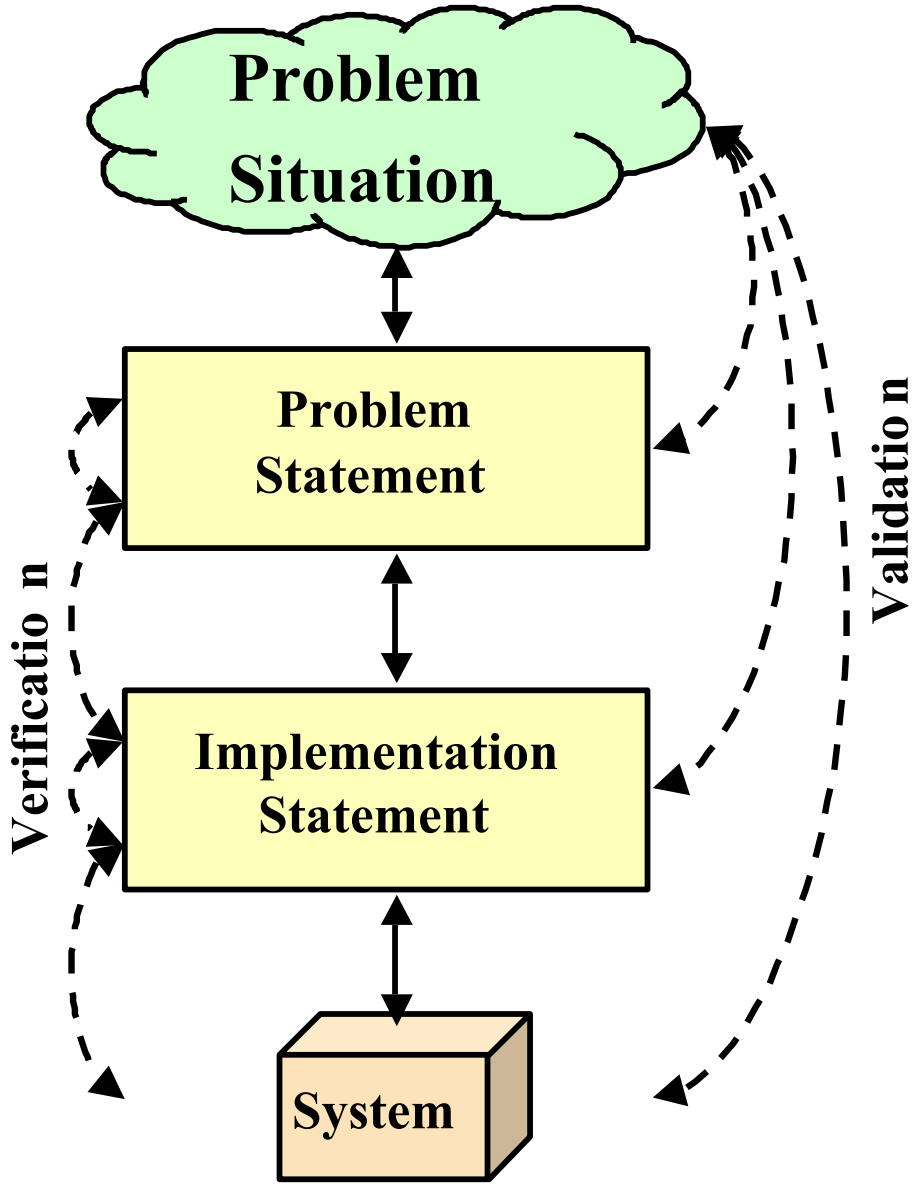
Lecture 12:
Kiểm tra và Kiểm chứng
(Verification and Validation)

- **Khái niệm:** Định nghĩa V&V
- **Các kỹ thuật kiểm chứng**
  - Lập bản mẫu (Prototyping)
  - Phân tích mô hình (Model Analysis) (e.g. Model Checking)
  - Kiểm duyệt (Inspection)
- **Các kỹ thuật kiểm tra (Verification Techniques)**
  - Thực hiện lưu vết đặc tả (Specifications Traceable) (Bài 19)
  - Kiểm thử (Testing)
  - Kiểm duyệt mã lệnh (Code Inspection)
  - Phân tích mã lệnh (Code analysis)
- **V&V độc lập**

Verification and Validation

- **Kiểm chứng (Validation)**
  - "Chúng ta đã xây dựng đúng hệ thống ?"
  - Khai báo vấn đề đã thực sự nắm bắt được vấn đề thực tế?
  - Hệ thống đã đáp ứng được nhu cầu của tất cả đối tác?
- **Kiểm tra (Verification)**
  - "Chúng ta đã xây dựng hệ thống đúng?"
  - Thiết kế đáp ứng đặc tả?
  - Cài đặc đáp ứng đặc tả?
  - Hệ thống được phân phối sẽ thực hiện điều mà nó phải làm?
  - Các mô hình yêu cầu thống nhất với những mô hình khác?



Khái niệm: Tiêu chuẩnV&V

Source: *Adapted from Jackson, 1995, p170-171*

- **Sự khác biệt:**
  - **Domain Properties: những điều luôn luôn đúng trong lĩnh vực ứng dụng**
  - **Requirements: những điều chúng ta mong là đúng trong lĩnh vực ứng dụng**
  - **Specification: sự mô tả các hành vi mà chương trình cần thực hiện để đáp ứng với các yêu cầu**
- **Hai tiêu chuẩn kiểm tra (verification)**
  - **Chương trình (Program) thực hiện trên một máy tính (Computer) cụ thể đáp ứng với đặc tả (Specification)**
  - **Đặc tả (Specification) được cho trong thuộc tính của lĩnh vực (Domain properties) thỏa mãn các yêu cầu (Requirements)**
- **Hai tiêu chuẩn kiểm chứng (validation)**
  - **Chúng ta đã xem xét (và hiểu) tất cả các yêu cầu (Requirements) quan trọng?**
  - **Chúng ta đã xem xét (và hiểu) tất cả các thuộc tính lĩnh vực(Domain properties) liên quan.**

Ví dụ V&V

Ví dụ

- **Requirement R:**
  - **"Phản lực chỉ có thể xảy ra khi máy bay đang chạy trên đường băng"**
- **Domain Properties D:**
  - **Xung lực bánh xe xảy ra khi và chỉ khi các bánh xe bật ra**
  - **Các bánh xe bật ra khi và chỉ khi nó chạy trên đường băng**
- **Specification S:**
  - **Phản lực có thể xảy ra khi và chỉ khi có xung lực bánh xe**

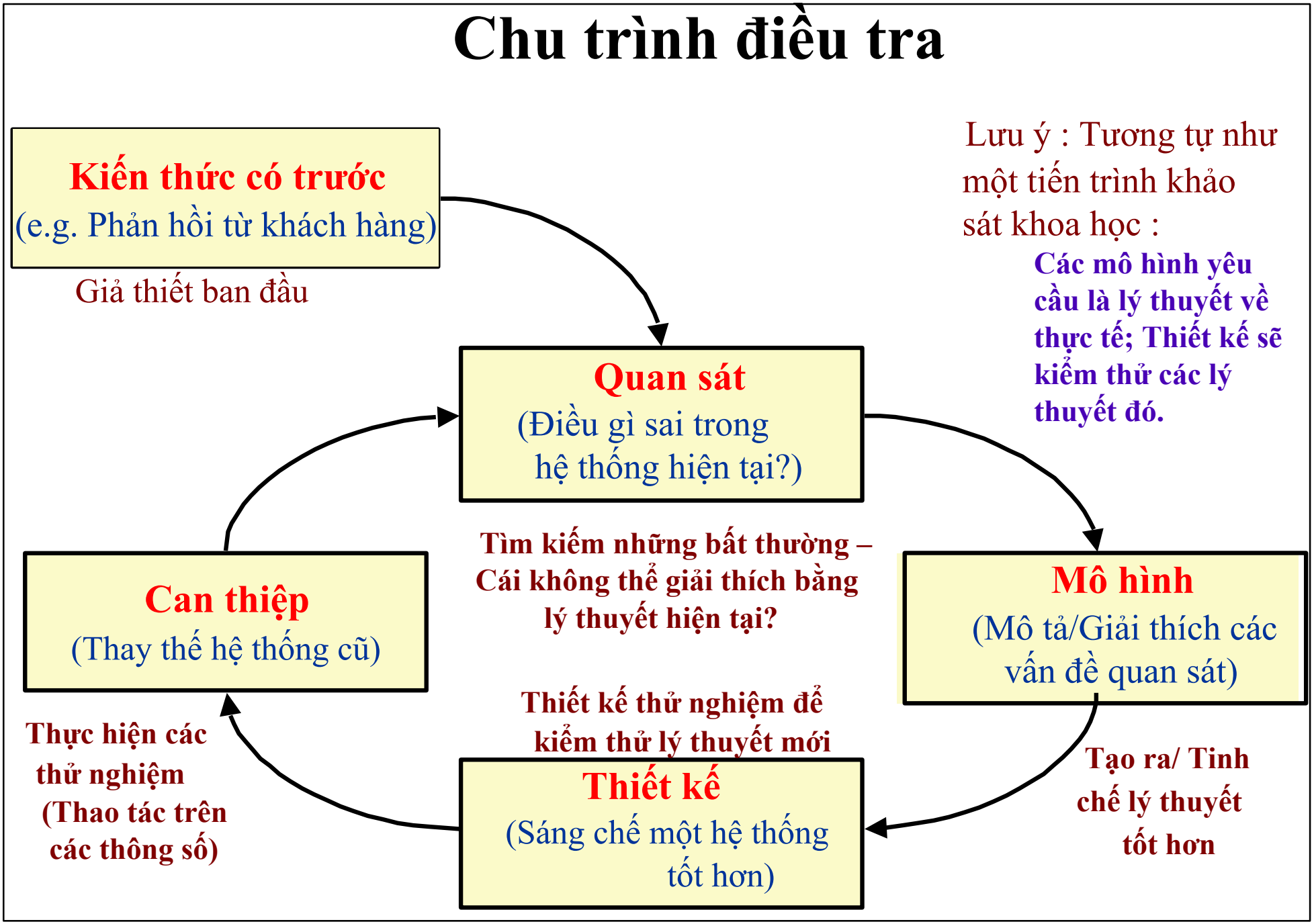
Kiểm tra

- **Phần mềm cho máy bay, P, thực thi trên máy tính trong buồng lái của máy bay, C, có hoàn toàn chính xác như đặc tả, S?**
- **S, trong ngữ cảnh của giả thuyết D, có đáp ứng R?**

Kiểm chứng

- **Giả thuyết của chúng ta, D, về lĩnh vực có thật chính xác? Có thiếu sót gì không?**
- **Yêu cầu, R, có thật sự cần thiết? Có thiếu sót gì không?**

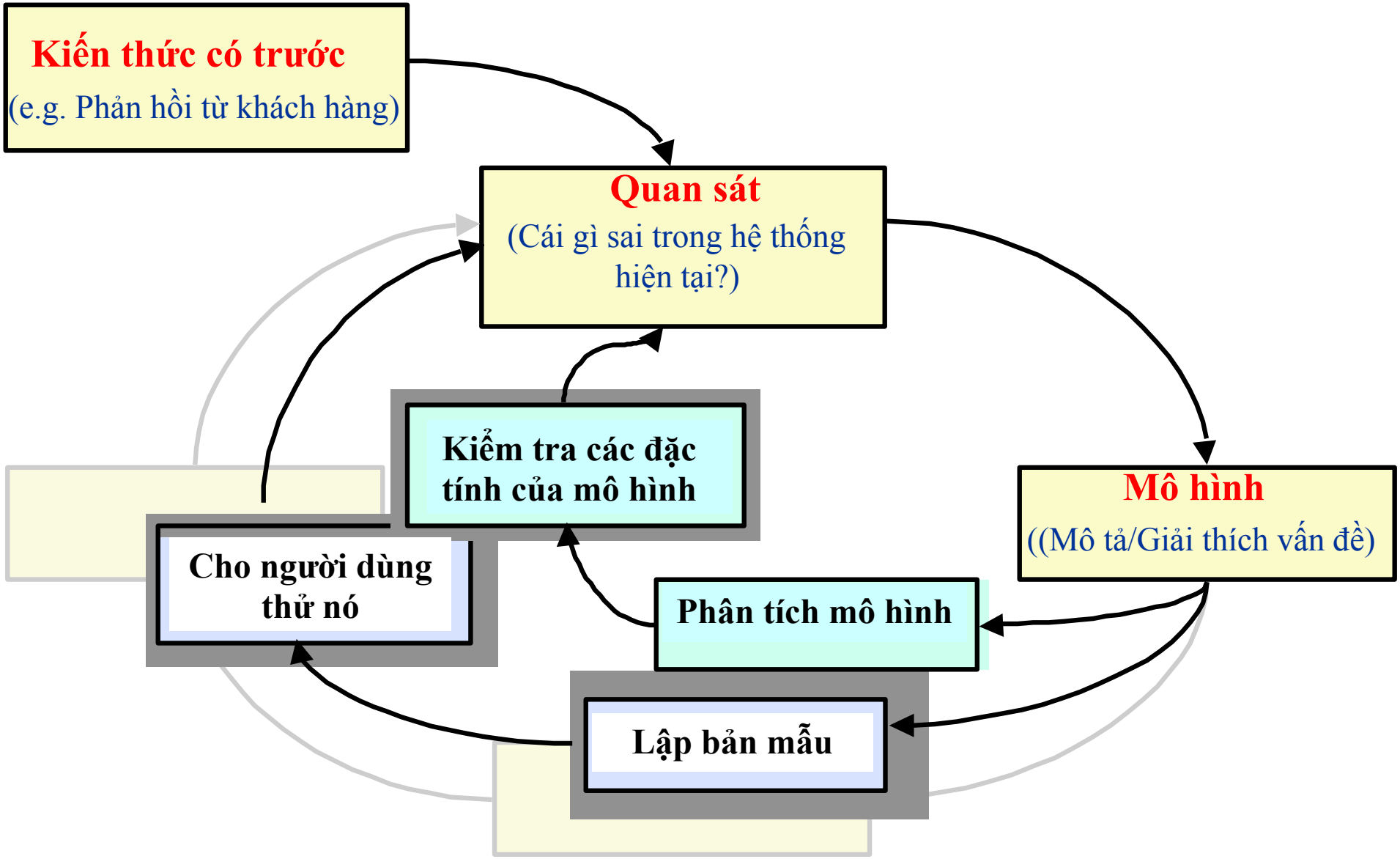
Chu trình điều tra



Lưu ý : Tương tự như một tiến trình khảo sát khoa học :

Các mô hình yêu cầu là lý thuyết về thực tế; Thiết kế sẽ kiểm thử các lý thuyết đó.

Chu trình điều tra nhanh



Lập bản mẫu

"Một bản mẫu phần mềm là một kiến trúc được cài đặt cụ thể trước để các khách hàng, người dùng hoặc nhà phát triển có thể hiểu rõ thêm về một vấn đề hay giải pháp của nó." [Davis 1990]

"Lập bản mẫu là tiến trình xây dựng mô hình làm việc của hệ thống"[Agresti 1986]

Hướng tiếp cận lập bản mẫu

- **Bản mẫu trình diễn**
  - Dùng để **chứng minh khái niệm**; giải thích các đặc tính thiết kế; etc.
  - Giải thích, minh họa và thông báo – sau đó bỏ đi
- **Bản mẫu thăm dò**
  - Dùng để **xác định vấn đề**, thu thập nhu cầu, làm rõ mục tiêu, so sánh các lựa chọn thiết kế
  - Không hình thức, không cấu trúc và cũng được bỏ đi.
- **Bản mẫu thử nghiệm**
  - Khai thác **các đặc tính kỹ thuật**; kiểm tra sự thích hợp của một kỹ thuật
  - Thông thường không bao gồm người dùng/khách hàng
- **Bản mẫu tiến triển (e.g. "bản mẫu vận hành", "hệ thống lái máy bay"):**
  - Được phát triển khi thấy tiến trình tiếp diễn sẽ tương thích với hệ thống
  - "Bản mẫu (Prototype)" như là một phân phối sớm, để tiếp tục cải tiến.

Dùng 1 lần hay tiếp tục phát triển ?

Bản mẫu dùng thử

- **Mục đích:**
  - để nghiên cứu nhiều hơn về vấn đề hoặc các giải pháp của nó…
  - bỏ đi khi đã thu được kiến thức mong muốn.
- **Cách dùng:** sớm hoặc trễ
- **Hướng tiếp cận:**
  - theo chiều ngang (horizontal) – thiết kế chỉ một tầng (e.g. UI)
  - "quick and dirty"
- **Thuận lợi:**
  - Phương tiên nghiên cứu cho sự hội tụ tốt hơn
  - Phân phối sớm → kiểm thử sớm → chi phí thấp
  - Thành công ngay cả khi nó bị thất bại!
- **Bất lợi:**
  - Nổ lực sẽ bị lãng phí nếu các yêu cầu thay đổi một cách nhanh chóng
  - Thường dùng thay thế một cách hợp lý các tài liệu yêu cầu
  - Có thể mong muốn của khách hàng là quá cao
  - Có thể phát triển thành sản phẩm cuối cùng

Bản mẫu tiến hóa

- **Mục đích**
  - để nghiên cứu nhiều hơn về vấn đề hoặc các giải pháp của nó…
  - …và giảm rủi ro bằng cách xây dựng sớm các bộ phận
- **Cách dùng:** phát triển dần; làm tiến hóa
- **Hướng tiếp cận:**
  - theo chiều dọc (vertical) – cài đặt từng phần của tất cả các tầng;
  - được thiết kế để mở rộng/thích ứng
- **Thuận lợi:**
  - Các yêu cầu không nhất thiết cố định
  - Trả về phiên bản sau cùng nếu phát hiện lỗi
  - Linh động(?)
- **Bất lợi:**
  - Có thể kết thúc với một hệ thống phức tạp, không cấu trúc – rất khó để bảo trì
  - Kiến trúc lựa chọn sớm có thể kém
  - Các giải pháp tối ưu thì không được bảo đảm
  - Thiếu kiểm soát và phương hướng

Brooks: "Kế hoạch để bỏ đi một bản mẫu – sẽ luôn luôn làm"

Phân tích mô hình (Model Analysis)

- **Kiểm tra:**
  - "Mô hình có định dạng chuẩn?"
  - Các thành phần trong mô hình thống nhất với những cái khác?
- **Kiểm chứng:**
  - Dựng hoạt cảnh của mô hình trên các ví dụ nhỏ
  - Các thay đổi hình thức:
    - "liệu mô hình vẫn đúng khi những đặc tính sau uld hold..."
  - Các câu hỏi 'Cái gì sẽ xảy ra nếu …?' ('What if'):
    - nguyên nhân về hậu quả của những yêu cầu ngoại lệ;
    - nguyên nhân về sự tác động của những thay đổi có thể
    - "hệ thống có khi nào thực hiện như sau ..."
  - Khảo sát trạng thái
    - E.g. sử dụng kiểm tra mô hình để phát hiện ra sự đáp ứng của một số đặc tính

Cơ sở kiểm tra chéo với UML

Biểu đồ Use Case

- Mỗi use case có một người dùng?
  - Mỗi người dùng có ít nhất một use case?
- Có tài liệu cho mỗi use case không?
  - Sử dụng sơ đồ trình tự hoặc tương tự

Biểu đồ lớp (Class Diagrams)

- Sơ đồ lớp có nắm bắt được tất cả các lớp được đề cập đến trong những sơ đồ khác không?
- Mỗi lớp có các phương pháp để lấy/đặt các thuộc tính của nó không?

Biểu đồ trình tự (Sequence Diagrams)

- Mỗi lớp có trong sơ đồ lớp không?
- Mỗi thông điệp có thể được gửi đi không?
  - Có một quan hệ kết hợp lớp người gửi và lớp người nhận trong biểu đồ lớp không?
  - Có một phương pháp gọi trong lớp gửi cho mỗi thông điệp gửi không?
  - Có một phương pháp gọi trong lớp nhận cho mỗi thông điệp nhận không?

Biểu đồ chuyển trạng (StateChart)

- Mỗi biểu đồ chuyển trạng có nắm bắt được (các trạng thái của) một lớp không?
  - Lớp đó có trong biểu đồ lớp không?
- Mỗi bước chuyển (transition) có một sự kiện kích hoạt (trigger event) không?
  - Có rõ ràng đối tượng nào khởi tạo mỗi sự kiện?
  - Mỗi sự kiện có được liệt kê như một phương thức thuộc lớp của đối tượng đó trong sơ đồ lớp?
- Mỗi trạng thái có biểu diễn một sự kết hợp phân biệt của các giá trị thuộc tính?
  - Có rõ ràng sự kết hợp nào với những giá trị thuộc tính?
  - Tất cả những thuộc tính đó được chỉ ra trong biểu đồ lớp?
- Có phương pháp gọi trong biểu đồ lớp cho mỗi bước chuyển không?
  - …một phương pháp gọi sẽ cập nhật các giá trị thuộc tính cho một trạng thái mới?
  - …phương pháp gọi sẽ kiểm tra mọi điều kiện trên bước chuyển?
  - …phương pháp gọi sẽ thực hiện mọi hoạt động của bước chuyển?

Reviews, Walkthroughs, Inspections…

- **"Management reviews"**
  - E.g. preliminary design review (PDR), critical design review (CDR), …
  - Dùng để nêu ra sự chắc chắn mà thiết kế báo hiệu
  - Tham gia bởi nhà quản trị và các nhà bảo trợ (khách hàng)
  - Thường chỉ là một "dog-and-pony show"
- **"Walkthroughs"**
  - Kỹ thuật của người phát triển (thường là không hình thức)
  - Được sử dụng bởi đội ngũ phát triển để cải tiến chất lượng sản phẩm
  - Tập trung vào việc tìm kiếm khuyết điểm
- **"(Fagan) Inspections"**
  - Một công cụ quản lý tiến trình (luôn hình thức)
  - Dùng để cải tiến chất lượng của tiến trình phát triển
  - Thu thập dữ liệu lỗi để phân tích chất lượng của tiến trình
  - Viết output thì quan trọng
  - Đóng vai trò chính trong việc huấn luyện đội ngũ kế thừa và truyền tải kinh nghiệm

- **Các định nghĩa này thì không được thống nhất chung!**
  - **Các thuật ngữ khác đã dùng:**
    - Formal Technical Reviews (FTRs)
    - Formal Inspections
- **"Cách thức" có thể đa dạng:**
  - **Không hình thức:**
    - Gặp gỡ (over coffee),
    - Họp mặt nhóm thông thường,
    - etc.
  - **Hình thức:**
    - Lập lịch hội thảo,
    - Chuẩn bị thành viên tham dự,
    - Xác định thời gian,
    - Mô tả cách thức,
    - Lập tài liệu kết quả

Lợi ích của kiểm duyệt hình thức (formal inspection)

Source: *Adapted from Blum, 1992, Freedman and Weinberg, 1990, & notes from Philip Johnson.*

- **Formal inspection tác động tốt đối với việc lập trình:**
  - **Đối với lập trình ứng dụng:**
    - **Hiệu quả hơn kiểm thử**
    - **Hầu hết chương trình reviewed thực hiện chính xác lần đầu tiên**
    - **So sánh: 10-50 lần thử kiểm tra/gặp lỗi**
  - **Dữ liệu từ các dự án lớn**
    - **Giảm lỗi bởi một nguyên nhân : 5; (10 trong một số báo cáo)**
    - **Cải thiện hiệu quả : 14% đến 25%**
    - **Phần trăm lỗi đươc phát hiện bởi kiểm duyệt: 58% đến 82%**
    - **Giảm chi phí 50%-80% cho V&V (bao gồm cả chi phí kiểm duyệt)**
  - **Hiệu quả trên thu nhập của nhân viên:**
    - **Tăng tinh thần, giảm thay đổi nhân sự**
    - **Lập lịch biểu và đánh giá tốt hơn (nhiều kiến thức về sơ lược các khuyết điểm)**
    - **Quản lý tốt hơn việc nhận định năng lực của nhân viên**

- **Các lợi ích này cũng ứng dụng vào kiểm duyệt yêu cầu**
  - **Nhiều khảo sát theo lối kinh nghiệm phát hiện rằng các tiến trình kiểm duyệt rất đa dạng**
  - **Các kết quả hòa lẫn trong các lợi ích liên quan của các tiến trình khác nhau**

Các vai trò (Roles)

Source: *Adapted from Blum, 1992, pp369-373*

Formal Walkthrough

- **Lãnh đạo Review**
  - Chủ trì cuộc họp
  - Bảo đảm các sự chuẩn bị
  - Giữ sự tập trung review
  - Báo cáo kết quả
- **Người ghi chép**
  - Giữ vết của các kết quả công bố
- **Reader**
  - Tổng kết các mẫu sản phẩm bởi các mẫu trong suốt quá trình review
- **Tác giả**
  - Người tham dự một cách tích cực (e.g. như reader)
- **Các Reviewers khác**
  - Công việc là tìm và viết ra báo cáo

Fagan Inspection

- **Người điều tiết (Moderator)**
  - Phải là một lập trình viên thành thạo
  - Cần được huấn luyện đặc biệt
  - Có thể đến từ dự án khác
- **Người thiết kế**
  - Các lập trình viên – người phát sinh thiết kế thông qua kiểm duyệt
- **Người viết code/cài đặt**
  - Các lập trình viên có trách nhiệm dịch từ thiết kế sang mã lệnh
- **Người kiểm thử**
  - Người có trách nhiệm viết/thực thi các tình huống kiểm thử

Lập cấu trúc sự kiểm duyệt

- **Checklist**
  - Dùng 1 danh sách các câu hỏi kiểm tra/vấn đề
  - review được cấu trúc bởi vấn đề trong danh sách
- **Walkthough**
  - Một người phải có mặt trong từng bước kiểm tra sản phẩm
  - review được cấu trúc bởi sản phẩm
- **Round Robin**
  - Mỗi reviewer lần lượt nếu ra một vấn đề
  - review thì được cấu trúc bởi đội review
- **Tốc độ Review**
  - Mỗi reviewer có 3 phút để xem xét lại 1 vấn đề, sau đó chuyển sang cho người kế tiếp
  - Tốt cho việc đánh giá khả năng hiểu thấu vấn đề!

Tại sao dùng sự kiểm duyệt?

- **Sự kiểm duyệt thì rất hiệu quả**
  - Kiểm duyệt **mã lệnh** thì tốt hơn việc kiểm thử để tìm khuyết điểm
  - Đối với **đặc tả (Specifications)**, kiểm duyệt thì đã có sẵn (bạn không thể "kiểm thử" một đặc tả!)
- **Các khóa:**
  - Chuẩn bị: những người kiểm duyệt thì thực hiện riêng lẻ trước
  - Tập hợp họp mặt: người kiểm duyệt họp lại để kết hợp các danh sách khuyết điểm lại với nhau
  - Phân tích mỗi khuyết điểm, nhưng không tốn thời gian cố gắng sửa nó
  - Buổi họp đóng một vai trò quan trọng:
    - Người kiểm duyệt học hỏi từ người khác khi họ so sánh danh sách của họ
    - Bổ sung thêm những khuyết điểm chưa thấy được
  - Các hồ sơ có khuyết điểm trong khâu kiểm duyệt thì rất quan trọng cho tiến trình cải tiến
- **Mở rộng việc lựa chọn các kỹ thuật kiểm duyệt:**
  - Đóng vai trò gì trong buổi họp?
  - Cấu trúc buổi họp như thế nào?
  - Loại danh sách kiểm tra nào được dùng?

V&V độc lập

- **V&V được thực hiện bởi một nhà thầu riêng**
  - Để đáp ứng V&V độc lập thì cần một quan điểm kỹ thuật độc lập.
  - Chi phí khoảng giữa 5% đến 15% của chi phí phát triển
  - Các khảo sát chỉ ra tăng gấp 5 lần lợi nhuận trên vốn đầu tư:
    - Lỗi phát hiện sớm, rẻ hơn để sửa, rẻ hơn để thử lại
    - Đặc tả rõ hơn
    - Các nhà phát triển thích dùng các thực nghiệm tốt hơn.
- **3 kiểu của sự độc lập:**
  - Độc lập quản lý:
    - Phân chia trách nhiện từ việc phát triển các phần mềm
    - Có thể quyết định khi nào và ở đâu cần tập trung nổ lực V&V
  - Độc lập tài chánh:
    - Phân chia chi phí và nguồn tài chính
    - Không mạo hiểm phân phối nguồn tài nguyên khi sắp sửa gặp khó khăn
  - Độc lập kỹ thuật:
    - Tổ chưc nhân sự riêng biệt, để tránh thành kiến của các nhà phân tích
    - Dùng những công cụ và kỹ thuật khác nhau

Kết luận

- **Kiểm chứng** (Validation) bạn đã giải quyết đúng vấn đề.
  - Prototyping – cho phản hồi của khách hàng sớm
  - Inspection – các chuyên gia lĩnh vực đọc đặc tả một cách kỹ lưỡng
  - Formal Analysis – phân tích toán học các mô hình của bạn
  - …cộng thêm các buổi họp & giao tiếp thường xuyên với các đối tác
- **Kiểm tra** (Verification) các bước trong công nghệ là đúng
  - Kiểm tra tính nhất quán – các mô hình có thống nhất với cái khác?
  - Khả năng lưu vết – làm cho việc thiết kế/viết mã lệnh/kiểm thử phản ánh đúng các yêu cầu?
- Sử dụng V&V thích hợp :
  - Lấy thông tin phản hồi từ khách hàng sớm nếu mô hình mới chỉ là phác thảo.
  - Phân tích và kiểm tra tính nhất quán nếu mô hình đã là bản đặc tả
  - V&V độc lập nếu hệ thống cần an toàn nghiêm ngặt.