Vòng Đời và Các Mô Hình Phát Triển Phần Mềm

Công Nghệ Phần Mềm Nâng Cao

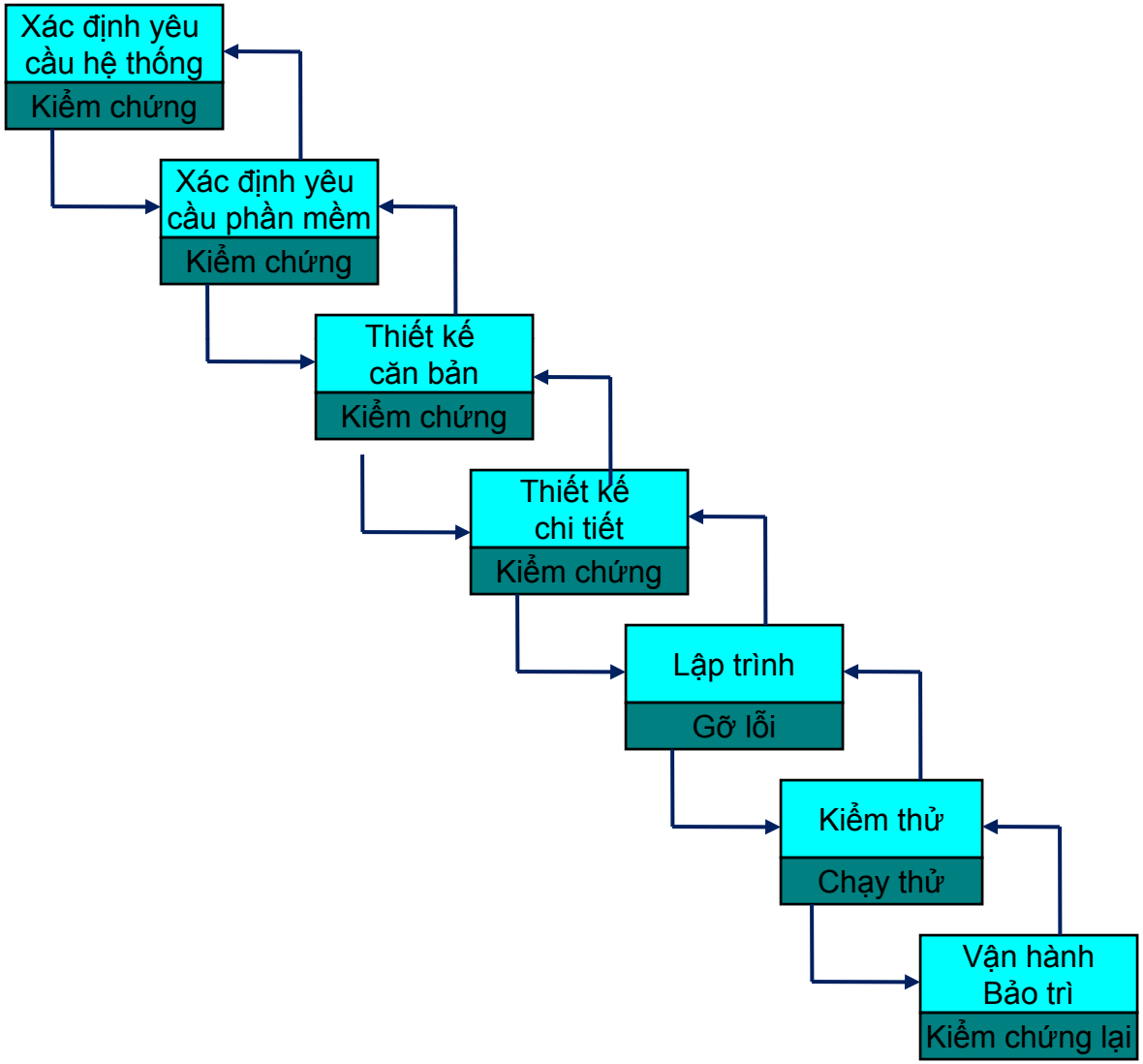
Outline

- Software life-cycle
- Qui trình phát triển Phần mềm
- Các mô hình phát triển
  - Mô hình tuyến tính
  - Mô hình chế thử
  - Mô hình phát triển ứng dụng nhanh
  - Các mô hình tiến hóa
  - Mô hình phát triển đồng thời
  - Mô hình hướng thành phần

Vòng đời phần mềm (Software life-cycle)

- Vòng đời phần mềm là thời kỳ tính từ khi phần mềm được sinh (tạo) ra cho đến khi chết đi (từ lúc hình thành đáp ứng yêu cầu, vận hành, bảo dưỡng cho đến khi loại bỏ không đâu dùng)
- Quy trình phần mềm (vòng đời phần mềm) được phân chia thành các pha chính: phân tích, thiết kế, chế tạo, kiểm thử, bảo trì. Biểu diễn các pha có khác nhau theo từng người

Mô hình vòng đời phần mềm của Boehm



Suy nghĩ mới về vòng đời phần mềm

(1) Pha xác định yêu cầu và thiết kế có vai trò quyết định đến chất lượng phần mềm, chiếm phần lớn công sức so với lập trình, kiểm thử và chuyển giao phần mềm

(2) Pha cụ thể hóa cấu trúc phần mềm phụ thuộc nhiều vào suy nghĩ trên xuống (top-down) và trừu tượng hóa, cũng như chi tiết hóa

(3) Pha thiết kế, chế tạo thì theo trên xuống, pha kiểm thử thì dưới lên (bottom-up)

Suy nghĩ mới về vòng đời phần mềm

(4) Trước khi chuyển sang pha kế tiếp phải đảm bảo pha hiện nay đã được kiểm thử không còn lỗi

(5) Cần có cơ chế kiểm tra chất lượng, xét duyệt giữa các pha nhằm đảm bảo không gây lỗi cho pha sau

(6) Tư liệu của mỗi pha không chỉ dùng cho pha sau, mà chính là đối tượng quan trọng cho kiểm tra và đảm bảo chất lượng của từng quy trình và của chính phần mềm

Suy nghĩ mới về vòng đời phần mềm

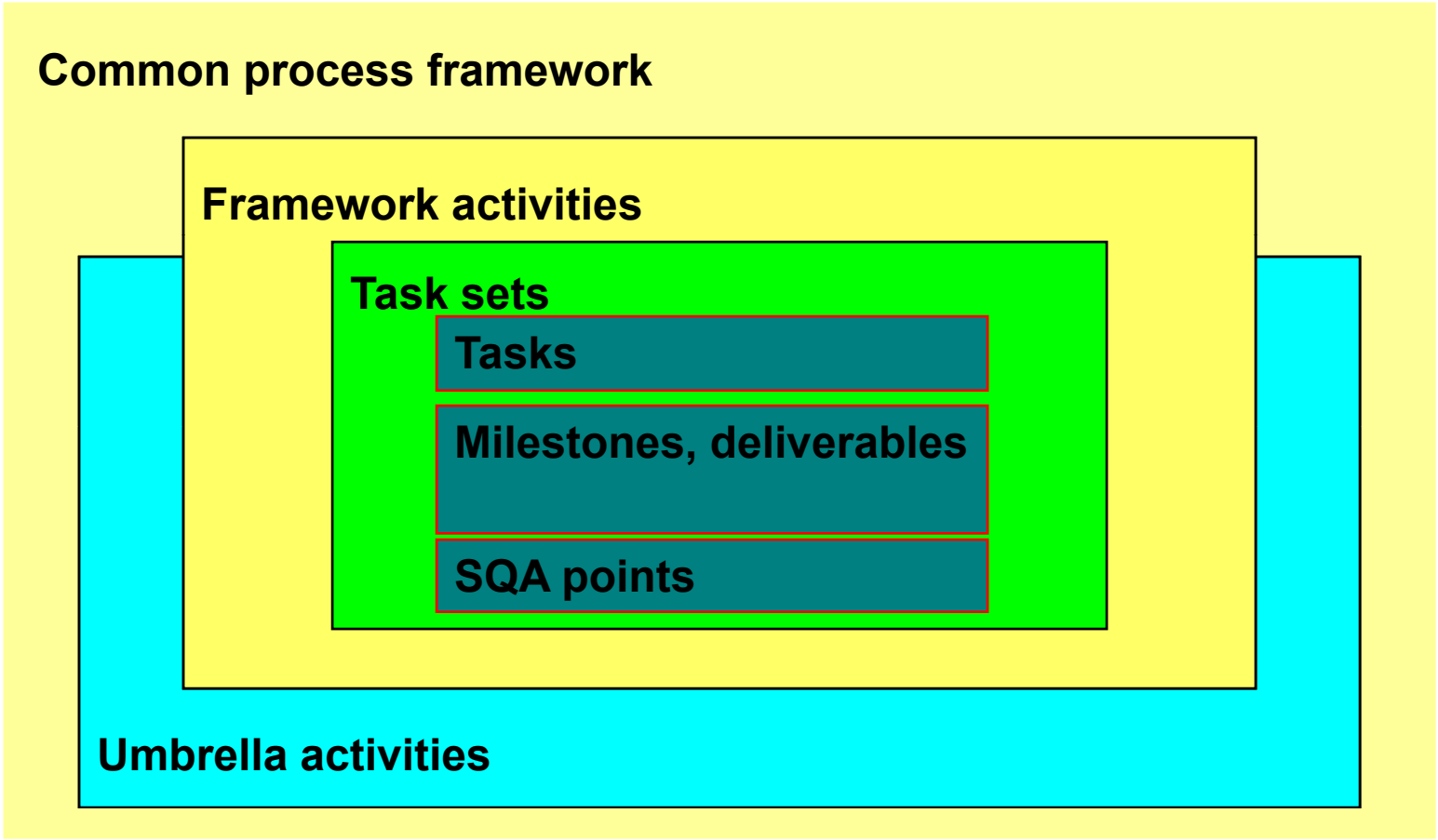
(7) Cần chuẩn hóa mẫu biểu, cách ghi chép tạo tư liệu cho từng pha, nhằm đảm bảo chất lượng phần mềm

(8) Thao tác bảo trì phần mềm là việc xử lý quay vòng trở lại các pha trong vòng đời phần mềm nhằm biến đổi, sửa chữa, nâng cấp phần mềm

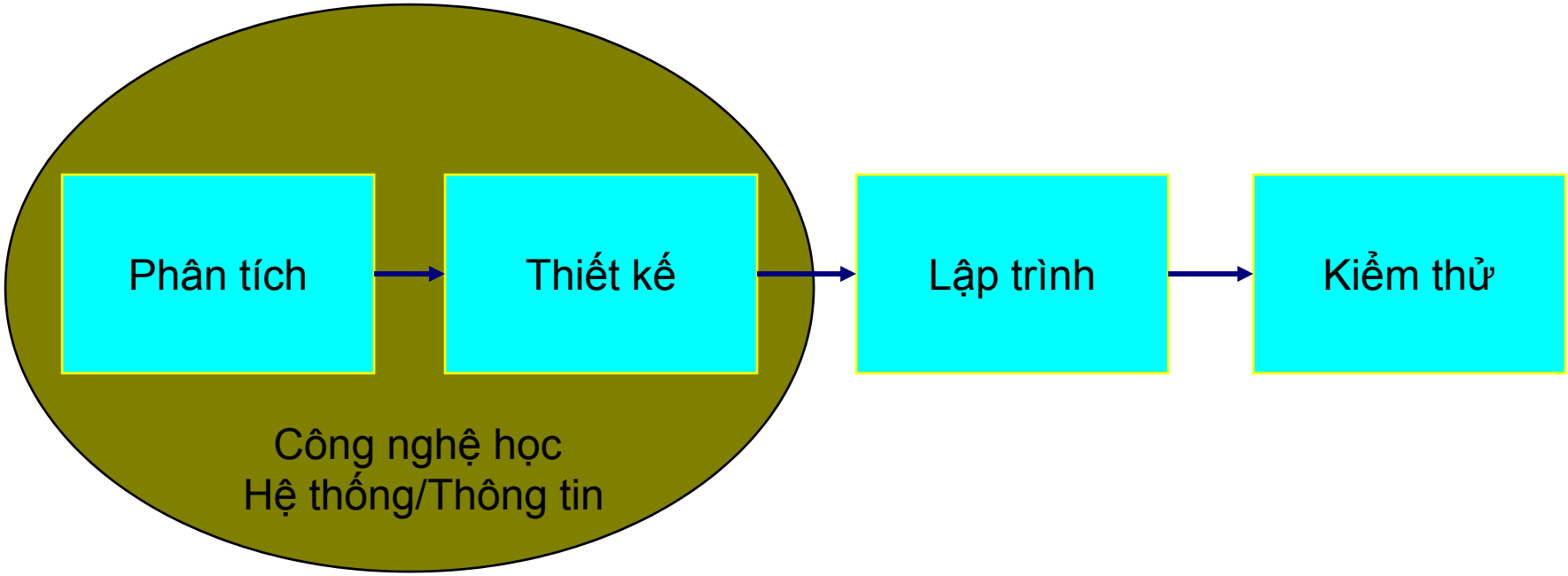
Các phương pháp luận và kỹ thuật cho từng pha

Tên pha	Nội dung nghiệp vụ	Phương pháp, kỹ thuật
Xác định yêu cầu	Đặc tả yêu cầu người dùng Xác định yêu cầu phần mềm	Phân tích cấu trúc hóa
Thiết kế hệ thống	Thiết kế cơ bản phần mềm Thiết kế cấu trúc ngoài của phần mềm	Thiết kế cấu trúc hóa
Thiết kế chương trình	Là thiết kế chi tiết: Thiết kế cấu trúc bên trong của phần mềm (đơn vị chương trình hoặc môđun)	Lập trình cấu trúc Phương pháp Jackson Phương pháp Warnier
Lập trình	Mã hóa bởi ngôn ngữ lập trình	Mã hóa cấu trúc hóa
Đảm bảo chất lượng	Kiểm tra chất lượng phần mềm đã phát triển	Phương pháp kiểm thử chương trình
Vận hành Bảo trì	Sử dụng, vận hành phần mềm đã phát triển. Biến đổi, điều chỉnh phần mềm	Chưa cụ thể

Quy trình phát triển phần mềm



Mô hình tuyến tính



Điển hình là mô hình vòng đời cổ điển
(mô hình thác nước) Classic life cycle /
waterfall model: là mô hình hay đựoc dùng nhất

Mô hình tuyến tính

- Công nghệ học Hệ thống/Thông tin và mô hình hóa (System / Information engineering and modeling): thiết lập các yêu cầu, ánh xạ một số tập con các yêu cầu sang phần mềm trong quá trình tương tác giữa phần cứng, người và CSDL
- Phân tích yêu cầu (Requirements analysis): hiểu lĩnh vực thông tin, chức năng, hành vi, tính năng và giao diện của phần mềm sẽ phát triển. Cần phải tạo tư liệu và bàn thảo với khách hàng, người dùng

Mô hình tuyến tính

- Thiết kế (Design): là quá trình nhiều bước với 4 thuộc tính khác nhau của một chương trình: cấu trúc dữ liệu, kiến trúc phần mềm, biểu diễn giao diện và chi tiết thủ tục (thuật toán). Cần tư liệu hóa và là một phần quan trọng của cấu hình phần mềm
- Tạo mã / lập trình (Code generation/programming): Chuyển thiết kế thành chương trình máy tính bởi ngôn ngữ nào đó. Nếu thiết kế đã được chi tiết hóa thì lập trình có thể chỉ thuần túy cơ học

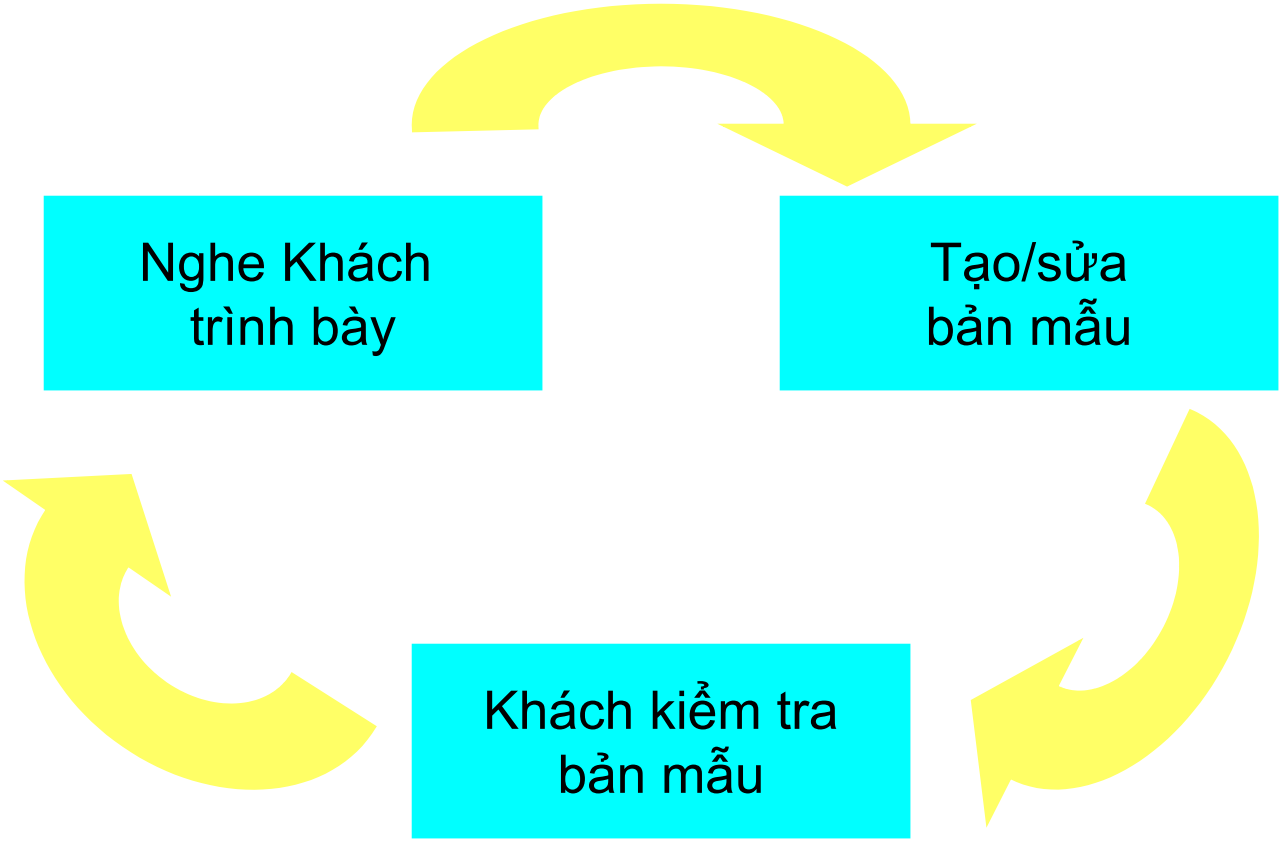
Mô hình tuyến tính

- Kiểm thử (Testing): Kiểm tra các chương trình và môđun cả về lôgic bên trong và chức năng bên ngoài, nhằm phát hiện ra lỗi và đảm bảo với đầu vào xác định thì cho kết quả mong muốn
- Hỗ trợ / Bảo trì (Support / Maintenance): Đáp ứng những thay đổi, nâng cấp phần mềm đã phát triển do sự thay đổi của môi trường, nhu cầu

Điểm yếu của Mô hình tuyến tính

- Thực tế các dự án ít khi tuân theo dòng tuần tự của mô hình, mà thường có lặp lại (như mô hình của Boehm)
- Khách hàng ít khi tuyên bố rõ ràng khi nào xong hết các yêu cầu
- Khách hàng phải có lòng kiên nhẫn chờ đợi thời gian nhất định mới có sản phẩm. Nếu phát hiện ra lỗi nặng thì là một thảm họa!

Mô hình chế thử (Prototyping model)



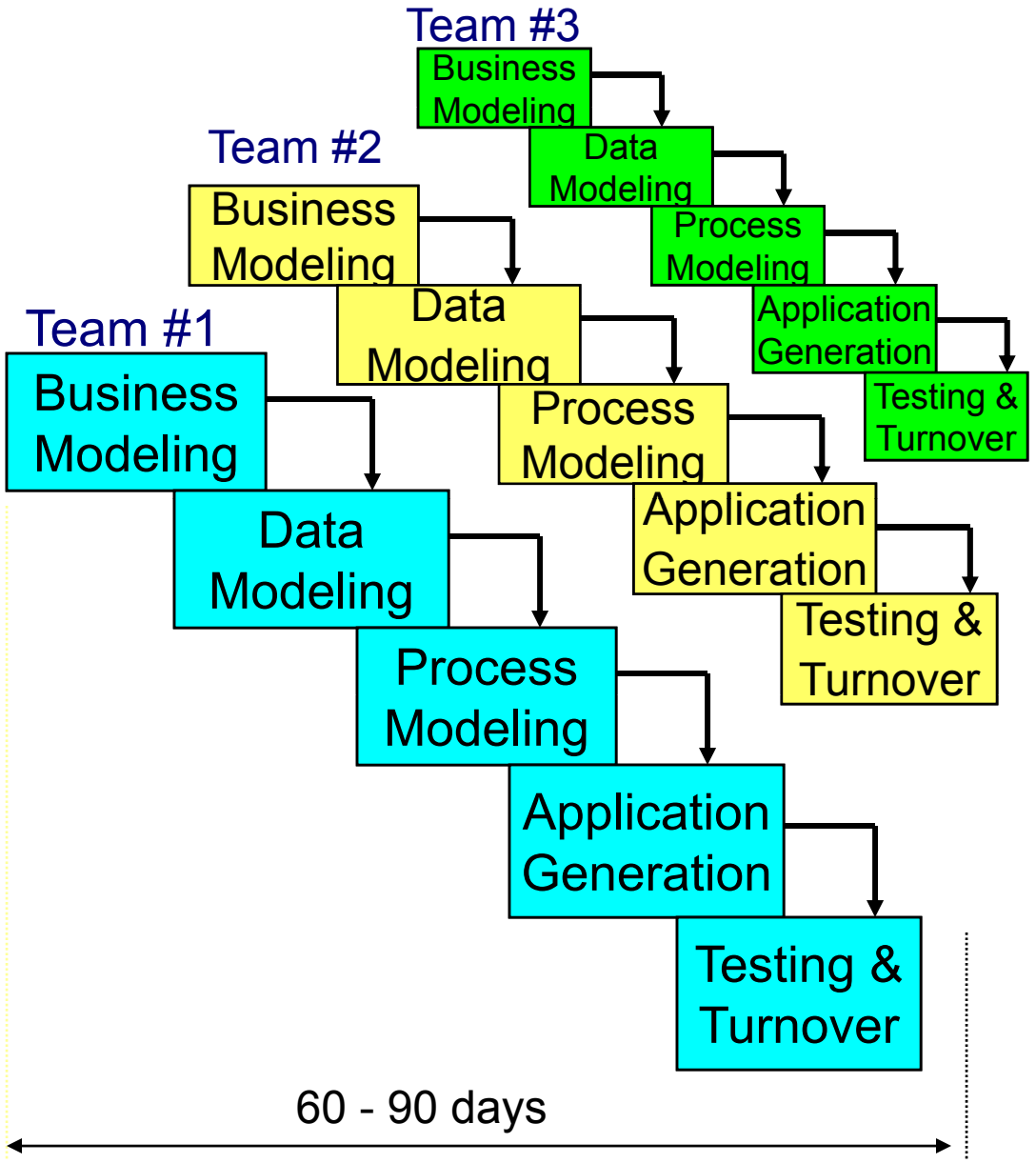
Mô hình chế thử: Khi nào?

- Khi mới rõ mục đích chung chung của phần mềm, chưa rõ chi tiết đầu vào hay xử lý ra sao hoặc chưa rõ yêu cầu đầu ra
- Dùng như "Hệ sơ khai" để thu thập yêu cầu người dùng qua các thiết kế nhanh
- Các giải thuật, kỹ thuật dùng làm bản mẫu có thể chưa nhanh, chưa tốt, miễn là có mẫu để thảo luận gợi yêu cầu của người dùng

Mô hình phát triển ứng dụng nhanh (Rapid Application Development: RAD)

- Là quy trình phát triển phần mềm gia tăng, tăng dần từng bước (Incrimental software development) với mỗi chu trình phát triển rất ngắn (60-90 ngày)
- Xây dựng dựa trên hướng thành phần (Component-based construction) với khả năng tái sử dụng (reuse)
- Gồm một số nhóm (teams), mỗi nhóm làm 1 RAD theo các pha: Mô hình nghiệp vụ, Mô hình dữ liệu, Mô hình xử lý, Tạo ứng dụng, Kiểm thử và đánh giá (Business, Data, Process, Appl. Generation, Test)

Mô hình phát triển ứng dụng nhanh



RAD: Business modeling

Luồng thông tin được mô hình hóa để trả lời các câu hỏi:

- Thông tin nào điều khiển xử lý nghiệp vụ?
- Thông tin gì được sinh ra?
- Ai sinh ra nó?
- Thông tin đi đến đâu?
- Ai xử lý chúng?

RAD: Data and Process modeling

- Data modeling: các đối tượng dữ liệu cần để hỗ trợ nghiệp vụ (business). Định nghĩa các thuộc tính của từng đối tượng và xác lập quan hệ giữa các đối tượng
- Process modeling: Các đối tượng dữ liệu được chuyển sang luồng thông tin thực hiện chức năng nghiệp vụ. Tạo mô tả xử lý để cập nhật (thêm, sửa, xóa, khôi phục) từng đối tượng dữ liệu

RAD: Appl. Generation and Testing

- Application Generation: Dùng các kỹ thuật thế hệ 4 để tạo phần mềm từ các thành phần có sẵn hoặc tạo ra các thành phần có thể tái dụng lại sau này. Dùng các công cụ tự động để xây dựng phần mềm
- Testing and Turnover: Kiểm thử các thành phần mới và kiểm chứng mọi giao diện (các thành phần cũ đã được kiểm thử và dùng lại)

RAD: Hạn chế?

- Cần nguồn nhân lực dồi dào để tạo các nhóm cho các chức năng chính
- Yêu cầu hai bên giao kèo trong thời gian ngắn phải có phần mềm hoàn chỉnh, thiếu trách nhiệm của một bên dễ làm dự án đổ vỡ
- RAD không phải tốt cho mọi ứng dụng, nhất là với ứng dụng không thể môđun hóa hoặc đòi hỏi tính năng cao
- Mạo hiểm kỹ thuật cao thì không nên dùng RAD

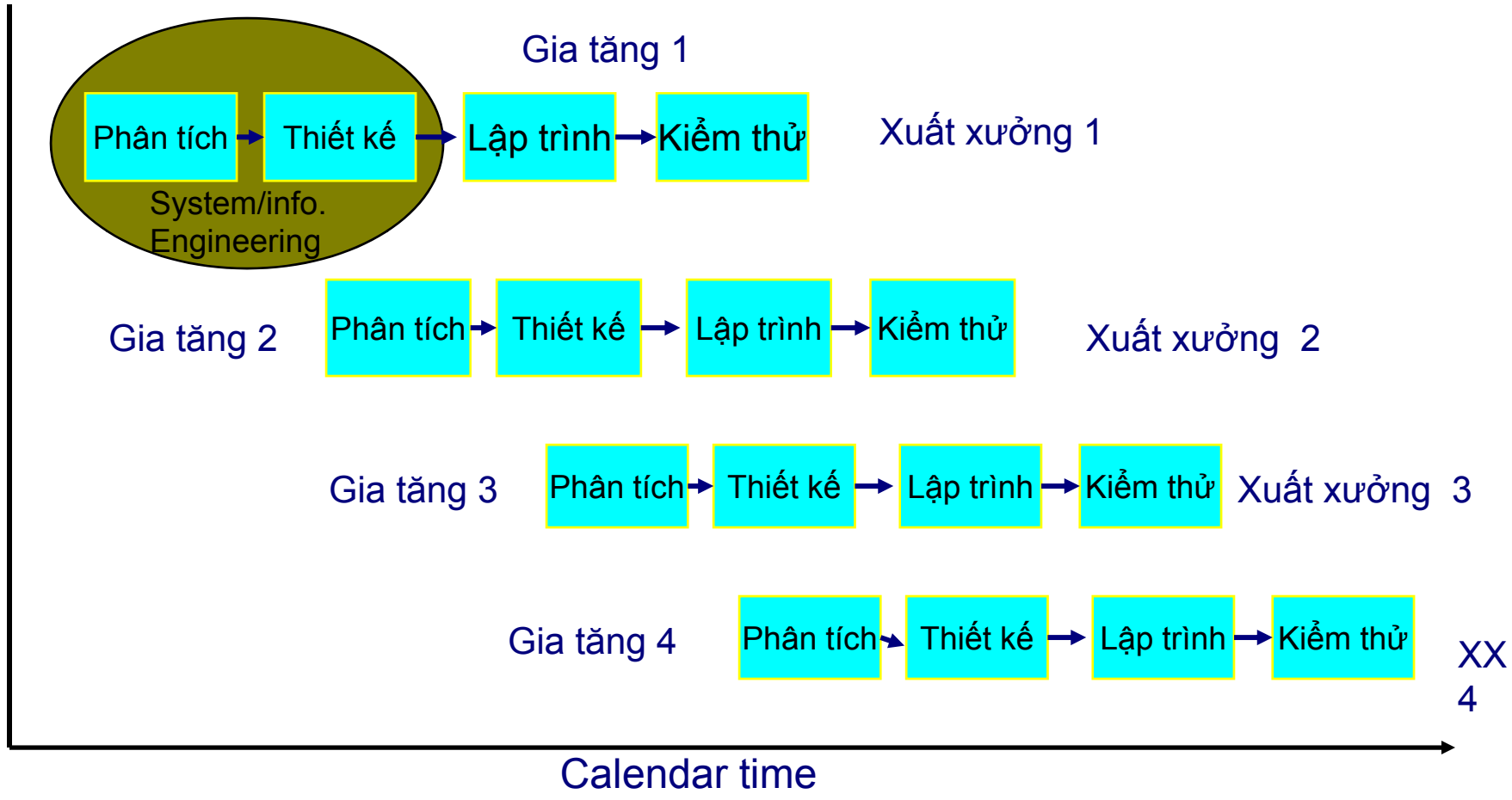
Các mô hình tiến hóa: gia tăng, xoắn ốc...

- Phần lớn các hệ phần mềm phức tạp đều tiến hóa theo thời gian: môi trường thay đổi, yêu cầu phát sinh thêm, hoàn thiện thêm chức năng, tính năng
- Các mô hình tiến hóa (evolutionary models) có tính lặp lại. Kỹ sư phần mềm tạo ra các phiên bản (versions) ngày càng hoàn thiện hơn, phức tạp hơn
- Các mô hình: incremental, spiral, WINWIN spiral, concurrent development model

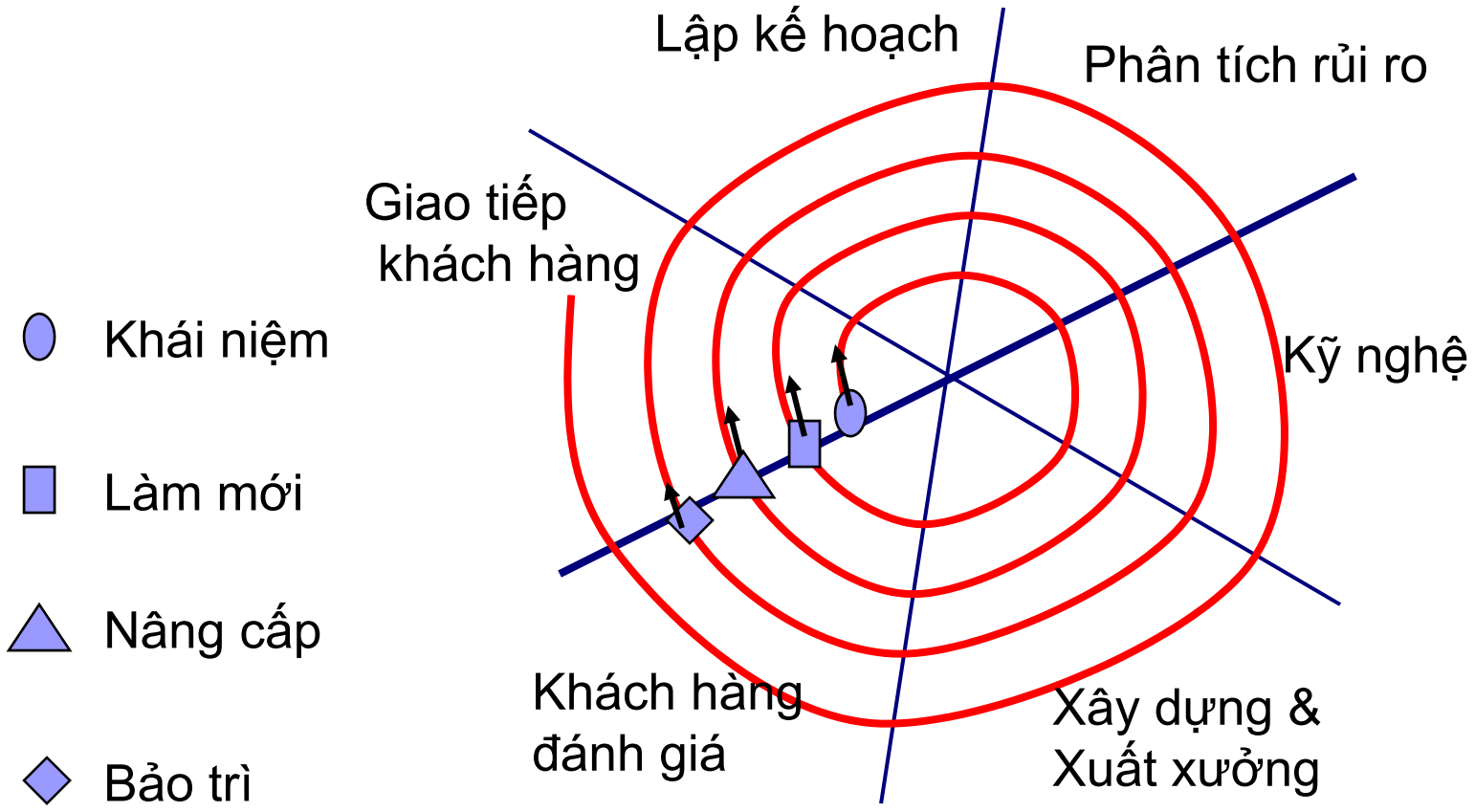
Mô hình gia tăng (The incremental model)

- Kết hợp mô hình tuần tự và ý tưởng lặp lại của chế bản mẫu
- Sản phẩm lõi với những yêu cầu cơ bản nhất của hệ thống được phát triển
- Các chức năng với những yêu cầu khác được phát triển thêm sau (gia tăng)
- Lặp lại quy trình để hoàn thiện dần

Mô hình gia tăng



Mô hình xoắn ốc (spiral)



Mô hình xoắn ốc (tiếp)

- Giao tiếp khách hàng: giữa người phát triển và khách hàng để tìm hiểu yêu cầu, ý kiến
- Lập kế hoạch: Xác lập tài nguyên, thời hạn và những thông tin khác
- Phân tích rủi ro: Xem xét mạo hiểm kỹ thuật và mạo hiểm quản lý
- Kỹ nghệ: Xây dựng một hay một số biểu diễn của ứng dụng

Mô hình xoắn ốc (tiếp)

- Xây dựng và xuất xưởng: xây dựng, kiểm thử, cài đặt và cung cấp hỗ trợ người dùng (tư liệu, huấn luyện, . . .)
- Đánh giá của khách hàng: Nhận các phản hồi của người sử dụng về biểu diễn phần mềm trong giai đoạn kỹ nghệ và cài đặt

Mô hình xoắn ốc: Mạnh và yếu?

- Tốt cho các hệ phần mềm quy mô lớn
- Dễ kiểm soát các mạo hiểm ở từng mức tiến hóa
- Khó thuyết phục khách hàng là phương pháp tiến hóa xoắn ốc có thể kiểm soát được
- Chưa được dùng rộng rãi như các mô hình tuyến tính hoặc chế thử

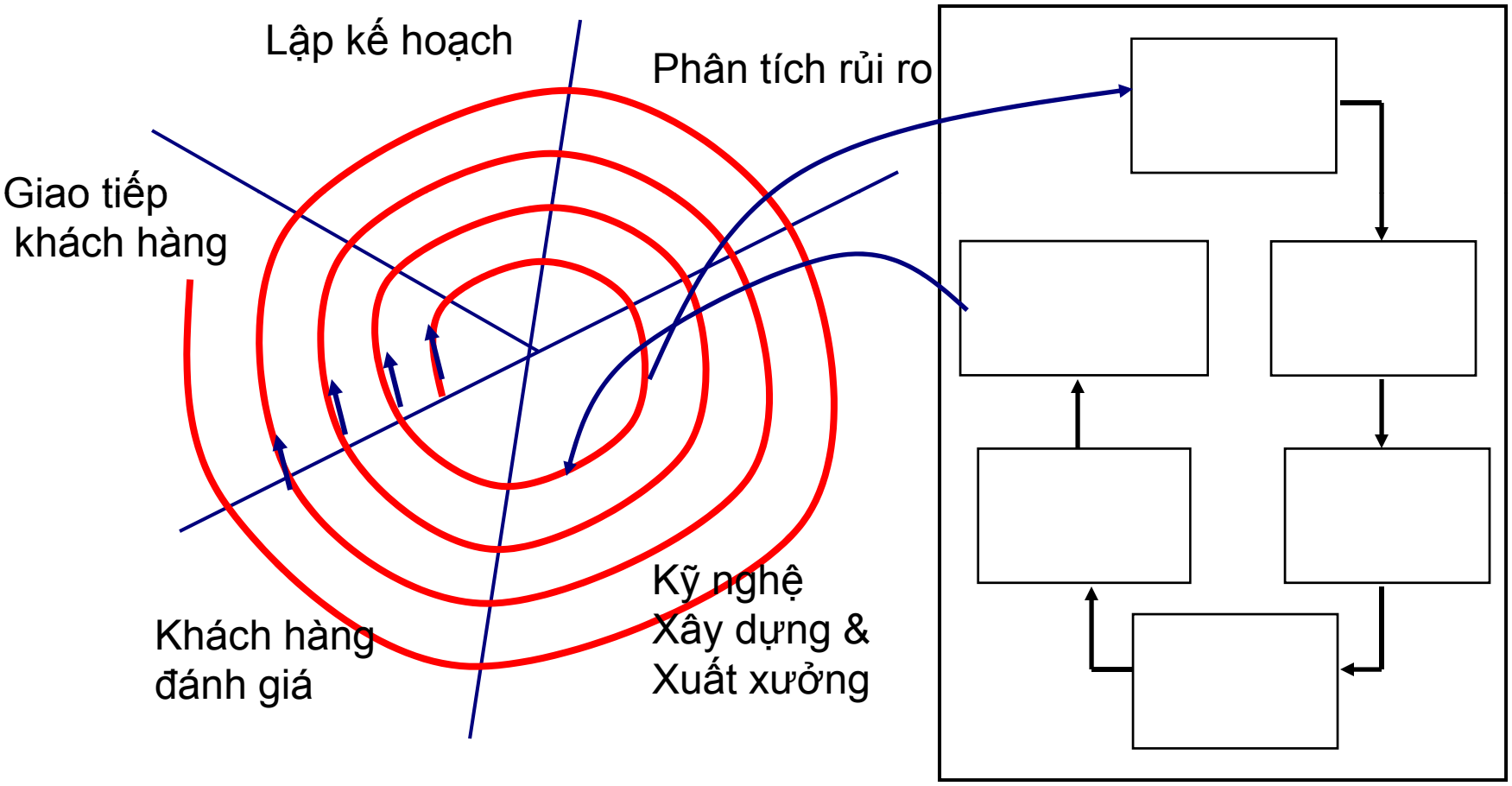
Mô hình phát triển đồng thời (The concurrent development model)

- Xác định mạng lưới những hoạt động đồng thời (Network of concurrent activities)
- Các sự kiện (events) xuất hiện theo điều kiện vận động trạng thái trong từng hoạt động
- Dùng cho mọi loại ứng dụng và cho hình ảnh khá chính xác về trạng thái hiện trạng của dự án
- Thường dùng trong phát triển các ứng dụng khách/chủ (client/server applications): system and componets are developed concurrently

Mô hình hướng thành phần (Component-based model)

- Gắn với những công nghệ hướng đối tượng (Object-oriented technologies) qua việc tạo các lớp (classes) có chứa cả dữ liệu và giải thuật xử lý dữ liệu
- Có nhiều tương đồng với mô hình xoắn ốc
- Với ưu điểm tái sử dụng các thành phần qua Thư viện / kho các lớp: tiết kiệm 70% thời gian, 80% giá thành, chỉ số sản xuất 26.2/16.9
- Với UML như chuẩn công nghiệp đang triển khai

Mô hình hướng thành phần



Mô hình hình thức (Formal model)

- Còn gọi là CNHPM phòng sạch (Cleanroom SE)
- Tập hợp các công cụ nhằm đặc tả toán học phần mềm máy tính từ khâu định nghĩa, phát triển đến kiểm chứng
- Giúp kỹ sư phần mềm phát hiện và sửa các lỗi khó
- Thường dùng trong phát triển SW cần độ an toàn rất cao (y tế, hàng không, . . .)

Mô hình hình thức: Điểm yếu?

- Cần nhiều thời gian và công sức để phát triển
- Phí đào tạo cao vì ít người có nền căn bản cho áp dụng mô hình hình thức
- Khó sử dụng rộng rãi vì cần kiến thức toán và kỹ năng của khách hàng

Outline

- Software life-cycle
- Qui trình phát triển Phần mềm
- Các mô hình phát triển
  - Mô hình tuyến tính
  - Mô hình chế thử
  - Mô hình phát triển ứng dụng nhanh
  - Các mô hình tiến hóa
  - Mô hình phát triển đồng thời
  - Mô hình hướng thành phần

Thảo luận

- Chọn mô hình nào cho phát triển PM?

Thực trạng của ngành Công nghiệp PM

- Hầu hết các tổ chức/cty PM đều tách biệt hoàn toàn pha phát triển và pha bảo trì
  - -> HT (sản phầm PM) khó bảo trì
  - -> Khó sử dụng lại các thành phần
  - Quản lý dự án vs. quản lý sản phẩm
  - Mở rộng mục tiêu của việc quản lý để bao quát được một họ các sản phẩm hơn là quản lý từng sản phẩm riêng biệt


Question?