PHÁT TRIỂN VẬN HÀNH BẢO TRÌ PHẦN MỀM

ThS. NGUYỄN THỊ THANH TRÚC

Chương 5:
KHẢ NĂNG SỬ DỤNG LẠI VÀ KiỂM THỬ

5.1 **TÍNH DÙNG LẠI VÀ KHẢ NĂNG DÙNG LẠI**

5.2 **KiỂM THỬ**

KHẢ NĂNG SỬ DỤNG LẠI VÀ KiỂM THỬ

- **5.1 TÍNH DÙNG LẠI VÀ KHẢ NĂNG DÙNG LẠI**
  - Giới thiệu
  - Định nghĩa
  - Mục đích của việc sử dụng lại
  - Mục tiêu và lợi ích của việc dùng lại
  - Hướng tiếp cận của dùng lại
  - Phân tích phạm vi
  - Công nghệ cấu phần
  - Mô hình qui trình dùng lại
  - Các yếu tố tác động lên việc sử dụng lại
- **5.2 KiỂM THỬ**
  - Giới thiệu
  - Định nghĩa
  - Tại sao kiểm thử phần mềm
  - Công việc của người kiểm thử phần mềm
  - Kiểm thử gì và như thế nào
  - Phân loại kiểm thử
  - Thẩm định và đánh giá
  - Kế hoạch kiểm thử

5.1 TÍNH DÙNG LẠI VÀ KHẢ NĂNG DÙNG LẠI

- Giới thiệu
- Định nghĩa
- Mục đích của việc sử dụng lại
- Mục tiêu và lợi ích của việc dùng lại
- Hướng tiếp cận của dùng lại
- Phân tích phạm vi
- Công nghệ cấu phần
- Mô hình qui trình dùng lại
- Các yếu tố tác động lên việc sử dụng lại

Mục đích của việc sử dụng lại

- *để tăng năng suất:*
- *để nâng cao chất lượng:*
- *dễ dàng chuyển dịch code:*
- *Giảm thời gian bảo trì và nỗ lực thực hiện:*
- *cải thiện khả năng bảo trì*

Bài tập

- **Exercise 8.3** Nêu lý do tại sao việc sử dụng phần mềm dùng lại thì quan trọng thay vì viết chúng từ phần không chọn lọc (scratch).
- **Exercise 8.4** Những lợi ích có thể dẫn xuất từ phần mềm sử dụng lại?

Những tiếp cận của Reuse

- ***Composition-Based Reuse***
  - *Black-box reuse:*
  - *White-box reuse:*
- ***Generation-Based Reuse***
  - **Application Generator Systems**
  - **Transformation-Based Systems**
  - **Evaluation of the Generator-Based Systems**

Phân tích phạm vi

- thành phần theo chiều ngang và chiều dọc thành phần tái sử dụng

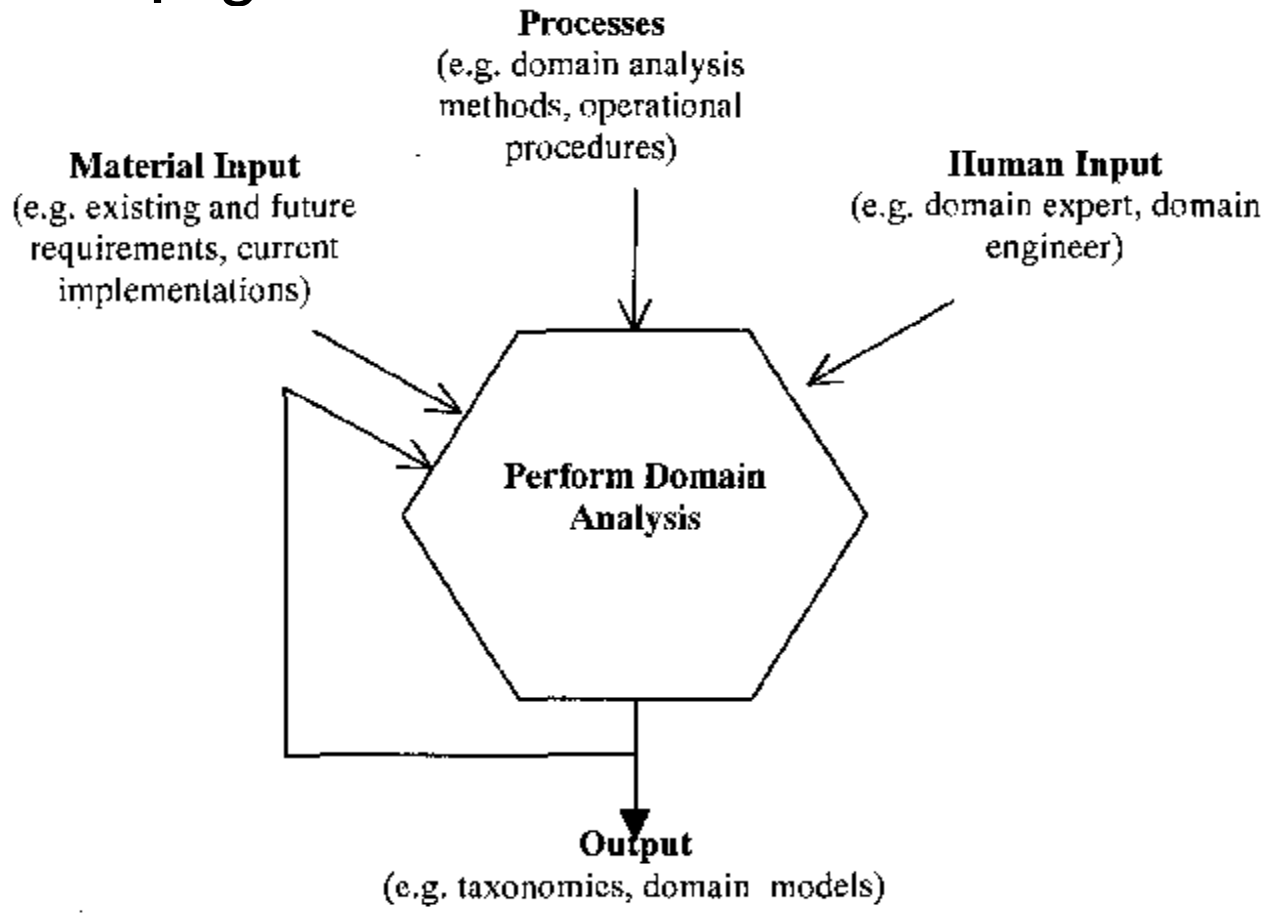


Figure 8.2 The inputs and outputs of domain analysis

Công nghệ cấu phần (Components engineering)

- ***Thiết kế dành cho Reuse***
  - **Đặc trưng của thành phần có khả năng dùng lại**
  - **Những vấn đề với thư viện dùng lại**
- *Reverse Engineering*
- ***Qui trình dựa trên cấu phần***

Đặc trưng của thành phần có khả năng tái sử dụng

- Tổng quát
- Cohesion versus coupling:
- Sự tương tác
- Tính đồng nhất và tiêu chuẩn hóa
- Tính trừu tượng Data và control
- Khả năng tương tác:

Những vấn đề với thư viện dùng lại

- *Các chi tiết và kích thước tiến thoái lưỡng nan :*
- *Vấn đề tra cứu*
- *Vấn đề phân lớp:*
- *Các vấn đề về đặc tả và tính linh hoạt :*

Bài tập

- Exercise 8.5 So sánh những tiếp cận khác nhau của reuse, cho ví dụ những hệ thống có thể chứa với mỗi tiếp cận này.

Mô hình qui trình dùng lại

- Đây là một kết quả của nhiều yếu tố [133]:
- Phần mềm tái sử dụng vốn không phải là từ trên xuống, như là một số các mô hình vòng đời (ví dụ, waterfall model).
- Trong sử dụng lại phần mềm, các nhà phát triển hoặc duy trì có một cái nhìn mở rộng vượt ra ngoài các dự án đơn lẻ hoặc các hệ thống.
- Tái sử dụng liên quan đến việc khai thác phổ biến ở nhiều cấp độ trừu tượng bên cạnh đó dễ dàng nắm bắt trong mã.
- Tái sử dụng phụ thuộc, đến một mức độ lớn vào khả năng phân tích các lĩnh vực cụ thể để trích xuất các thành phần tối đa có thể dùng lại. phương pháp có cấu trúc được thiết kế cho các mô hình vòng đời từ trên xuống, tuy nhiên, hiếm khi cung cấp các kỹ thuật cụ thể để phân tích lĩnh vực.

generic reuse model /Reusability Model

Các bước của mô hình dùng lại tóm tắt như sau:

- **Bước 1**: Bước này gồm hiểu vấn đề được giải quyết và sau nhận diện cấu trúc giải pháp dựa trên thành phần đã được định nghĩa trước.
- **Bước 2**: Cấu trúc của giải pháp được cấu hình lại để tối ưu tính dùng lại thiết yếu cho hiện tại và giai đoạn sau.
- **Bước 3**: Tác vụ chính ở giai đoạn này là chuẩn bị những thành phần dùng lại đồng nhất trong cấu trúc giải pháp sẳn sàng cho tích hợp.
- **Bước 4**: Mục đích chính tại giai đoạn này là tích hợp thành phần hoàn chỉnh trong sản phẩm được yêu cầu cho giai đoạn tiếp theo của chu trình sống phần mềm.
- **Bước 5**: Trong bước này, kinh nghiệm từ những bước trước được dùng để đánh giá khía cạnh khả dụng của những thành phần mà cần được phát triển cho vấn đề con mà không có thành phần khả dùng tồn tại

Mô hình qui trình dùng lại

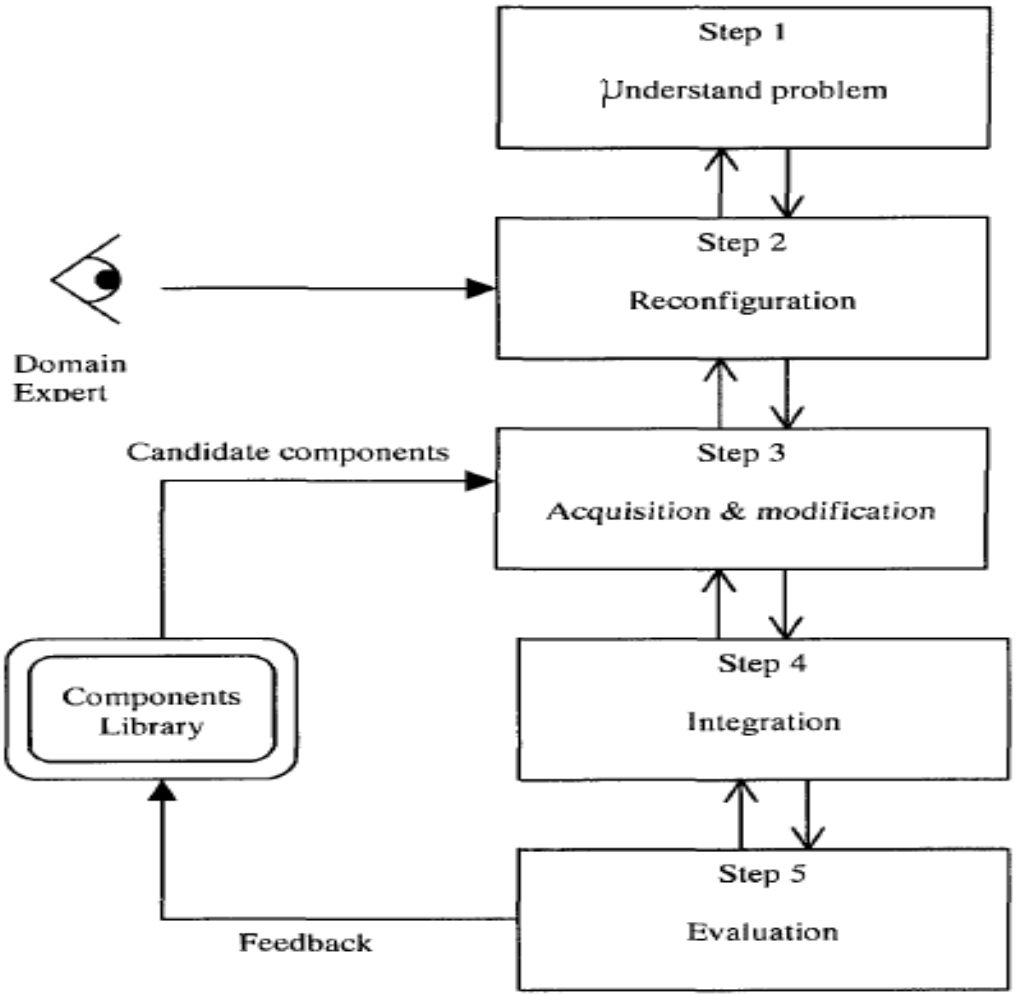
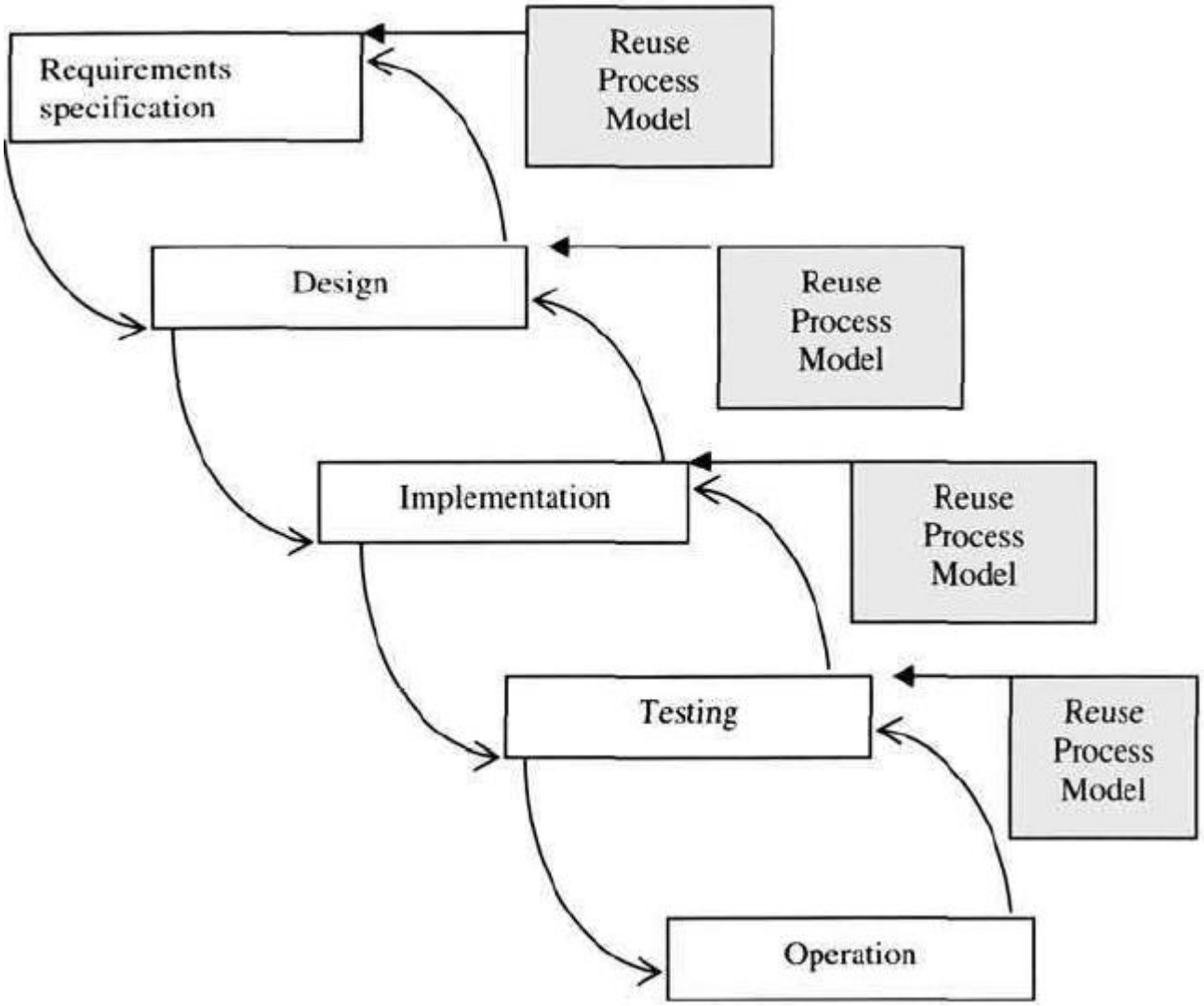


Figure 8.4 A generic reuse process model

Tính dung hòa mô hình qui trình ReUse



Các yếu tố tác động lên việc sử dụng lại

- *Yếu tố kỹ thuật*
  - Ngôn ngữ lập trình
  - Representation of Information
  - **Reuse Library**
  - **Reuse-Maintenance Vicious Cycle**
- ***Yếu tố Phi kỹ thuật***
  - **Initial Capital Outlay**
  - **Not Invented Here Factor**
  - **Commercial Interest**
  - **Education**
  - **Project Co-ordination**
  - **Legal Issues**

Bài tập

- **Exercise 8.6** Bạn vừa được tham gia nhóm kỹ sư trong đó bạn là người duy nhất học và thực tập phần mềm sử dụng lại và khả năng sử dụng lại. Công ty mà bạn làm việc không có chương trình reuse mặc dù họ sẵn sàng bắt đầu. Bạn được yêu cầu thực hiện chương trình reuse
  - Bước đầu tiên bạn làm đã trải qua?
  - Outline các bước kỹ thuật, quản lý, tổ chức bạn đã làm. Những thủ tục mà bạn đã triển khai để chương trình đi đến thành công
  - Những khó khăn mà bạn nhận biết và làm thế nào để vượt qua?

Bài tập

- **Exercise** 8.7 Một contractor đang sử dụng một hệ thống phần mềm lớn phức tạp viết bằng Fortran trên 12 năm. Không tài liệu cho hệ thống và người bảo trì đã bỏ qua công ty khác. Để thuận lời của phương pháp máy song song, contractor muốn phần mềm được thực hiện lại trên nền tảng song song.
  - Mô tả vắn tắt kỹ thuật mà bạn cần để hoàn thành tác vụ
  - Làm thế nào để thực hiện công việc, chứa phần sử dụng lại của phần mềm?

5.2 KiỂM THỬ

- Giới thiệu
- Định nghĩa
- Tại sao kiểm thử phần mềm
- Công việc của người kiểm thử phần mềm
- Kiểm thử gì và như thế nào
- Phân loại kiểm thử
- Thẩm định và đánh giá
- Kế hoạch kiểm thử
- Công cụ

Tại sao kiểm thử phần mềm

- Câu hỏi: Tại sao chúng kiểm thử phần mềm?
  - Trả lời: Xem nó có làm việc không?

Một câu hỏi và trả lời khác để bạn so sánh:

- Câu hỏi: Tại sao bạn lái xe qua thành phố hôm nay?
  - Trả lời: Nhìn xem thông báo giờ mở cửa của cửa hàng, xem nếu nó sẽ được mở vào thứ bảy ngày 3 tháng sáu trong năm sau.

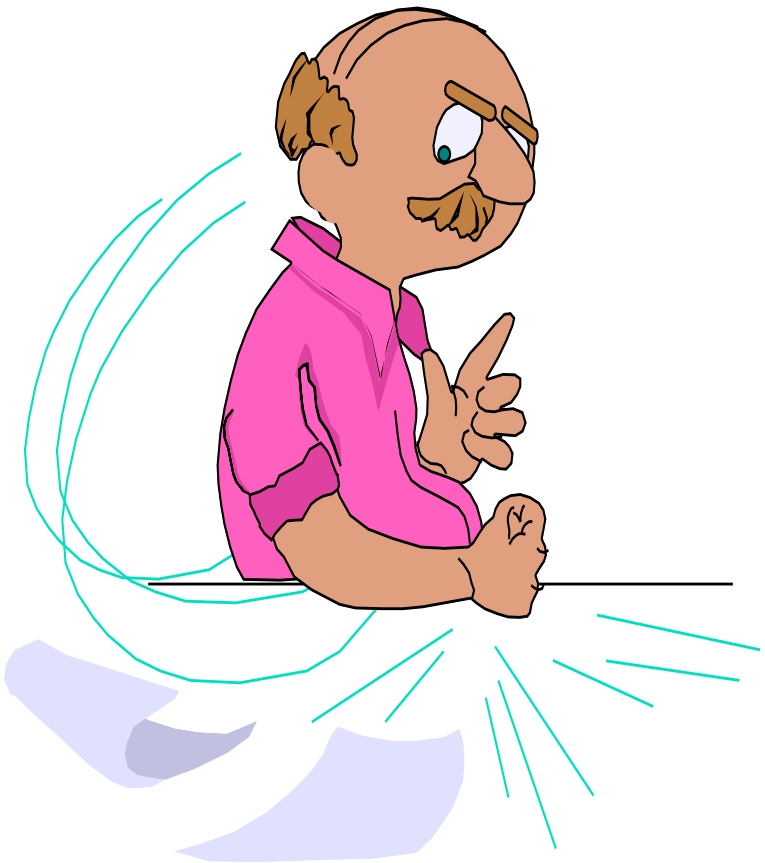
Tại sao phải kiểm tra phần mềm?

- Mặc dù được tự động hoá một phần bởi các công cụ CASE, rất nhiều công đoạn trong quá trình sản xuất phần mềm vẫn được thực hiện bởi con người

→ vẫn có khả năng xảy ra lỗi.

- Lỗi có thể xảy ra trong tất cả các giai đoạn: phân tích yêu cầu, thiết kế, mã hoá

→Do đó phải kiểm thử chương trình trước khi chính thức sử dụng



Tại sao kiểm thử lại cần thiết?

- Nhằm tăng độ tin cậy cũng như chất lượng của phần mềm.
- Giảm chi phí trong quá trình phát triển, nâng cấp, bảo trì phần mềm
- Ví dụ:
  - Website công ty có nhiều lỗi chính tả trong câu chữ →Khách hàng có thể lãng tránh công ty với lý do công ty trông có vẻ không chuyên nghiệp.
  - Một phần mềm tính toán lượng thuốc trừ sâu dùng cho cây trồng, vì lý do tính sai số lượng lên gấp 10 lần →Nông dân phải bỏ nhiều tiền mua, cây trồng hư hại, môi trường sống, nguồn nước bị ảnh hưởng,…

Lỗi tăng lên khi nào?

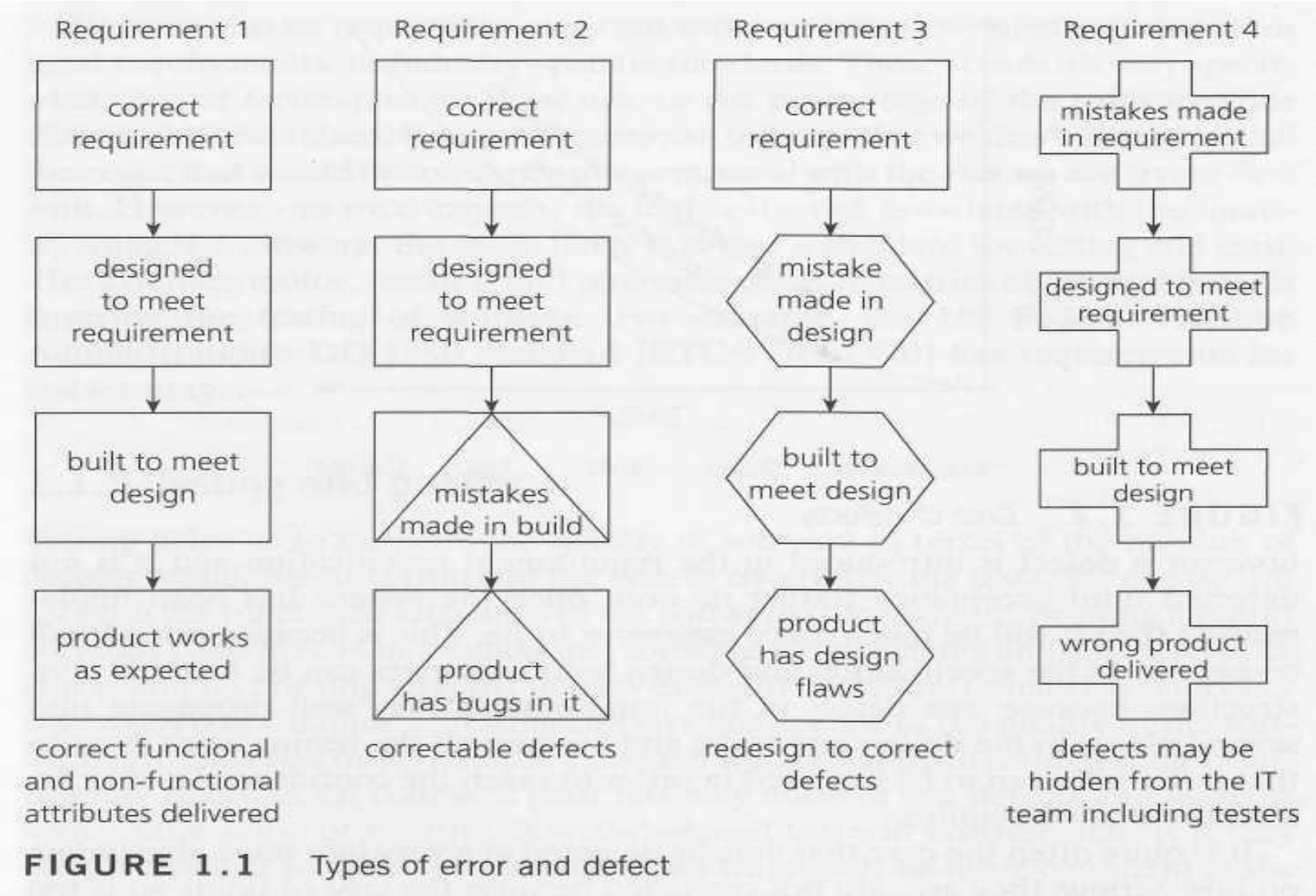


FIGURE 1.1 Types of error and defect

Lỗi tăng lên khi nào?

- Chi phí cho việc tìm thấy và sửa lỗi tăng dần trong suốt chu kỳ sống của phần mềm. Lỗi tìm thấy càng sớm thì chi phí để sửa càng thấp và ngược lại.

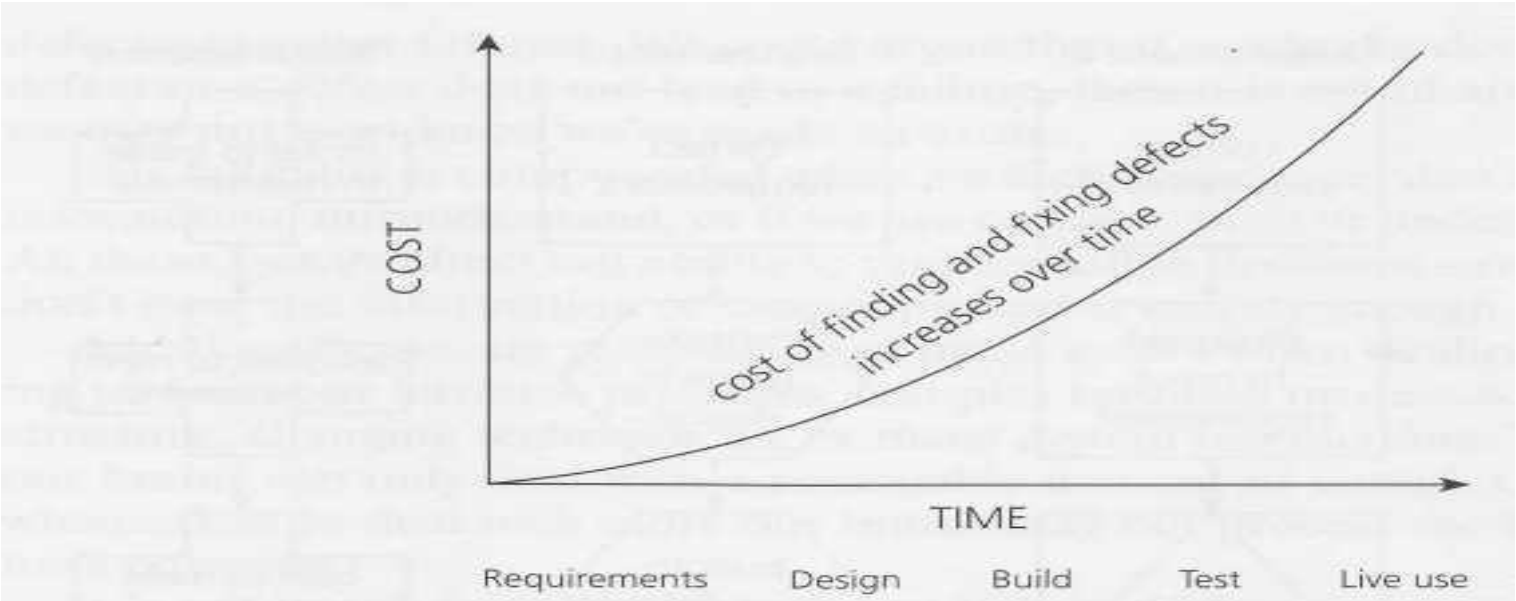


FIGURE 1.2 Cost of defects

Các nguyên lý trong kiểm thử PM

- Lập trình viên không nên thực hiện kiểm thử trên phần mềm mà mình đã viết
- Cần phải kiểm tra các chức năng mà phần mềm không thực hiện
- Tránh việc kiểm thử phần mềm với giả định rằng sẽ không có lỗi nào được tìm thấy
- Test case phải định nghĩa kết quả đầu ra rõ ràng
- Test case phải được lưu trữ và thực thi lại mỗi khi có sự thay đổi xảy ra trong hệ thống

Các nguyên lý kiểm thử phần mềm

- Việc kiểm thử nên hướng về yêu cầu của khách hàng
- Vấn đề kiểm thử nên được hoạch định trước một thời gian dài.
- Áp dụng nguyên lý Pareto (nguyên tắc 80-20):
  - 80% lỗi có nguyên nhân từ 20% các module
  - Cô lập và kiểm tra những module khả nghi nhất.
- Nên tiến hành từ nhỏ đến lớn: bắt đầu từ những module riêng biệt rồi sau đó tích hợp các module lại.
- Không thể kiểm thử triệt để một phần mềm.
- Nên được thực hiện bởi những đối tượng **KHÔNG** tham gia vào quá trình phát triển phần mềm.

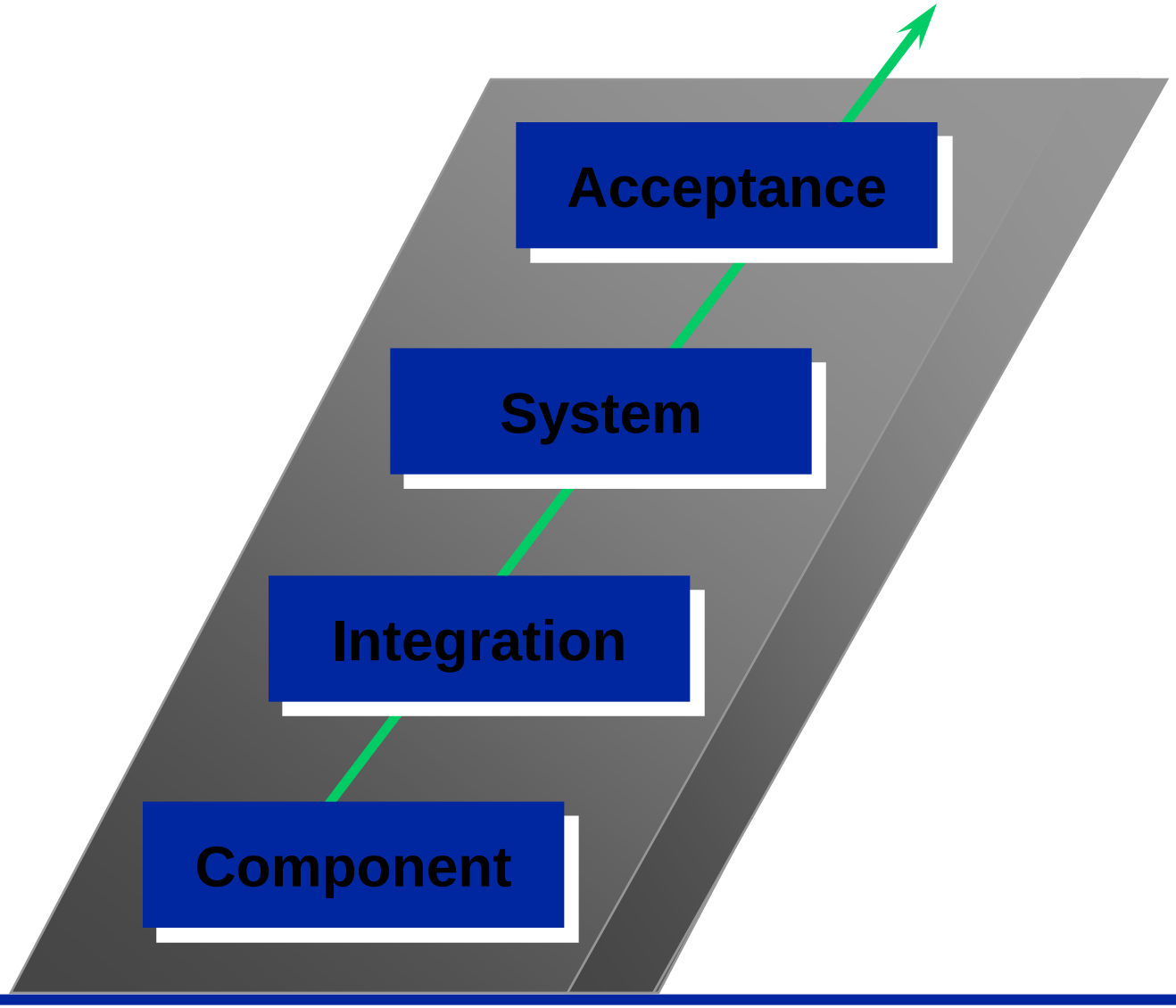
Vai trò kiểm thử

- Vai trò kiểm thử trong suốt quy trình sống của phần mềm
  - Kiểm thử không tồn tại độc lập.
  - Các hoạt động của kiểm thử luôn gắn liền với các hoạt động phát triển phần mềm.
  - Các mô hình phát triển phần mềm khác nhau cần các cách tiếp cận kiểm thử khác nhau.

Bài tập

- **Exercise 9.1** Chọn 2 hệ thống phần mềm và xem xét làm thế nào bạn thiết kế test case. Hệ thống có đặc tả hình thức bạn có sử dụng như cơ sở cho viết testcase? Bạn có nhận biết tập kiểm thử để thống kê toàn bộ chuỗi kết quả? Điều kiện đường biên thế nào?

Các mức độ kiểm thử (Test levels)



Các mức độ kiểm thử (Test levels)

- Component testing (unit testing):
  - Tìm lỗi trong các component của phần mềm như: modules, objects, classes,…
  - Do có kích thước nhỏ nên việc tổ chức, kiểm tra, ghi nhận và phân tích kết quả trên Unit test có thể thực hiện dễ dàng
  - Tiết kiệm thời gian, chi phí trong việc dò tìm và sửa lỗi trong các mức kiểm tra sau

Các mức độ kiểm thử (Test levels)

- Integration testing:
  - Test sự kết hợp của các component, sự tác động của các phần khác nhau trong một hệ thống, sự kết hợp của các hệ thống với nhau,…

Các mức độ kiểm thử (Test levels)

- System testing:
  - Đảm bảo rằng hệ thống (sau khi tích hợp) thỏa mãn tất cả các yêu cầu của người sử dụng
  - Tập trung vào việc phát hiện các lỗi xảy ra trên toàn hệ thống
- Acceptance testing:
  - Test phần mềm đứng dưới góc độ người dùng để xác định phần mềm có được chấp nhận hay không.

Các kỹ thuật kiểm thử

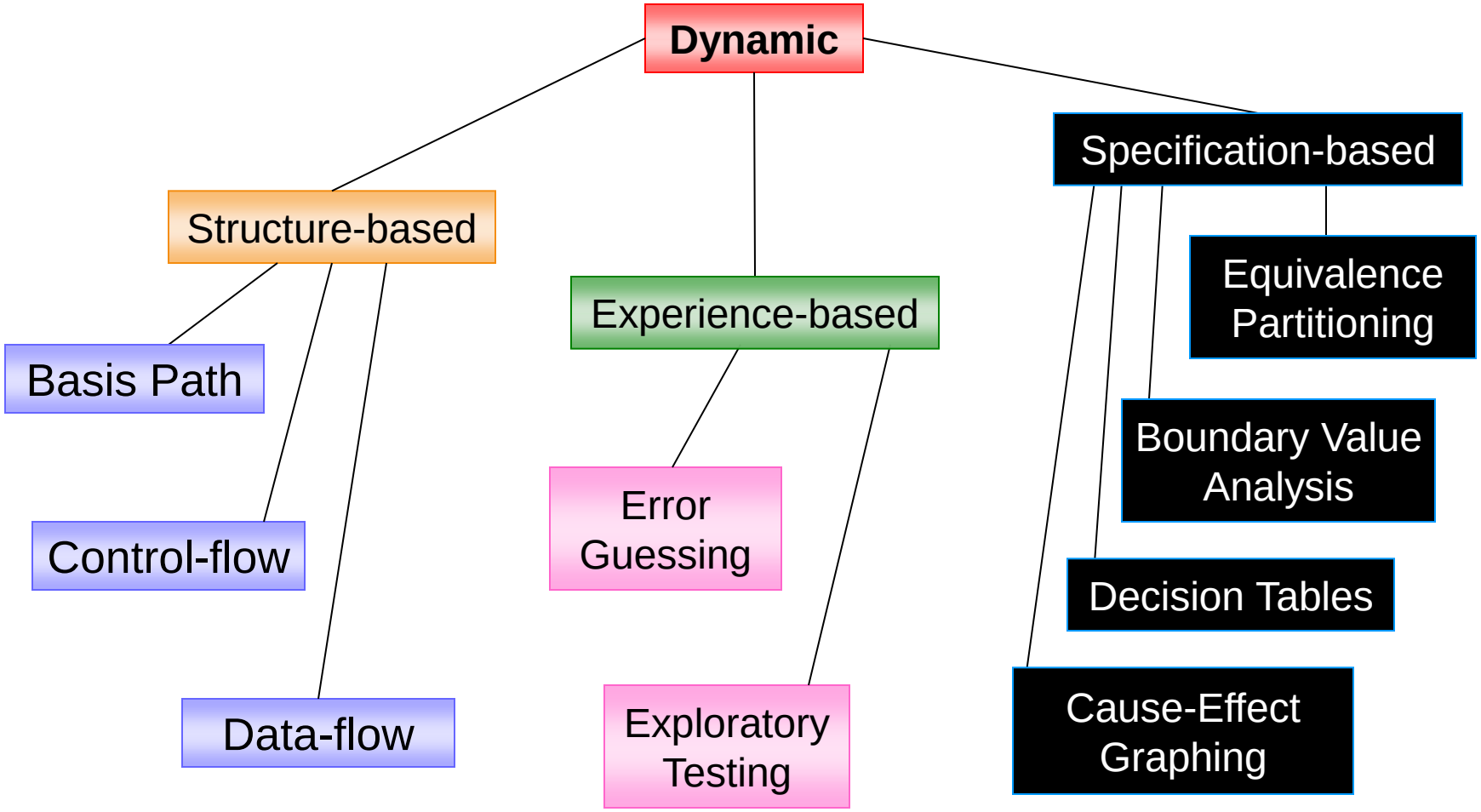
Test tĩnh (Static Verification)

- Thực hiện kiểm chứng mà không cần thực thi chương trình
- Kiểm tra tính đúng đắn của các tài liệu có liên quan được tạo ra trong quá trình xây dựng ứng dụng
- Đạt được sự nhất quán và hiểu rõ hơn về hệ thống
- Giảm thời gian lập trình, thời gian và chi phí test,…

Test động (Dynamic Testing)

- Thực hiện kiểm thử dựa trên việc thực thi chương trình

Dynamic Testing - Kiểm thử động



Các phương pháp kiểm thử (1)

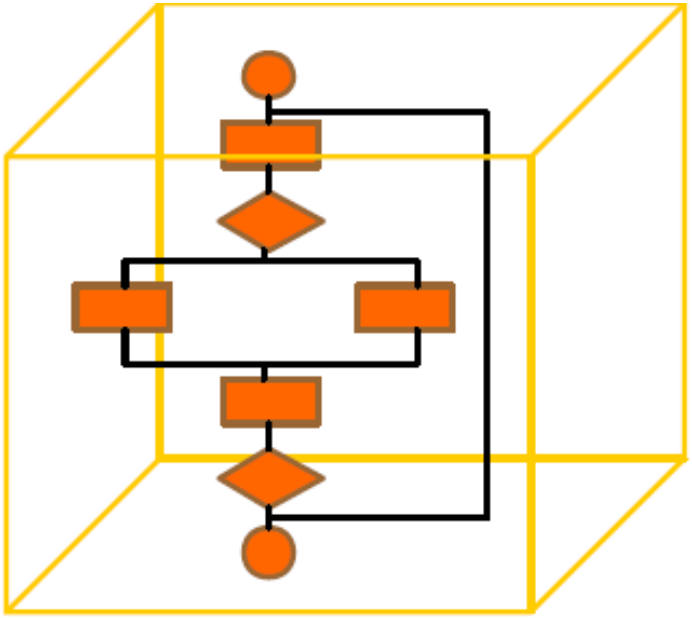
- Funtional Testing (Black Box Testing):
  - Test dựa trên mô tả, chúng ta xem xét phần mềm với các dữ liệu đầu vào và đầu ra mà không cần biết cấu trúc của phần mềm ra sao. Nghĩa là tester sẽ tập trung vào những gì mà phần mềm làm, không cần biết phần mềm làm như thế nào.
  - Ưu điểm:
    - ✓Không phụ thuộc vào việc thực hiện phần mềm
    - ✓Việc phát triển test case có thể diễn ra song song với quá trình thực hiện phần mềm → Rút ngắn thời gian thực hiện dự án

Các kỹ thuật kiểm thử hộp đen

- Kỹ thuật phân lớp tương đương (Equivalence Class Testing)
- Kỹ thuật dựa trên giá trị biên (Boundary Value Testing)
- Kỹ thuật dựa trên bảng quyết định (Decision Table-Based Testing)
- Kỹ thuật dựa trên đồ thị nguyên nhân – kết quả (causes-effects)
- …

Các phương pháp kiểm thử (2)

- Structural Testing (White Box Testing):
  - Test dựa trên cấu trúc còn được gọi là white-box hay glass-box bởi vì nó đòi hỏi sự hiểu biết về cấu trúc của phần mềm, nghĩa là phần mềm hoạt động như thế nào.



Các kỹ thuật kiểm thử hộp trắng

- Basis Path Testing
- Control-flow/Coverage Testing
- Data-flow Testing

Các phương pháp kiểm thử (3)

- Experience Testing (Test dựa trên kinh nghiệm)
  - Kỹ thuật này đòi hỏi sự hiểu biết, kỹ năng và kinh nghiệm của người test.
  - Dựa vào những kinh nghiệm thu thập được từ những hệ thống trước đó, tester có thể dễ dàng nhìn thấy được những điểm sai trong chương trình.

Verification and Validation

- Là chìa khóa trong hệ thống phát triển được tin tưởng.
- ***Verification***: các hành động để đảm bảo cho phần mềm được hiện thực đúng theo một chức năng cụ thể nào đó → **"Are we building the product right ?"**
- ***Validation***: các hành động để đảm bảo cho phần mềm được xây dựng theo đúng yêu cầu của khách hàng → **"Are we building the right product ?"**
- Đạt được hệ thống tốt hơn,i.e hệ thống với tính khả thi, tốc độ, chất lượng, và tinh hiệu quả chi phí cải tiến
- Tạo hệ thống phần mềm chất lượng [279], và hiệu quả hơn khi thực hiện độc lập nhóm phát triển hay bảo trì hệ thống

Test Plans

- The IEEE standard defines a test plan as
  - *"A document describing the scope, approach, resources, and schedule of intended testing activities. It identifies test items, the features to be tested, the testing tasks, who will do each task, and any risks requiring contingency planning."*

  [ANSI/IEEE Standard 829-1983 for Software Test Documentation]
- A test plan can either be a tool or a product [149 ch.12]
  - Kaner advises that a test plan whose purpose is to act as a tool is *"valuable ...to the extent that it helps you manage your testing project and find bugs. Beyond that, it is a diversion of resources"* [149 p.205].

Test Plans

- test plan tốt điều kiện thử nghiệm bằng nhiều cách bao gồm:
  - cung cấp danh sách các trường hợp thử nghiệm hữu ích xác định những thứ như điều kiện biên và các lớp dữ liệu thử nghiệm. Điều này cải thiện hiệu quả và có nghĩa là các trường hợp thử nghiệm quan trọng là ít có khả năng thể bỏ qua.
  - cung cấp thông tin về những quy mô của công việc có khả năng là những gì và nguồn lực sẽ được cần thiết
  - cung cấp thông tin để xác định và ưu tiên các nhiệm vụ, do đó tổ chức các đội kiểm tra giúp đỡ và xác định vai trò và trách nhiệm

Bài tập

- **Exercise 9.2** Viết test plan cho chương rình (kích thước có thể chấp nhận ít nhất 100LOC) mà không phải do bạn phát triển. Nếu bạn phải liên quan một phần của mẫu phá triển phần mềm này, hoán đổi code với sinh viên tập sự và viết test plan cho mỗi code khác nhau. Người viết code gốc nên nghiên cứu test plan tạo cho code của họ và thảo luận điểm mạnh điểm yếu. Cụ thể, xem bất kỳ không mong đợi mà có thể xuất phát trừ code của bạn.
- **Exercise** 9.3 xem những case study theo sau. Liệt kê những vấn đề liên quan đến message lỗi. Xem xét làm thế nào để có thể tránh và hình thành một số qui tắc để ngăn những vấn đề này trong hệ thống khác.
- 9.9 Case Study – Therac 25 (Đọc trong ebook chính)

Bài tập

- **Exercise 9.4 Báo cáo và phân tích biến cố** Therac-25 dễ dàng đạt được.Mô tả văn tắt case study được cho. Không bao hàm, ví dụ mở rộng mà người dùng Nó không bao gồm, ví dụ như mức độ thực sự mà người sử dụng là quan trọng trong việc khai thác những vấn đề, cũng không đi vào bất kỳ độ sâu về vấn đề tái sử dụng các thủ tục con phần mềm từ các phiên bản trước của phần mềm. Những sự kiện được sưu liệu tốt bị bẻ gãy bởi code enigma trong world war 2nd [286] và lội của Ariane 5 spacecraft, chuyến bay 501 trong 1996 [8]. Chọn một trong tình huống sau để kiểm tra kỹ:
  - Tập trung vào người dùng hệ thống, so sánh vai trò được thực bởi bởi người dùng của máy enigma (người tạo code và người breakers)với vai trò được thực hiện bởi người dùng của máy Therac.
  - So sánh tính khả dụng của qui trình con của phần mềm dùng lại, và vấn đề điều này gây ra, trong Therac-25 machine và Ariane 5 spacecraft.

Các công cụ hỗ trợ kiểm thử

- Các công cụ hỗ trợ quản lý quá trình kiểm thử
- Các công cụ hỗ trợ thực hiện các kỹ thuật kiểm thử
  - Công cụ kiểm thử hiệu năng (Performance)
  - Công cụ kiểm thử chức năng (Functional)
  - Công cụ kiểm thử bảo mật (Security)
  - Công cụ kiểm thử đơn vị (UnitTesting)
  - …

Các công cụ hỗ trợ quản lý quy trình kiểm thử phần mềm (1)

- Các đối tượng cần quản lý của 1 công cụ kiểm thử PM
  - Project
  - User
  - User Role
  - Requirement
  - Release: Phiên bản của project.
  - Test Plan: Kế hoạch test.
  - Test types: Các loại test.
  - Test cases: Các trường hợp test
  - Teststep: Các bước thực hiện cho mỗi test case
  - Result: Kết quả thực thi test.
  - Bug: Lỗi
  - Reports: Các thông báo về tình trạng của tiến trình: Tình trạng lỗi, tiến triển của công việc: …
  - Các tài liệu hướng dẫn sử dụng chương trình (Help)

Các công cụ hỗ trợ quản lý quy trình kiểm thử phần mềm (2)

- Các chức năng cần phải có
  - Quản lý project.
  - Quản lý User.
  - Phân quyền User.
  - Quản lý requirement theo phiên bản.
  - Quản lý release.
  - Quản lý các thành phần của release: build, component,..
  - Quản lý testplan.
  - Quản lý testcase.
  - Cập nhật kết quả cho test case.
  - Cập nhật tình trạng lỗi.
  - Thống kê lỗi cho mỗi release hoặc mỗi thành phần của release.
  - Tự động cập nhật kết quả kiểm thử

Các công cụ hỗ trợ quản lý quy trình kiểm thử phần mềm (3)

No	Name	Desc	REq	Download
1	TestLink		Apache, MySQL, PHP	48797
2	Fitnesse		Mac, Wnidows, POSIX	24475
3	QATraq		Windows, BSD, Linux, SunOS/Solaris	21992
4	Bugzilla Test Runner		Bugzilla 2.16.3 or above	17291
5	rth		All 32-bit MS Windows (95/98/NT/2000/XP), All POSIX (Linux/BSD/UNIX-like OSes), IBM AIX	9563
6	TestMaster		Linux, Apache, PostgreSQL	6728
7	TCW		Any (PHP/SQL/Apache)	4488
8	Tesly		OS Independent	3327
9	qaProjectManager		Platform Independent	3133
10	Testitool		Apache, PHP, MySQL	701

www.opensourcetestingtools.org

Công cụ kiểm thử hiệu năng

- Là một dạng kiểm tra tự động nhằm tìm ra những điểm "thắt cổ chai" của phần mềm, giúp cho người phát triển có những thay đổi thích hợp để tăng khả năng thực thi, tốc độ xử lý của phần mềm
- Giúp người kiểm tra xác định được những thông số ngưỡng của phần mềm, đề ra tiêu chuẩn cho những lần kiểm tra sau
- Thường được áp dụng đối với các PM được triển khai trên môi trường nhiều người dùng (ví dụ: ứng dụng web)
- Kết quả mong đợi của việc kiểm thử hiệu năng phải được định nghĩa một cách rõ ràng
- Ví dụ:
  - Số kết nối (session) đồng thời mà server có thể phục vụ
  - Thời gian (bao nhiêu phút/giây) mà trình duyệt nhận được kết quả từ server ….

Công cụ kiểm thử hiệu năng

No	Name	Requirements	Download
1	OpenSTA	Windows 2000, NT4 and XP	251965
2	Grinder	OS Independent	156458
3	TPTEST	MacOS/Carbon and Win32	108036
4	Database Opensource Test Suite	Linux, POSIX	103484
5	Sipp	Linux/Unix/Win32-Cygwin	102111
6	WebLOAD	32-bit MS Windows (NT/2000/XP), Linux, Windows Server 2003	39401
7	OpenWebLoad	Linux, DOS	31204
8	Hammerhead 2 - Web Testing Tool	Hammerhead has been used with Linux, Solaris and FreeBSD.	24814
9	Dieseltest	Windows	14618
10	DBMonster	OS Independent	13710

www.opensourcetestingtools.org

Các công cụ hỗ trợ kiểm thử đơn vị

- Có rất nhiều công cụ kiểm thử đơn vị được viết bằng nhiều ngôn ngữ khác nhau
  - ADA
  - C++
  - HTML
  - Java
  - .NET
  - Pert
  - PHP
  - SQL
  - XML
  - Ruby
  - …

Các công cụ hỗ trợ kiểm thử đơn vị

No	Name	Requirements	Download
1	JUnit	OS Independent	2151874
2	Findbugs	JRE (or JDK) 1.4.0 or later	379779
3	PMD	JDK 1.3 or higher	344688
4	Checkstyle	OS Independent	216780
5	EclEmma	Eclipse	209153
6	Dbunit	JUnit	129300
7	StrutsTestCase for JUnit v1.9.5	OS Independent	106860
8	Emma	Java	59435
9	MockObjects	OS independent	55457
10	JUnitEE	JUnit	54618

www.opensourcetestingtools.org

Các công cụ hỗ trợ kiểm thử đơn vị

No	Name	Requirements	Download
1	NUnit	Windows NT/2000	1061875
2	NUnitAsp	Windows NT/2000	72724
3	NUnit Addin for Visual Studio.NET	Windows	58588
4	NUnitForms	Windows NT/2000	46880
5	csUnit	csUnit has been tested using the Microsoft .NET framework 1.0 Service Pack 2 runtime on an Intel-compatible platform.	31483
6	NCover	All 32-bit MS Windows (95/98/NT/2000/XP)	14264
7	VSNUnit	All 32-bit MS Windows (95/98/NT/2000/XP)	8763
8	dotUnit	All 32-bit MS Windows (95/98/NT/2000/XP)	6230
9	.NETUnit	OS Independent (Written in an interpreted language)	5558
10	ASPUnit	Microsoft Internet Information Server 5.0 or 5.1	5197

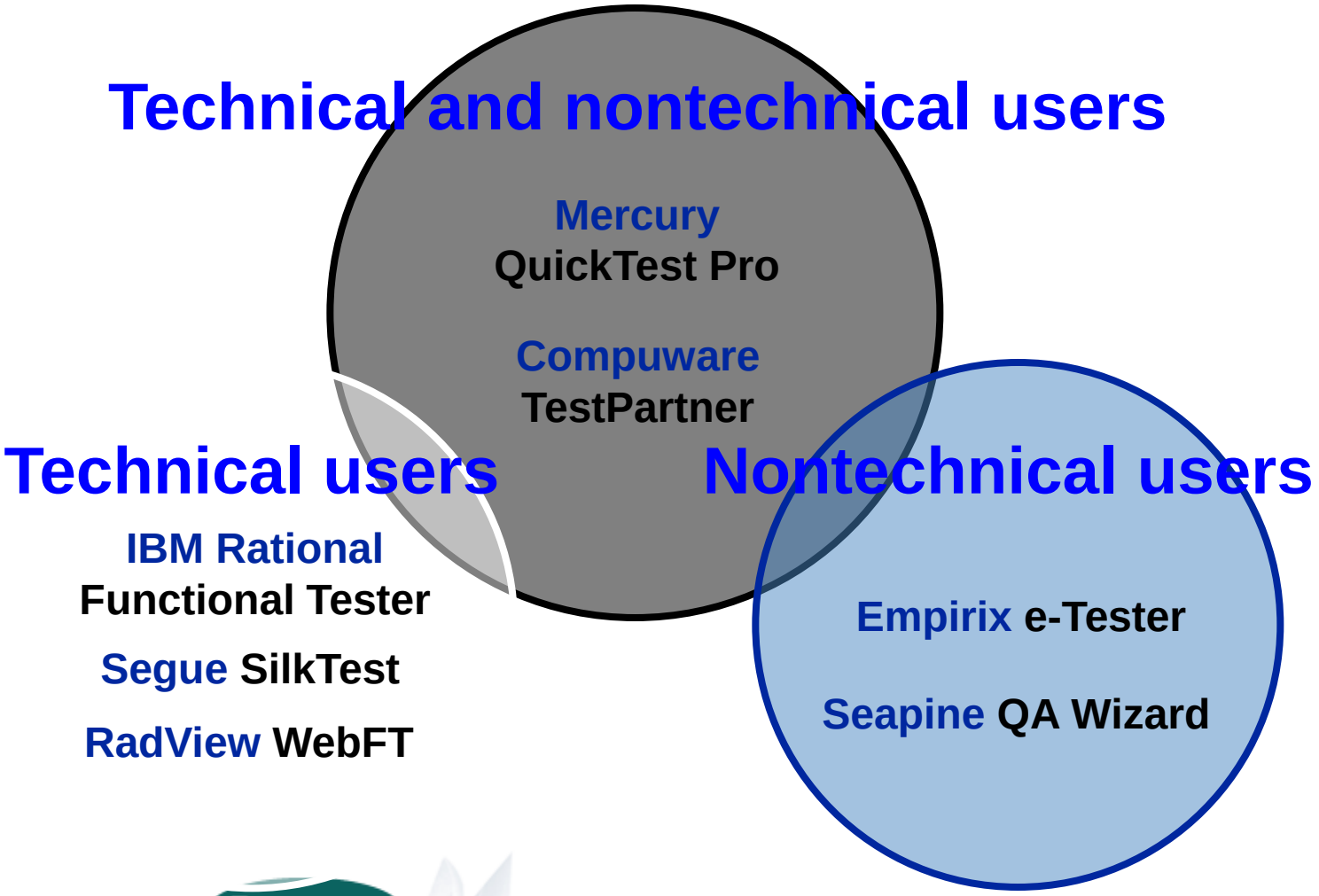
Một sô công cụ hỗ trợ- kiểm thử chức năng

No	Name	Desc	Req	Download
1	Software Testing Automation Framework (STAF)		Windows, Linux, Solaris, AS/400, AIX, HP-UX, Irix	212018
2	soapui		Java 1.5	178985
3	Linux Test Project		Linux	103484
4	jWebUnit		OS Independent	56526
5	Abbot Java GUI Test Framework		TBC	56118
6	Software Automation Framework Support		All 32-bit MS Windows (95/98/NT/2000/XP)	43735
7	Jameleon		OS Independent, JDK 1.4 or higher	43507
8	WebInject		Windows, OS Independent, Linux	40891
9	Marathon		Java 1.3 or later	30328
10	Solex		Eclipse 2.1 or above	29591

Các công cụ kiểm thử thương mại

Vendor	Tool	Name of testing suite or companion tools
Compuware	TestPartner	*QA*Center Enterprise Edition+
Empirix	e-Tester	e-TEST suite
IBM Rational	Functional Tester	Test Manager, Manual Tester, Performance Tester
Mercury	QuickTest Professional	Quality Center
RadView	WebFT	TestView Suite
Seapine	QA Wizard	TestTrack Pro
Segue	SilkTest	SilkCentral, SilkPerformer

Các công cụ kiểm thử thương mại



FORRESTER® MAKING LEADERS SUCCESSFUL EVERY DAY

Tài liệu tham khảo

- Software Testing, A Craftsman's Approach, Paul C.Jorgensen
- Practical Software Testing, EleneBurnstein
- Slides: Software Testing ISEB Foundation Certificate Course
- Slides: Software Testing, Dr. Balla Katalin
- Slide: Equivalence Class Testing, Prof. Schlingloff & Dr. M Roggenbach
- Slide: Decision Table Based Testing, Neelam Gupta, The University of Arizona Tucson, Arizona, USA
- Object Oriented Testing, Ali Kamandi, Sharif University of Technology

Yêu cầu thực hiện tuần tiếp theo

- Viết lại các báo cáo cho các thảo luận trên lớp và các bài tập
- Đọc thêm các tài liệu cung cấp ReUse-1.ppt & ReUse-2.ppt
- Đọc thêm tài liệu về kiểm thử
- Tiếp tục chuẩn bị công việc cho nhóm
- Mỗi nhóm tự chuẩn bị tìm hiểu và thử nghiệm một trong các công cụ hỗ trợ qui trình bảo trì → hướng dẫn sử dụng và demo trước lớp