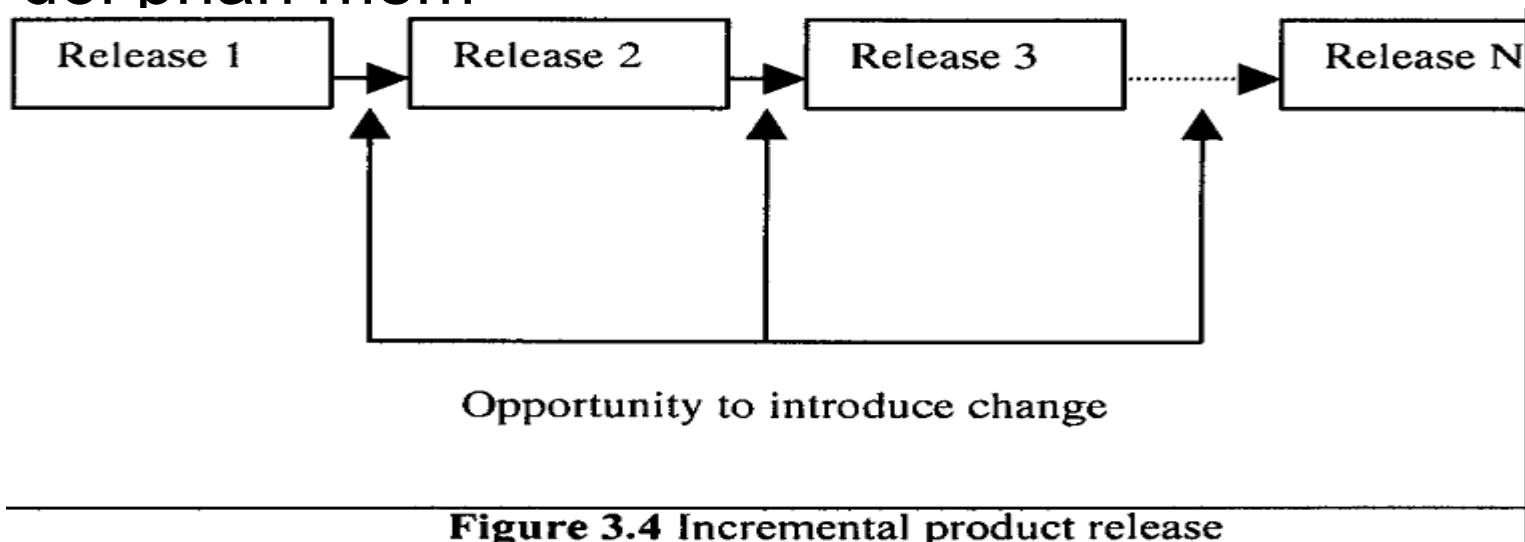
- Nếu platform phần cứng sẽ không thay đổi qua thời gian hệ thống, bảo trì cho nguyên nhân này sẽ không cần

Khác nhau giữa Bảo trì và Phát triển mới

- ❑ Ràng buộc của hệ thống tồn tại
 - Thay đổi phải phù hợp hay tương thích với kiến trúc tồn tại, thiết kế, ràng buộc mã nguồn
- ❑ Khung thời gian ngắn hơn
 - Phát triển khoảng 6 tháng trở lên
 - Bảo trì tính giờ hay ngày cho đến 6 tháng
- ❑ Tính sẵn sàng dữ liệu kiểm thử
 - Phát triển tạo tất cả dữ liệu từ hỗn tạp
 - Bảo trì dùng dữ liệu kiểm tra tồn tại với regression testing, tạo dữ liệu mới từ sự thay đổi

Bài tập & Thảo luận (1)

- ❑ Bài tập 3.1 Mô tả những thay đổi khác nhau mà sản phẩm phần mềm trải qua và chỉ ra lý do cơ bản cho mỗi loại
- ❑ Bài tập 3.2 : Giải thích tại sao cho là quan trọng để phân biệt giữa những loại khác nhau của sự thay đổi phần mềm



Bài tập & Thảo luận (2)

- ❑ **Bài tập 3.3** Ongoing support không được cho là cần thiết dẫn đến sự thay đổi của chương trình như vậy nó nên xem là một phần của bảo trì không?. Ý kiến của bạn là gì.
- ❑ **Bài tập 4.1:** Chọn gói phần mềm (software package) đã biết liên quan nhiều version. Liệt kê những thay đổi đã được thực hiện trong những version khác nhau. Có những Rào cản gì trong thực hiện những chức năng cụ thể này trong những version.

2.3 GIẢI PHÁP TIỀM NĂNG ĐỐI VỚI VẤN ĐỀ BẢO TRÌ

❑ Tái cấp phát Ngân sách và Nỗ lực (Effort)

- Ít tài nguyên để phát triển cho hệ thống khó bảo trì và khó, ngược lại đầu tư nhiều cho phát triển, đặc tả và thiết kế với hệ thống có thể bảo trì
- Sử dụng nhiều đặc tả yêu cầu thiết kế trước đã **phê duyệt**, kỹ thuật thiết kế, công cụ, chất lượng đảm bảo tiêu chuẩn ISO ... làm hệ thống có thể **bảo trì** hơn

❑ Hoàn chỉnh thay thế hệ thống

- Ràng buộc kinh tế
- Lỗi thường trú trong hệ thống mới
- Cơ sở dữ liệu của thông tin

❑ Bảo trì hệ thống tồn tại

Những vấn đề đối mặt của người bảo trì

❑ 5 vấn đề hàng đầu:

*Source: Adapted Pfleeger 1998, p423-424.
See also, van Vliet, 1999, pp464-467*

- (Kém) Chất lượng sưu liệu
- Nhu cầu người dùng cho việc cải tiến và mở rộng
- Nhu cầu đối đầu thời gian người bảo trì
- Khó khăn trong trong buổi họp cam kết lịch trình
- Doanh thu trong tổ chức người dùng

❑ Sự hiểu biết hạn chế

- 47% nỗ lực bảo trì phần mềm dành cho hiểu phần mềm
 - ✓ Thí dụ: nếu một hệ thống có m thành phần và chúng ta cần thay đổi k trong chúng ...
 - ✓ ... Có $k^*(m-k) + k*(k-1)/2$ giao diện kiểm tra tác động
- Cùng , >50% nỗ lực đóng góp cho sự thiếu hiểu biết của người dùng
 - ✓ I.e. báo cáo không hoàn chỉnh và sai lầm của lỗi và cải tiến

❑ Tinh thần Thấp

- Bảo trì phần mềm được xem xét như ít thích thú hơn việc phát triển

Cách tiếp cận bảo trì

❑ Triết lý bảo trì

Source: van Vliet, 1999, pp473-475

- “throw-it-over-the-wall” – người nào khác chịu trách nhiệm cho bảo trì
 - ✓ Đầu tư kiến thức và kinh nghiệm là uổng phí
 - ✓ Bảo trì trở thành thách thức reverse engineering
- “mission orientation” – nhóm phát triển thực hiện cam kết giới hạn để bảo trì phần mềm

❑ Mô hình quy trình bảo trì của Basili:

- Quick-fix model
 - ✓ Thay đổi làm ở mức lập trình (code) dễ dàng có thể
 - ✓ Phân rã (manh mún) nhanh cấu trúc của phần mềm
- Iterative enhancement model
 - ✓ Thay đổi thực hiện dựa trên phân tích hệ thống tồn tại
 - ✓ Cố gắng kiểm soát độ phức tạp và duy trì thiết kết tốt
- Full-reuse model
 - ✓ Bắt đầu bằng những yêu cầu hệ thống mới, tái sử dụng nhiều như có thể
 - ✓ Cần tăng trưởng văn hoá dùng lại để thành công

Làm trẻ lại phần mềm (Software Rejuvenation)

❑ Redocumentation

Source: van Vliet, 1999, Pp455-457

- Tạo hay xem lại những thể hiện sự thay đổi phần mềm
 - ✓ Tại cùng mức độ của trừu tượng hoá
- Phát sinh :
 - ✓ Bảng giao diện dữ liệu, đồ hình gọi hàm, thành phần/biến qua bảng tham chiếu etc.

❑ Restructuring

- Chuyển dịch phần code của hệ thống mà không thay đổi hành vi

❑ Reverse Engineering

- Phân tích hệ thống để chính xác thông tin về hành vi hay cấu trúc
 - ✓ Cũng khôi phục thiết kế - Tạo lại trừu tượng thiết kế từ code, tài liệu, và kiến thức phạm vi
- Phát sinh:
 - ✓ Sơ đồ cấu trúc, lược đồ quan hệ thực thể, DFD, mô hình yêu cầu

❑ Reengineering

- Kiểm tra và thông báo hệ thống khôi phục nó trong hình thức khác
- Cũng cần biết như nâng cấp, phân loại lại

Reuse

❑ Dùng lại phần mềm để cắt giảm chi phí Source: van Vliet, 1999, Chapter 17

- Phát triển phần mềm tốn kém, vì vậy dùng lại cho những hệ thống liên quan
 - ✓ Hướng tiếp cận thành công tập trung sử dụng lại tri thức và kinh nghiệm hơn sản phẩm phần mềm
 - ✓ Tính kinh tế việc dùng lại là phức tạp khi nó tốn nhiều hơn để phát triển *reusable* software

❑ Thư viện của thành phần có thể sử dụng lại

- Thư viện phạm vi cụ thể (e.g. Math libraries)
- Thư viện phát triển chương trình (e.g. Java AWT, C libraries)

❑ Domain Engineering

- Phân chia phát triển phần mềm thành 2 phần :
 - ✓ Phân tích phạm vi – nhận diện thành phần dùng lại có chung đặc điểm cho một phạm vi vấn đề
 - ✓ Phát triển ứng dụng – dùng thành phần phạm vi cho ứng dụng cụ thể.

❑ Họ phần mềm

- Nhiều công ty đề nghị dãy các hệ thống phần mềm liên quan
 - ✓ Chọn kiến trúc ổn định cho họ phần mềm
 - ✓ Nhận diện sự đa dạng cho những thành viên khác nhau trong họ
- Thể hiện quyết định chiến lược kinh doanh về phần mềm gì để phát triển

Thúc đẩy độ ngũ bảo trì

- ❑ Thường xem xét đến dead-end trong tổ chức cũng như sự chán nản
- ❑ Quyết định đến sự thành công của tổ chức
- ❑ Chiến lược có thể
 - Cập mục tiêu phần mềm và mục đích của tổ chức
 - Cập Phần thưởng bảo trì phần mềm với thực hiện tổ chức
 - Tích hợp nhân sự bảo trì phần mềm với nhóm vận hành
 - Tạo biện pháp, hoàn chỉnh/ngăn ngừa, ngân sách bảo trì cho phép nhóm bảo trì quyết định khi re-engineer hệ thống
 - Lỗi kéo độ ngũ bảo trì sớm trong qui trình phần mềm trong suốt chuẩn bị qui chuẩn, review, và chuẩn bị kiểm thử

Tài liệu tham khảo

❑ van Vliet, H. “Software Engineering: Principles and Practice (2nd Edition)” Wiley, 1999.

○Chapter 14 is a very good introduction to the problems and approaches to software maintenance. Chapter 17 covers software reuse in far more detail than we'll go into on this course.

❑Lehman, M.M. “Programs, Life Cycles, and Laws of Software Evolution”. Proceedings of the IEEE, vol 68, no 9, 1980.

○Lehman was one of the first to recognise that software evolution is a fact of life. His experience with a number of large systems led him to formulate his laws of evolution. This paper is included in the course readings. It is widely cited.

❑Pfleeger, S. L. “Software Engineering: Theory and Practice” Prentice Hall, 1998.

○Pfleeger's chapter 10 provides some additional data on the costs of maintenance.

Yêu cầu thực hiện tuần tiếp theo

- ☐ Nộp báo cáo các bài tập làm trên lớp (*)
- ☐ Tìm hiểu và minh họa các công cụ (tools) quản lý sự thay đổi (*)
- ☐ *Chuẩn bị các thuyết minh và chương trình để họp với nhóm khách hàng -> ghi nhận yêu cầu, thay đổi (xem template **Report1.pdf** và **Report2.pdf**, **SMP_documents.rar***
- ☐ Nhóm khách hàng (**customer group**): bắt cặp ngược theo thứ tự nhóm đã được cho ?, và ngược lại
- ☐ Hai nhóm sẽ có đại diện để trao đổi ghi nhận thông tin, trao đổi thông tin với nhau thông qua **group link** của nhau.