Kỹ thuật phần mềm

Thẩm định và kiểm định

Mục tiêu

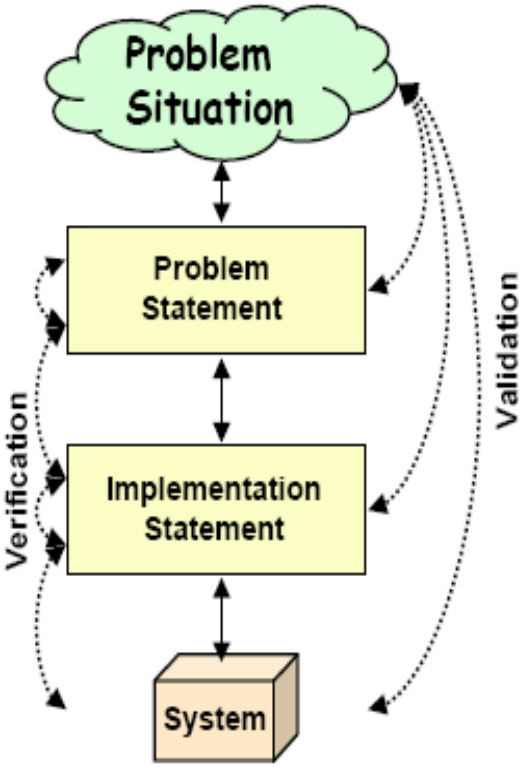
- Thẩm định và kiểm định phần mềm là gì?
  - Phân biệt
- Quy trình kiểm tra chương trình và vai trò của nó trong thẩm định và kiểm định
- Kĩ thuật kiểm định phân tích tĩnh

Các chủ đề

- Lập kế hoạch thẩm định và kiểm định
- Software inspections
- Phân tích tĩnh được tự động hóa

Thẩm định và kiểm định– V&V

- Thẩm định - Validation:
  "Are we building the right product?"
  Phát biểu bài toán có phản ánh chính xác bài toán thực hay không?
  Ta đã xét đến nhu cầu của tất cả các stakeholder chưa?
- Kiểm định - Verification:
  "Are we building the product right?"
  Thiết kế có tuân theo theo đặc tả không?
  Cài đặt có thỏa mãn đặc tả không?
  Hệ thống được giao cho khách hàng có thực hiện đúng những gì mà ta nói là nó sẽ làm?
  Các mô hình yêu cầu của ta có nhất quán với nhau không?

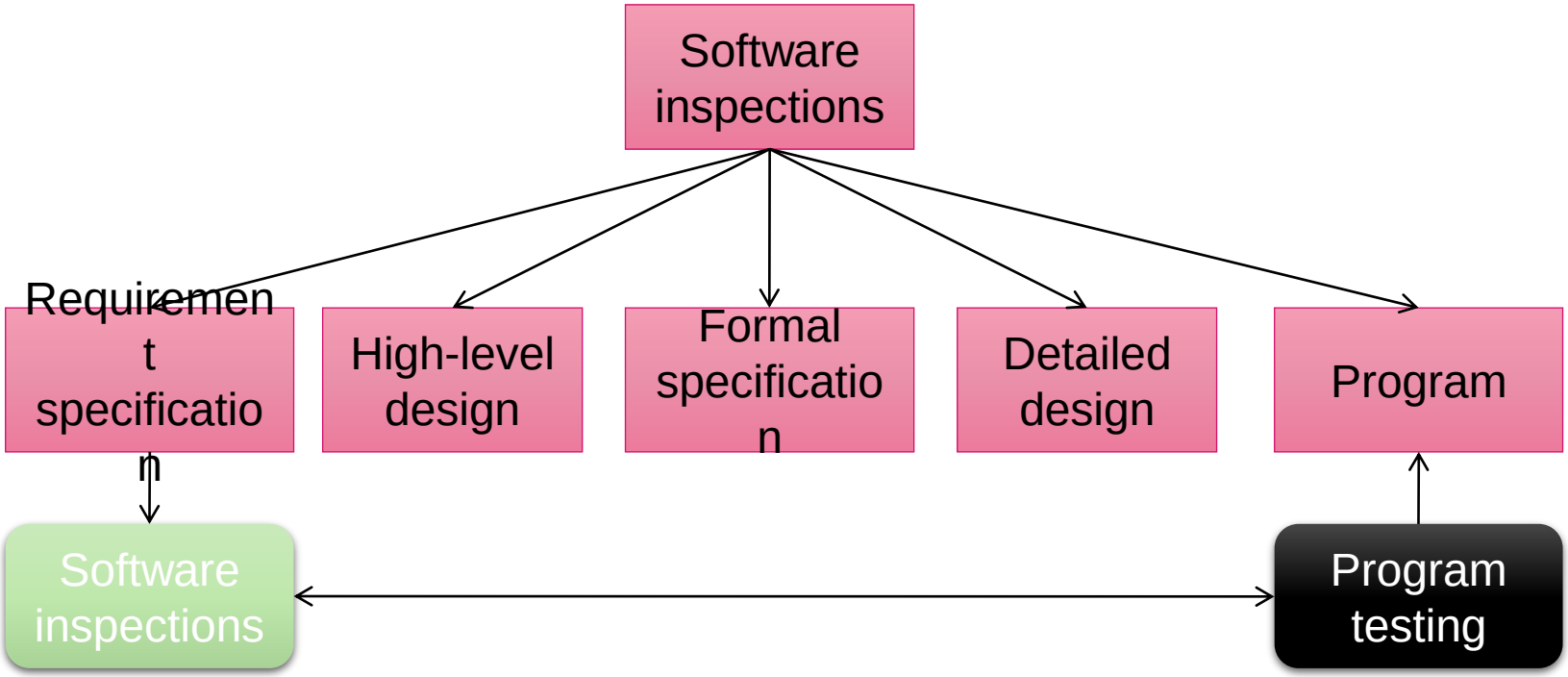


Source: Steve Easterbrook, 2008. CSC320, Uni of Toronto

Quy trình V&V

- Quy trình kéo dài toàn bộ chu trình sống
  - V&V phải được áp dụng tại từng bước trong quy trình phần mềm
- Hai mục tiêu chính
  - Phát hiện các khiếm khuyết trong một hệ thống;
  - Đánh giá xem hệ thống có hữu ích và dùng được trong một tình huống vận hành hay không.

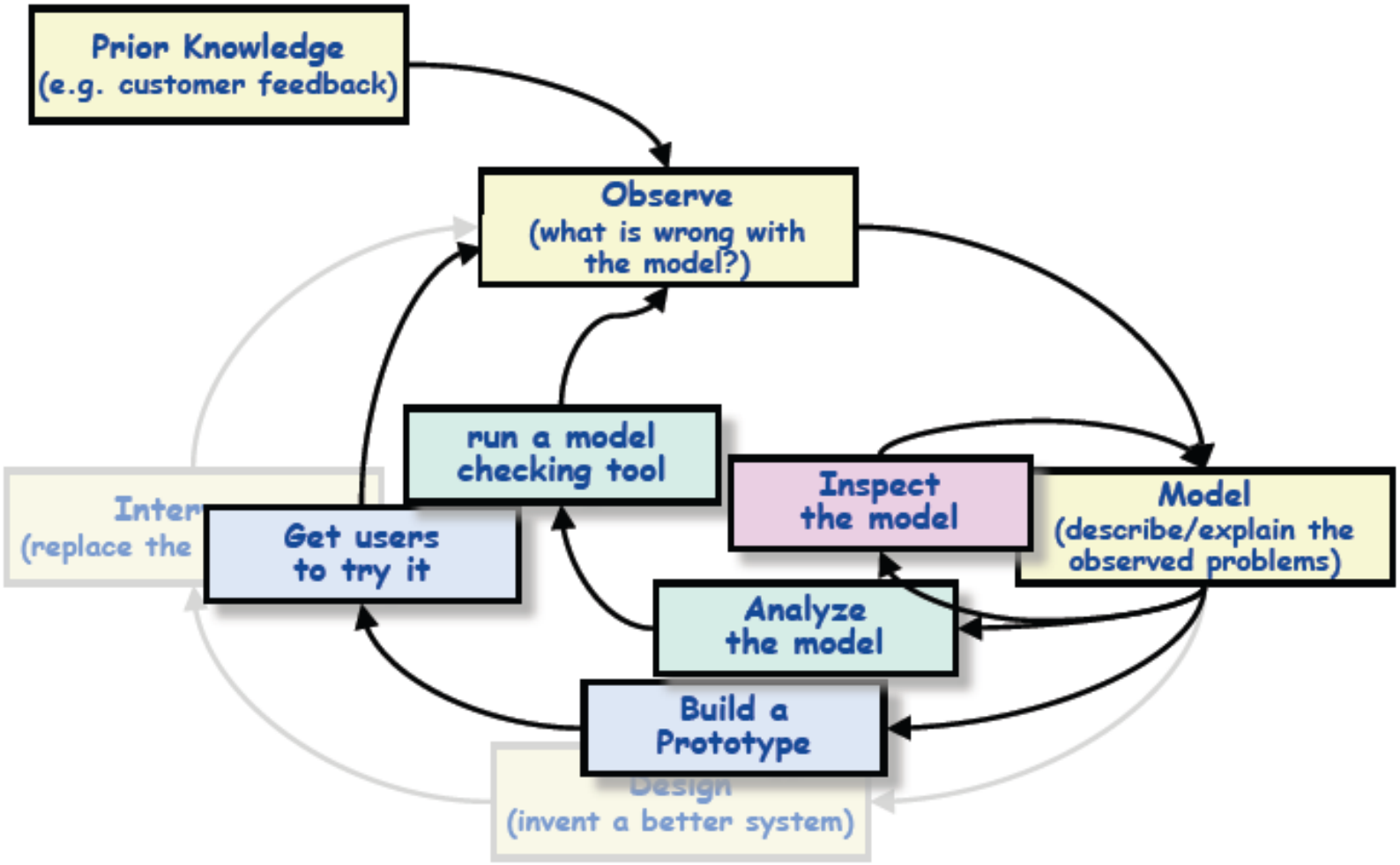
Static and dynamic V&V



Kiểm định

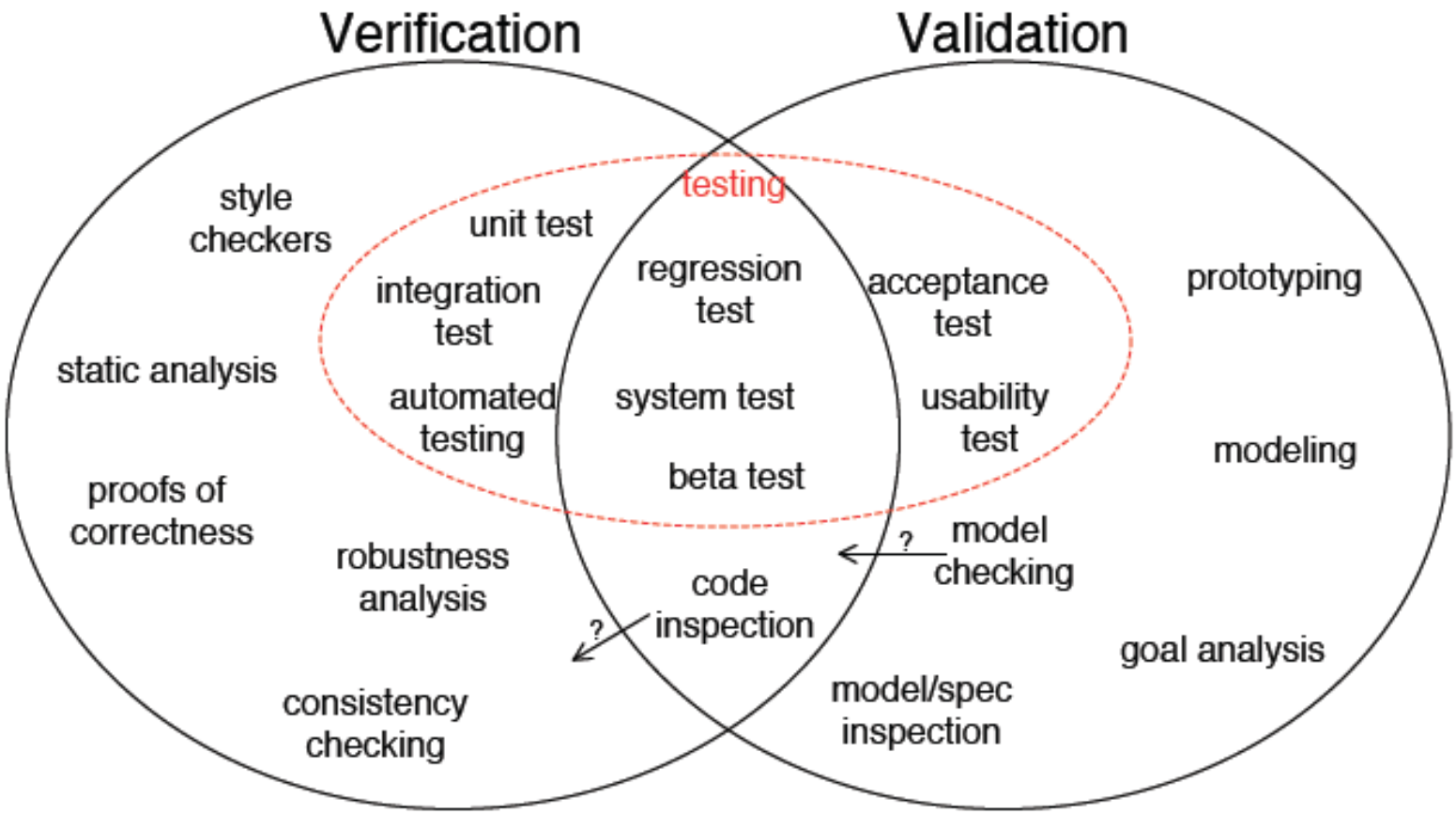
- Kiểu truyền thống (code verification)
  - Kiểm thử chương trình – testing
  - Duyệt chương trình – inspection, reviews
- Dựa vào mô hình (model-based verification)
  - Các use case có thỏa mãn yêu cầu không? – goal analysis
  - Mô hình lớp đối tượng có thỏa mãn các use case không? – robustness analysis
  - Mã chương trình có nhất quán với mô hình hay không? – consistency checking

Các kĩ thuật thẩm định



Source: Steve Easterbrook, 2008. CSC320, Uni of Toronto

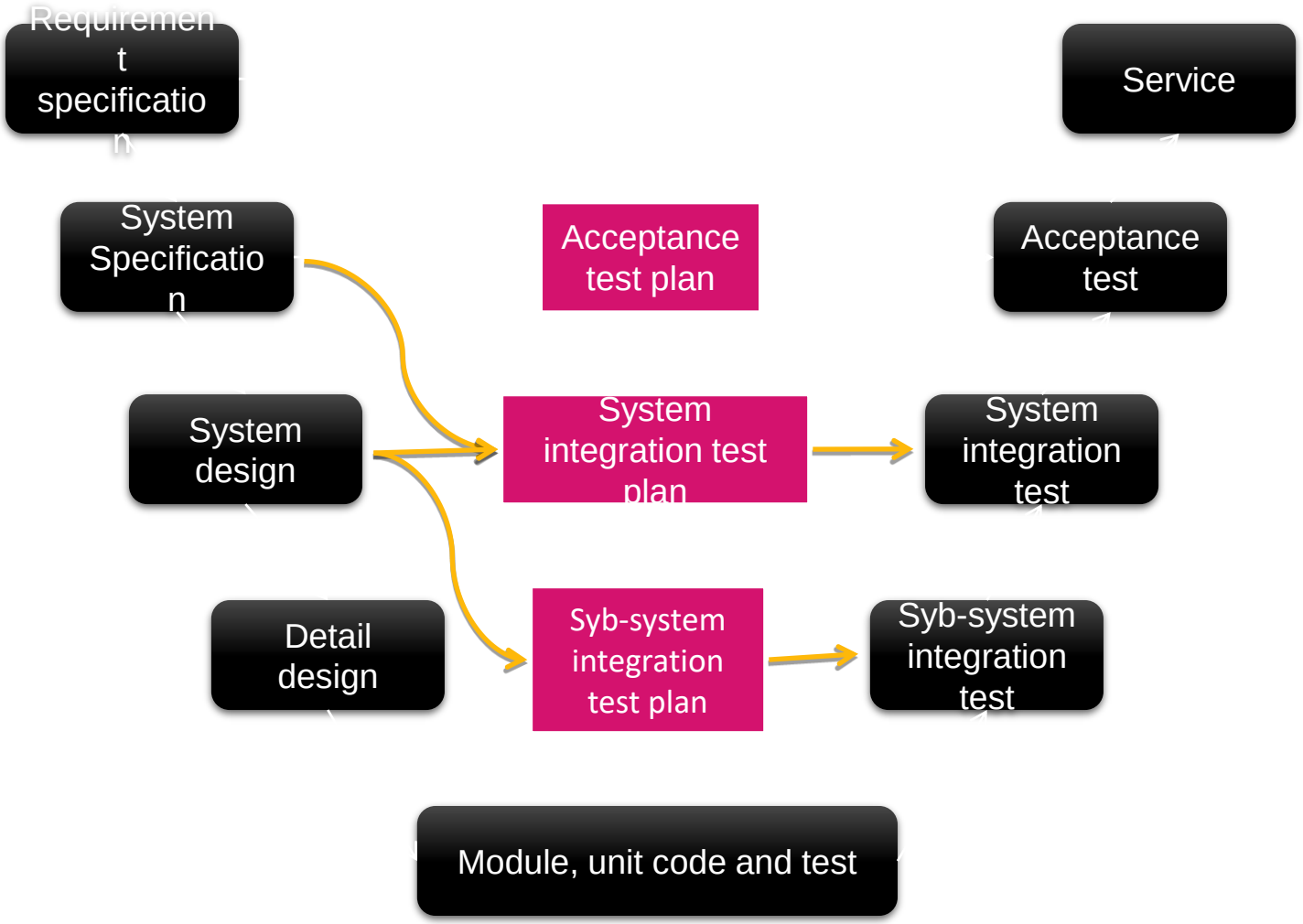
Lựa chọn kĩ thuật



Lập kế hoạch V&V

- V&V là quy trình rất tốn kém, có thể chiếm đến 50% tổng chi phí
- Lập kế hoạch tốt để thu được hiệu quả cao nhất của các quy trình kiểm thử (testing) và duyệt (inspection)
- Cần bắt đầu sớm trong quy trình phát triển.
- Kế hoạch cần xác định sự cân bằng giữa kiểm thử và duyệt

The V-model of development



Cấu trúc của một test plan

- The testing process.
  - Mô tả các pha của quy trình test
- Requirements traceability.
  - Đảm bảo từng yêu cầu đều được test
- Tested items.
  - Liệt kê các sản phẩm quy trình cần được test
- Testing schedule.
  - Lịch kiểm thử và các tài nguyên cấp phát cho việc kiểm thử
- Test recording procedures.
  - Quy trình thủ tục để ghi lại quá trình test một cách có hệ thống
- Hardware and software requirements.
  - Các công cụ phần mềm cần dùng và ước lượng về nhu cầu phần cứng
- Constraints.
  - Các hạn chế ảnh hưởng đến việc kiểm thử, chẳng hạn thiếu nhân lực

Software inspections
Duyệt phần mềm

- Kiểm tra phần mềm để phát hiện bất thường, lỗi, thiếu....
- Áp dụng cho mọi loại biểu diễn của hệ thống
  - Tài liệu yêu cầu, thiết kế, dữ liệu cấu hình, dữ liệu test, mã nguồn,.....
- Đã được chứng tỏ là kĩ thuật hiệu quả cho việc tìm lỗi chương trình
- Bổ trợ và kết hợp với software testing trong quy trình V&V
  - Kiểm tra được xem đặc tả có được tuân theo hay không
  - Không kiểm tra được các tính chất phi chức năng (hiệu năng, độ tin cậy ...)

Program inspections
Kiểm tra chương trình

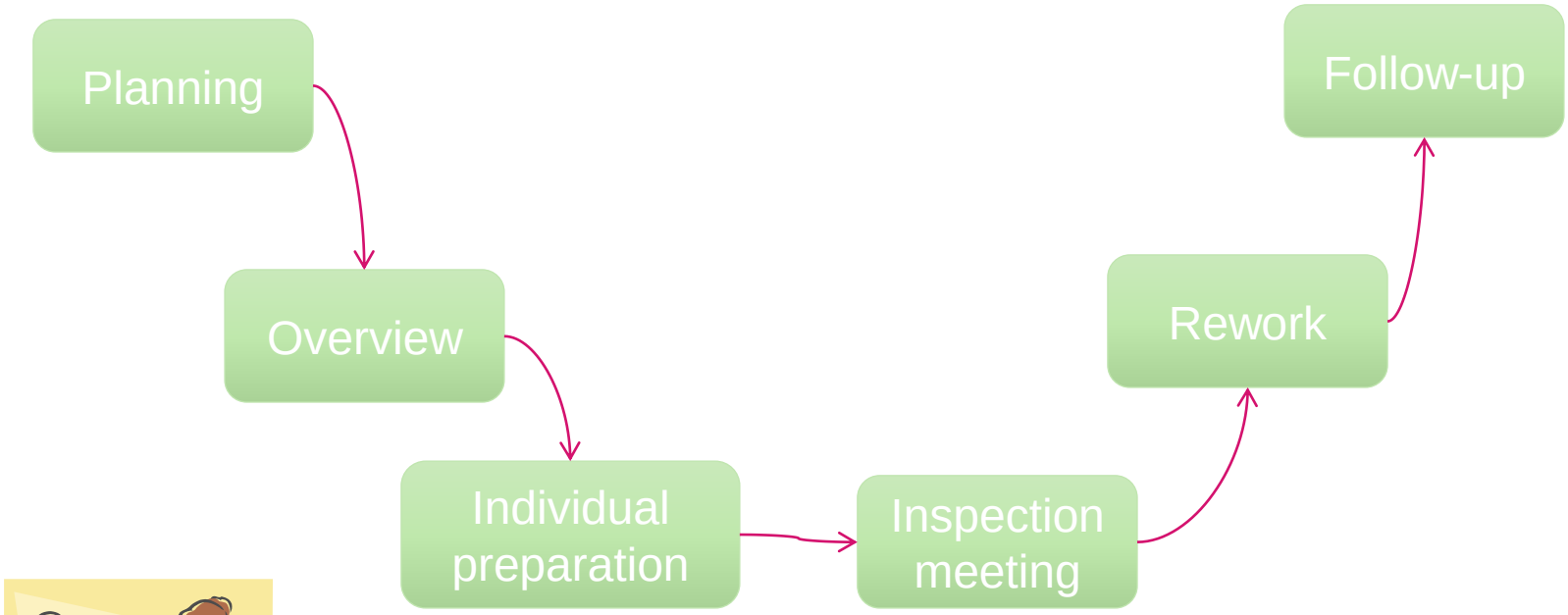
- Cách tiếp cập đã được chuẩn hóa cho việc duyệt tài liệu
- Mục tiêu là để phát hiện khiếm khuyết (không phải để sửa)
- Các khiếm khuyết có thể là các lỗi lô-gic, các bất thường trong mã mà có thể là dấu hiệu của lỗi
  - Biến chưa khởi tạo
  - Không theo chuẩn
  - ...

```
int f() {
  int x;
  int y = x * 2;
  return y;
}
```

Chuẩn bị trước khi kiểm tra

- Phải có một đặc tả chính xác
- Đội kiểm tra phải quen với các chuẩn của tổ chức
- Mã chương trình hoặc tài liệu khác phải đúng cú pháp và đã đủ nội dung
- Cần chuẩn bị một error checklist
- Quản lý
  - Không dùng các cuộc kiểm tra để khen/chê nhân viên
    - tìm ra ai là thủ phạm

Quy trình kiểm tra



Các thủ tục kiểm tra

1. Trình bày tổng quan hệ thống cho đội kiểm tra
2. Mã chương trình và các tài liệu có liên quan được giao trước cho đội kiểm tra
3. Thực hiện kiểm tra và ghi lại các lỗi phát hiện được
4. Sửa các lỗi đã phát hiện
5. Một cuộc tái kiểm tra có thể cần hoặc không cần đến

Vai trò kiểm tra

Author or owner	The programmer or designer responsible for producing the program or document. Responsible for fixing defects discovered during the inspection process.
Inspector	Finds errors, omissions and inconsistencies in programs and documents. May also identify broader issues that are outside the scope of the inspection team.
Reader	Presents the code or document at an inspection meeting.
Scribe	Records the results of the inspection meeting.
Chairman or moderator	Manages the process and facilitates the inspection. Reports process results to the Chief moderator.
Chief moderator	Responsible for inspection process improvements, checklist updating, standards development etc.

Checklist trong kiểm tra

- Checklist cho các lỗi thường gặp
- Error checklist phụ thuộc ngôn ngữ lập trình
  - vd. C++: rò rỉ bộ nhớ, con trỏ lạc...
- Nói chung, ngôn ngữ định kiểu (type checking) càng yếu thì checklist càng dài
  - Java: định kiểu mạnh hơn
  - PHP: định kiểu yếu hơn
- Ví dụ: khởi tạo, tên hằng, kết thúc vòng lặp, giới hạn mảng, v.v..

Đánh giá

- Inspection là quy trình tốn kém
  - Trình bày tổng quan: 500 lệnh/giờ
  - Cá nhân kiểm tra: 125 lệnh/giờ
  - Kiểm tra trong buổi gặp: 90-125 lệnh/giờ
- Kiểm tra 500 dòng lệnh tốn tổng cộng khoảng 40 giờ làm việc (man/hour)

Tự động hóa phân tích tĩnh

- Static analyser là các công cụ phần mềm để xử lý mã nguồn dạng text
- Chúng phân tích text của chương trình để phát hiện các lỗi tiềm tàng
  - Errors and warnings
- Hiệu quả trong việc hỗ trợ inspection
  - Hỗ trợ chứ không thay thế

Kiểm tra phân tích tĩnh

Các dạng lỗi	Static analysis check
Data faults (Lỗi dữ liệu)	Biến dùng trước khi khởi tạo Biến được khai báo nhưng không dùng Biến được gán liền hai lần mà không dùng đến Có thể vượt ra ngoài mảng Biến chưa khai báo
Control faults (Lỗi điều khiển)	Lệnh không bao giờ chạy đến Rẽ nhánh không điều kiện vào vòng lặp
Input/output faults (Lỗi vào ra dữ liệu)	Biến được output hai lần liên tiếp mà không được gán trị ở giữa
Interface faults (Lỗi giao diện)	Kiểu tham số không khớp Số tham số không khớp không dùng đến kết quả trả về của hàm Hàm và thủ tục không được gọi
Storage management faults (Lỗi quản lý lưu trữ)	Con trỏ không được gán Các phép tính con trỏ

LINT static analysis

```
1. #include <stdio.h>
2. printarray (int Anarray) {
3.   printf("%d",Anarray);
4. }
5. main () {
6.   int Anarray[5];
7.   int i; char c;
8.   printarray (Anarray, i, c);
9.   printarray (Anarray) ;
10.}
```

lint_ex.c(8): warning: c may be used before set
lint_ex.c(8): warning: i may be used before set
printarray: variable # of args. lint_ex.c(2) :: lint_ex.c(8)
printarray, arg. 1 used inconsistently lint_ex.c(2) :: lint_ex.c(8)
printarray, arg. 1 used inconsistently lint_ex.c(2) :: lint_ex.c(9)
printf returns value which is always ignored

Sử dụng static analysis

- Đặc biệt giá trị cho các ngôn ngữ định kiểu yếu (weak typing)
  - Nhiều lỗi trình biên dịch không phát hiện được
  - C, C++
- Ít hiệu quả hơn cho các ngôn ngữ định kiểu mạnh
  - Trình biên dịch phát hiện được nhiều lỗi
  - Java, C#

Verification and formal methods
Kiểm định và các phương pháp hình thức

- Các phương pháp hình thức (PPHT) để phát triển phần mềm dựa vào biểu diễn toán học của phần mềm, thường ở dạng đặc tả hình thức (dùng đặc tả toán học)
- Các ppht quan tâm đến: phân tích toán học về đặc tả, biến đổi đặc tả thành một biểu diễn chi tiết hơn nhưng tương đương về ngữ nghĩa, chứng minh sự tương đương về ngữ nghĩa
  - chứng minh rằng một chương trình tuân theo đặc tả toán học của nó

$$
\begin{array}{lcll}
Proc & ::= & STOP & \\
 & | & SKIP & \\
 & | & e \rightarrow Proc & (prefixing) \\
 & | & Proc \,\Box\, Proc & (external\ choice) \\
 & | & Proc \,\sqcap\, Proc & (nondeterministic\ choice) \\
 & | & Proc \,|||\, Proc & (interleaving) \\
 & | & Proc \,|[\{X\}]|\, Proc & (interface\ parallel) \\
 & | & Proc \setminus X & (hiding) \\
 & | & Proc; Proc & (sequential\ composition) \\
 & | & \text{if } b \text{ then } Proc \text{ else } Proc & (boolean\ conditional) \\
 & | & Proc \,\triangleright\, Proc & (timeout) \\
 & | & Proc \,\triangle\, Proc & (interrupt)
\end{array}
$$

$$
\begin{array}{rcl}
P & ::= & x(y).P \\
 & | & \overline{x}\langle y\rangle.P \\
 & | & P|P \\
 & | & (\nu x)P \\
 & | & !P \\
 & | & 0
\end{array}
$$

Ưu/Nhược điểm của các phương pháp hình thức

- Phát hiện lỗi tại đặc tả yêu cầu
  - Việc tạo một đặc tả toán học đòi hỏi phân tích kĩ càng các yêu cầu
- Có thể phát hiện lỗi cài đặt trước khi test chương trình
  - Khi chương trình được phân tích cùng với đặc tả

- Khó
  - Cần đến các kí pháp chuyên sâu mà các chuyên gia trong miền ứng dụng không hiểu được
- Chi phí cao
  - Chi phí về thời gian và công sức cho việc viết đặc tả
  - Việc chứng minh một chương trình thỏa mãn đặc tả đó còn đắt đỏ hơn nữa

Tóm tắt

- Thẩm định (validation) và kiểm định (verification) không giống nhau
  - Thẩm định chứng tỏ rằng chương trình thỏa mãn nhu cầu khách hàng
  - Kiểm định chứng tỏ rằng chương trình tuân theo đặc tả
- Cần thiết kế các kế hoạch test để định hướng cho quy trình kiểm thử
- Các kĩ thuật kiểm thử tĩnh đòi hỏi kiểm tra và phân tích chương trình để phát hiện lỗi (không chạy chương trình)
- Program inspection rất hiệu quả cho việc phát hiện lỗi
- Các công cụ phân tích tĩnh giúp phát hiện các bất thường mà có thể là dấu hiệu của lỗi trong mã nguồn

Bài tập về nhà

1. Phân biệt giữa thẩm định và kiểm định. Giải thích vì sao thẩm định lại là một quy trình đặc biệt khó?
2. Tại sao cần lập kế hoạch test đủ chi tiết để khi tiến hành test có thể thực hiện một cách máy móc và có hệ thống?
3. Lập một checklist gồm những dạng lỗi có thể xảy ra trong các script PHP trong ứng dụng Web+CSDL.
4. Giải thích tại sao tuy các phương pháp hình thức đòi hỏi chi phí cao nhưng khi áp dụng cho những hệ thống đòi hỏi độ an toàn cao thì vẫn được cho là giải pháp có hiệu quả về tài chính.