NHẬP MÔN
CÔNG NGHỆ PHẦN MỀM
(INTRODUCTION TO SOFTWARE ENGINEERING)

Chương 2: Vòng đời phần mềm

- 1. Định nghĩa
- 2. Quy trình phát triển phần mềm
- 3. Một số mô hình phát triển phần mềm
  - 3.1. Mô hình CMM
  - 3.2. Mô hình tuyến tính
  - 3.3. Mô hình chế thử
  - 3.4. Mô hình phát triển ứng dụng nhanh
  - 3.5. Các mô hình tiến hóa
  - 3.6. Mô hình hướng thành phần
  - 3.7. Mô hình RUP
  - 3.8. Các kỹ thuật thế hệ thứ 4
- 4. Đánh giá sản phẩm và quy trình

Chương 2: Vòng đời phần mềm

2.1. Định nghĩa (Vòng đời phần mềm)

- 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

Vòng đời phần mềm

- Mọi sản phẩm phần mềm đều có vòng đời.
- Vòng đời thường khá dài — một số sản phẩm phần mềm đã “tồn tại” được 30 năm.
- Vòng đời có thể được rút ngắn do tiến bộ công nghệ

Các pha trong vòng đời PM

- Một cách rõ ràng hoặc rõ ràng, tất cả các sản phẩm phần mềm đều trải qua ít nhất các giai đoạn sau:
  - Yêu cầu — xác định nhu cầu của khách hàng và các ràng buộc của sản phẩm
  - Thiết kế — xác định cấu trúc / tổ chức của hệ thống phần mềm]
  - Mã hóa — viết phần mềm
  - Kiểm thử — vận hành hệ thống để tìm và loại bỏ các khiếm khuyết
  - Bảo trì — sửa chữa và nâng cao sản phẩm sau khi khách hàng triển khai

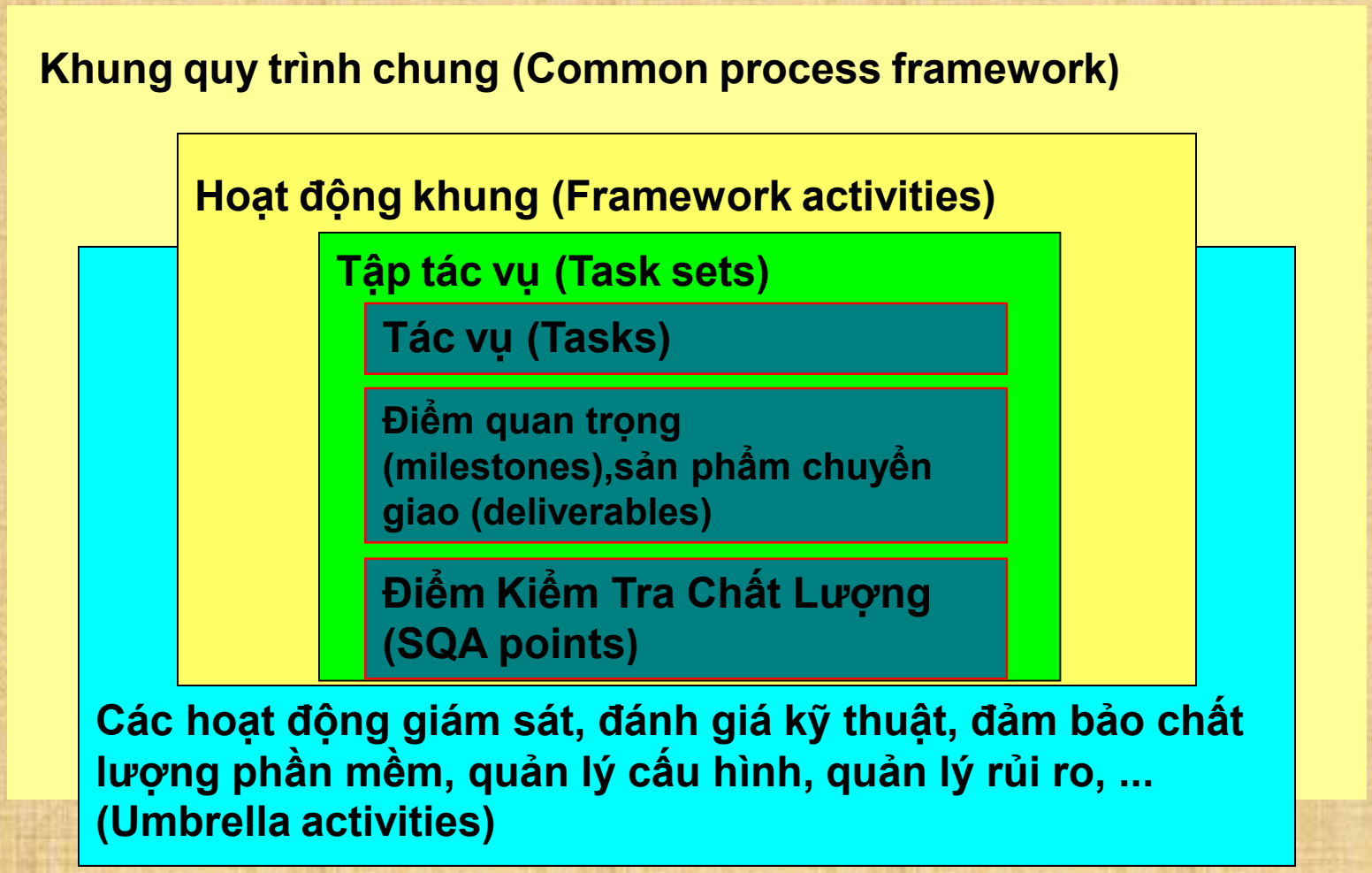
Các phương pháp luận và kỹ thuật cho từng pha

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

Các mô hình vòng đời phần mềm

- Quá trình là một tập hợp các hoạt động, với các đầu vào và đầu ra được xác định rõ ràng, để hoàn thành một số nhiệm vụ.
- Mô hình vòng đời là một mô tả về một quá trình thực hiện một sản phẩm phần mềm trong toàn bộ hoặc một phần vòng đời của nó.
  - Các mô hình vòng đời có xu hướng tập trung vào các pha chính của chu kỳ và mối quan hệ của chúng với các pha khác.
  - Các nghiên cứu gần đây về quy trình phần mềm đã xem xét chi tiết nhiều khía cạnh của việc phát triển và bảo trì.
  - Mô hình vòng đời là một mô tả quy trình phần mềm, nhưng thuật ngữ mô hình vòng đời có trước các cuộc thảo luận về quy trình phần mềm.

2. Quy trình phát triển phần mềm



3. Một số mô hình phát triển PM

3.1. Capability Maturity Model (CMM by SEI): Mô hình khả năng thuần thục

- Các khái niệm
- Tại sao sử dụng mô hình CMM
- Các thức sử dụng mô hình

3.1. Mô hình khả năng thuần thục

Tại sao phải sử dụng mô hình CMM trong công nghệ làm phần mềm?

- Khó khăn khi không sử dụng CMM
  - Các tiến trình phần mềm thường bị thay đổi cập nhật mà không có sự chuẩn bị trước.
  - Đặc tả một tiến trình phần mềm không chặt chẽ, dẫn đến sự khủng hoảng khi thực hiện một dự án.
  - Thiếu cơ sở để đánh giá chất lượng phần mềm, để đưa ra phương thức tiến hành và cách giải quyết các vấn đề phát sinh.
- Thuận lợi khi sử dụng CMM
  - Dễ dàng quản lý phát triển phần mềm. Các tiến trình được cập nhật qua sự điều khiển của các nhà phần tích và kiểm thử.
  - Vai trò, trách nhiệm của mỗi thành viên trong các tiến trình được phân định rõ ràng.
  - Quản lý chất lượng của phần mềm, thoả mãn các yêu cầu khách hàng. Có cơ sơ chuẩn xác đánh giá chất lượng, thời gian, chi phí và phân tích dự án và các tiến trình.

Các khái niệm cơ bản trong CMM

- Tiến trình (Process)
  - Một tiến trình phần mềm là một tập hợp các hành động, phương thức, thực hành, thay đổi mà người ta dùng để duy trì và phát triển phần mềm cũng như các thành phần liên quan tới chúng (ví dụ: kế hoạch dự án, thiết kế, lập trình, kiểm thử, tài liệu hướng dẫn...).
- Khả năng tiến trình phần mềm (Software Process Capability)
  - Cho biết phạm vi kết quả có thể mong đợi của một tiến trình phần mềm.
  - Dự đoán khả năng làm dự án phần mềm tiếp theo của công ty.

Thực thi tiến trình phần mềm
(Software Process Performance)

- Thực thi tiến trình phần mềm cho biết kết quả thực tế của một tiến trình phần mềm.
- Hướng tới kết quả đạt được còn khả năng tiến trình phần mềm cho thấy kết quả có thể mong đợi.
- Do phụ thuộc vào đặc trưng của dự án và từng trường hợp cụ thể, nên kết quả thực tế thường không phản ánh đầy đủ khả năng tiến trình của một công ty.

Độ thuần thục của tiến trình PM
(Software process maturity)

- Chỉ rõ một tiến trình phần mềm được xác định, quản lý, đánh giá, điều khiển, đạt hiệu quả một cách rõ ràng.
- Cho biết khả năng phát triển, chỉ ra giá trị của tiến trình phần mềm, tính vững chắc của dự án.

Mô hình chi tiết các thành phần trong cấu trúc CMM.

Mô hình 5 mức của CMM

Tiến trình phần mềm mang tính chất tuỳ tiện, lộn xộn, có ít tiến trình được xác định trước, hiệu quả của công việc mang tính riêng lẻ.
Khó có được một môi trường làm việc ổn định. Kế hoạch và ngân sách, chất lượng sản phẩm và vận hành không thể dự đoán trước được.

Quá trình vận hành phụ thuộc vào khả năng của từng cá nhân riêng lẻ, và thường xuyên thay đổi do phụ thuộc vào kỹ năng, trình độ hiểu biết và các hoạt động của từng thành viên trong dự án.

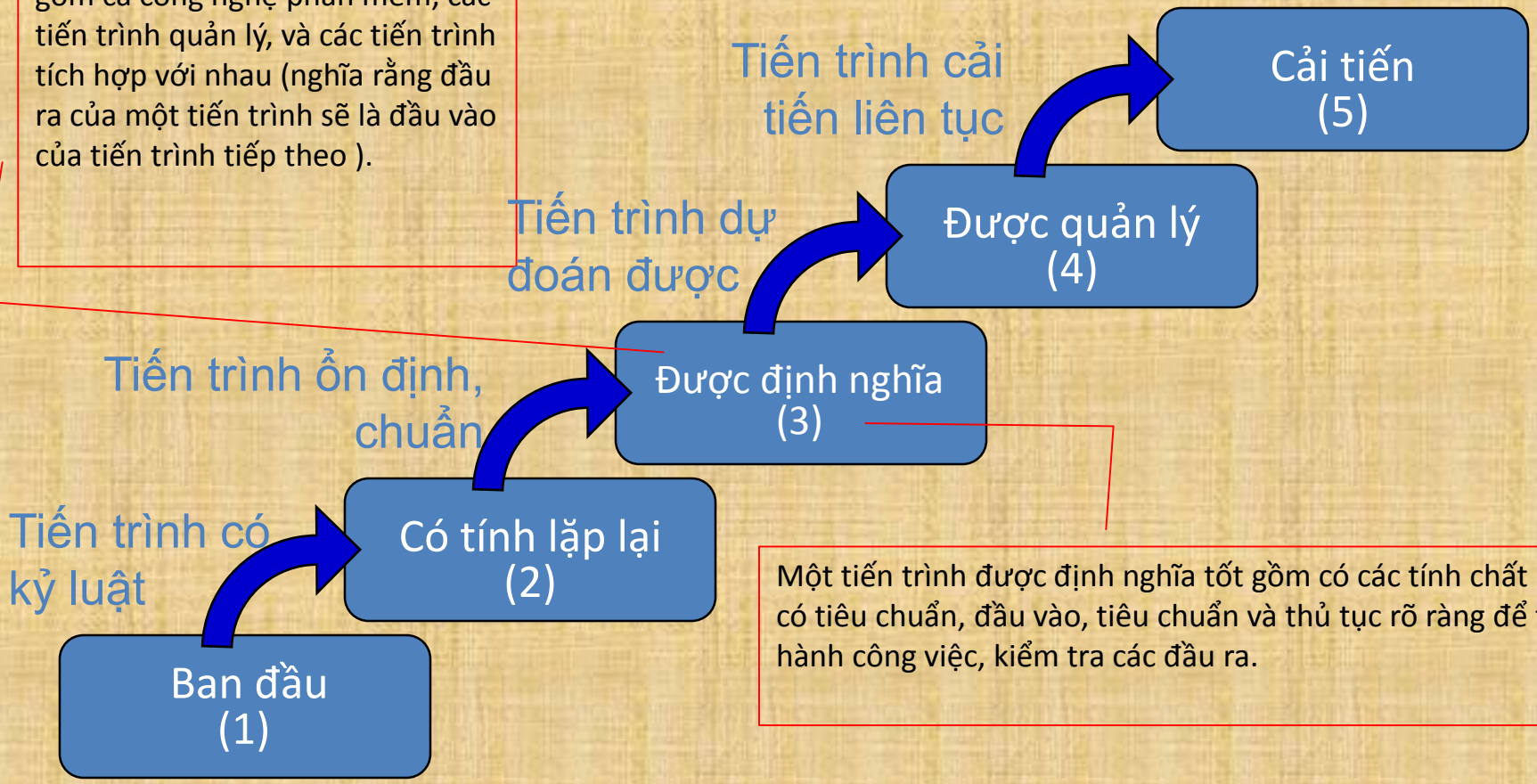
Mô hình 5 mức của CMM

Có sự cải tiến hơn, chiến lược quản lý dự án vụ thủ tục để thực thi chiến lược ấy được thiết lập. Các kế hoạch và quản lý dự án mới được dựa trên những kinh nghiệm của dự án cũ.

Kết quả là đưa được những hiệu quả quản lý tiến trình của một dự án nụy vào một dự án khác. Điều này cho phép lặp lại (repeatable) những thành công đối với một dự án tương tự mặc dù có thể các dự án này cũng có những điểm khác biệt.

Mô hình 5 mức của CMM

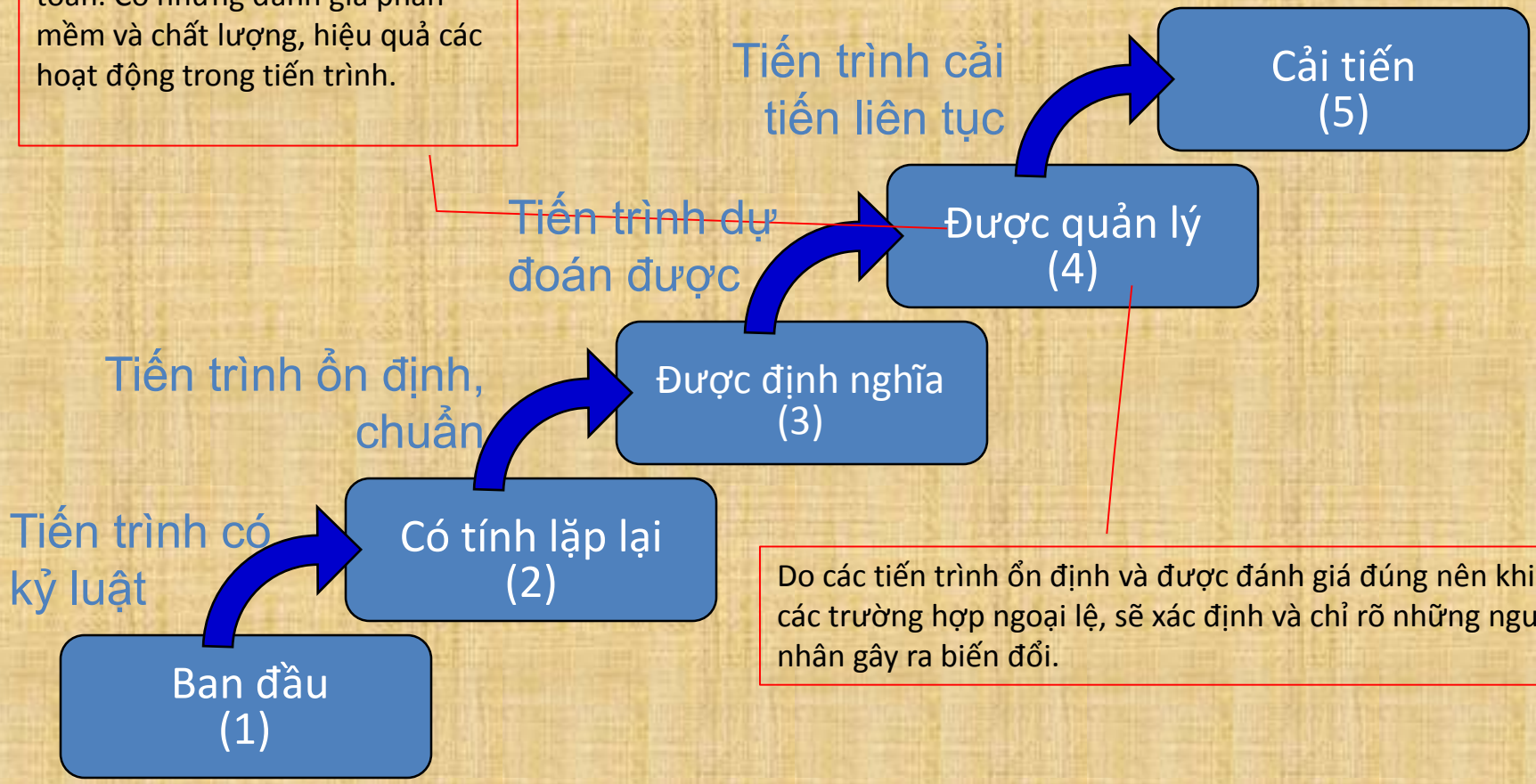
Lập được tài liệu tiến trình tiêu chuẩn đối với việc phát triển và bảo trì phần mềm có tổ chức, bao gồm cả công nghệ phần mềm, các tiến trình quản lý, và các tiến trình tích hợp với nhau (nghĩa rằng đầu ra của một tiến trình sẽ là đầu vào của tiến trình tiếp theo).



Một tiến trình được định nghĩa tốt gồm có các tính chất như có tiêu chuẩn, đầu vào, tiêu chuẩn và thủ tục rõ ràng để tiến hành công việc, kiểm tra các đầu ra.

Mô hình 5 mức của CMM

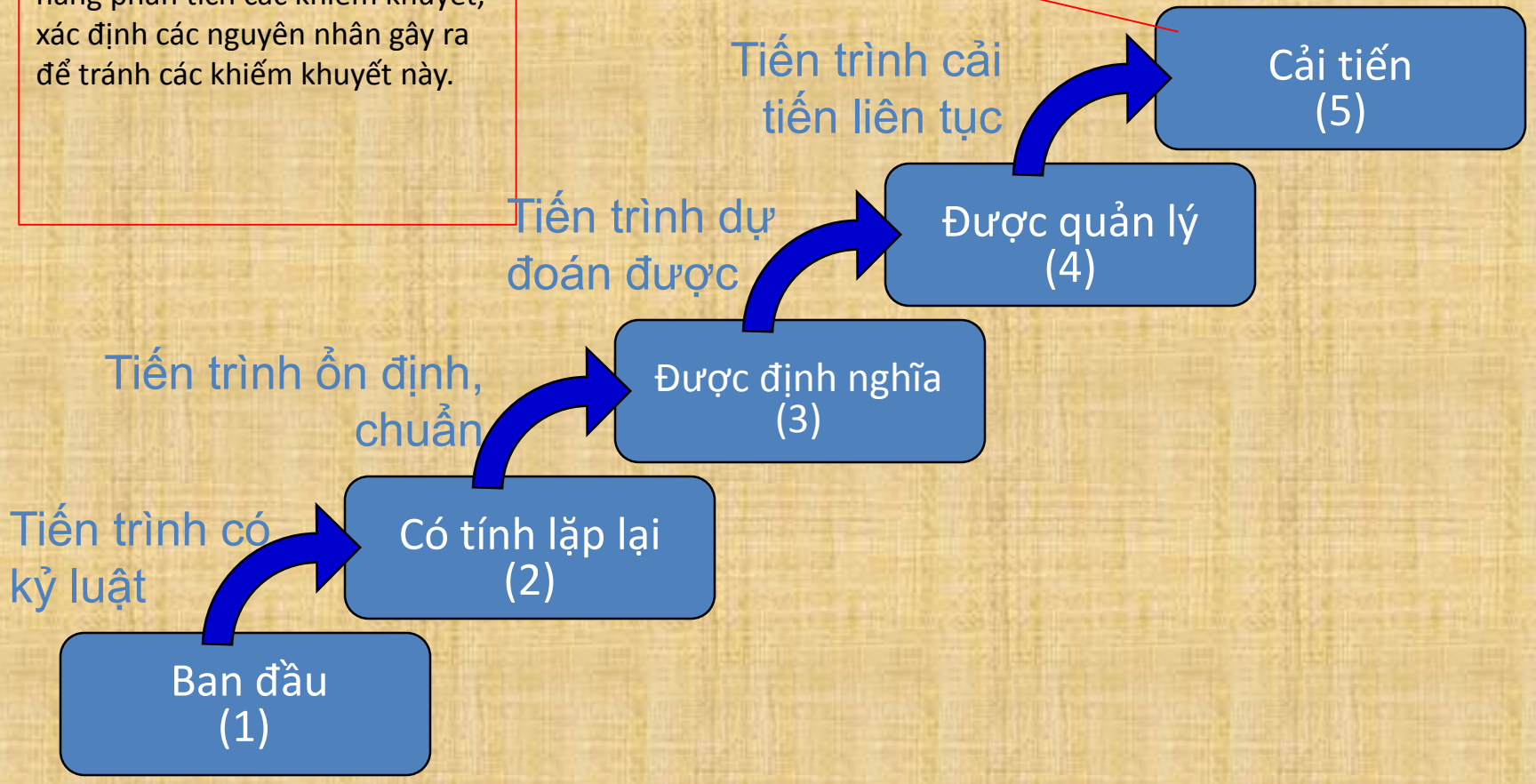
Mục tiêu là điều khiển tiến trình. Các tiến trình phần mềm được quản lý để vận hành ổn định, an toàn. Có những đánh giá phần mềm và chất lượng, hiệu quả các hoạt động trong tiến trình.



Do các tiến trình ổn định và được đánh giá đúng nên khi có các trường hợp ngoại lệ, sẽ xác định và chỉ rõ những nguyên nhân gây ra biến đổi.

Mô hình 5 mức của CMM

Tiếp tục cải tiến tiến trình, có thể xác định được những điểm mạnh và điểm yếu của tiến trình, có khả năng phân tích các khiếm khuyết, xác định các nguyên nhân gây ra để tránh các khiếm khuyết này.



18 Vùng tiến trình chính KPA (Key Process Area)

LEVEL 2: Có thể lặp
1. Quản lý cấu hình phần mềm
2. Đảm bảo chất lượng phần mềm
3. Quản lý hợp đồng con phần mềm
4. Theo dõi và giám sát dự án phần mềm
5. Lập kế hoạch dự án phần mềm
6. Quản lý yêu cầu

7. Xem xét ngang nhau
8. Hợp tác giữa các nhóm
9. Kỹ thuật sản phẩm phần mềm
10. Quản lý phần mềm tích hợp
11. Chương trình huấn luyện
12. Định nghĩa tiến trình tổ chức
13. Trọng tâm tiến trình tổ chức

MỨC 3: Được định nghĩa

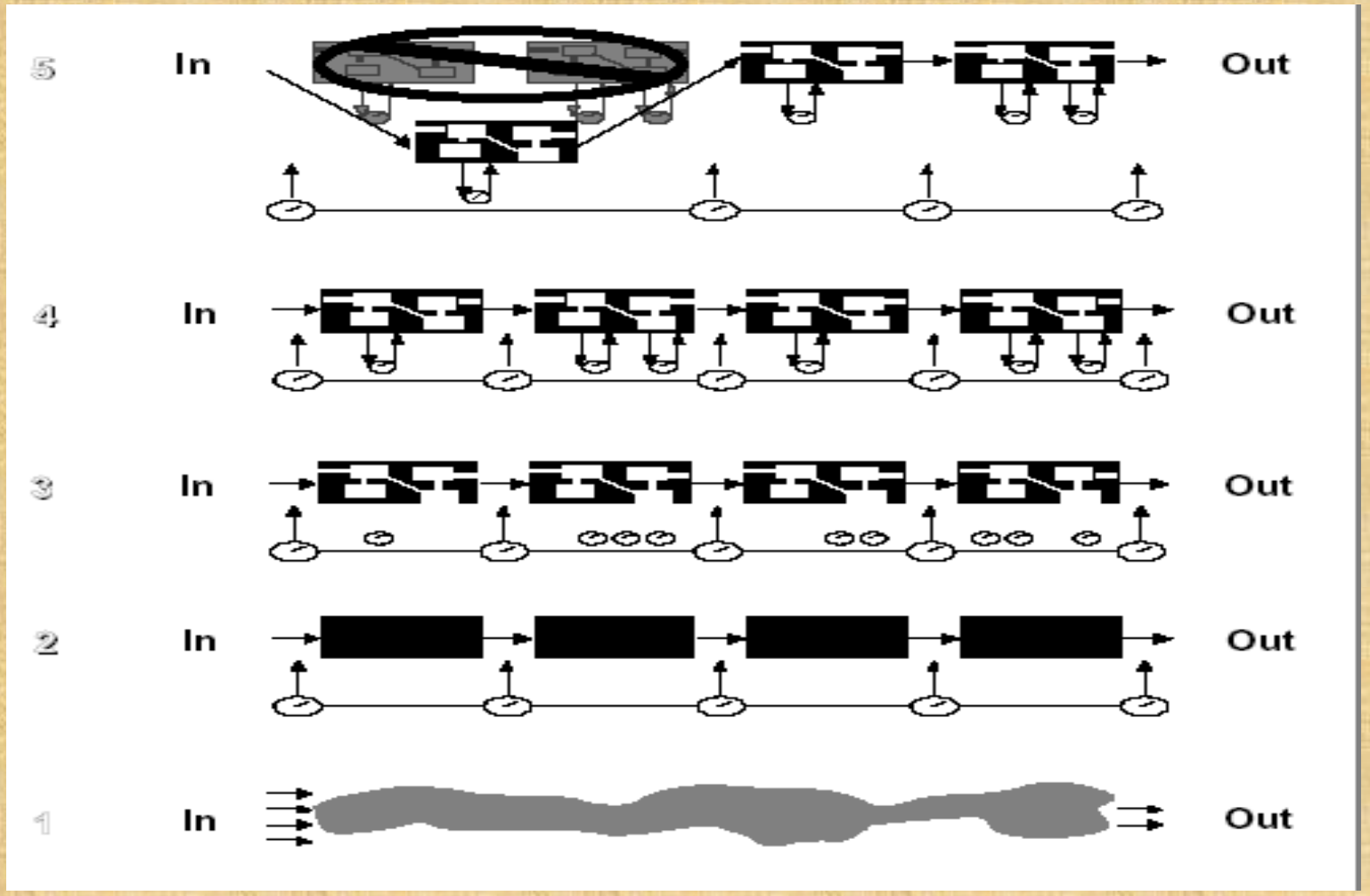
14. Quản lý chất lượng phần mềm
15. Quản lý quá trình định lượng

MỨC 4: Được quản lý

16. Quản lý thay đổi tiến trình
17. Quản lý thay đổi công nghệ
18. Phòng ngừa khiêm khuyết

MỨC 5: Cải tiến

Khả năng nhìn nhận tại mỗi mức thuần thục



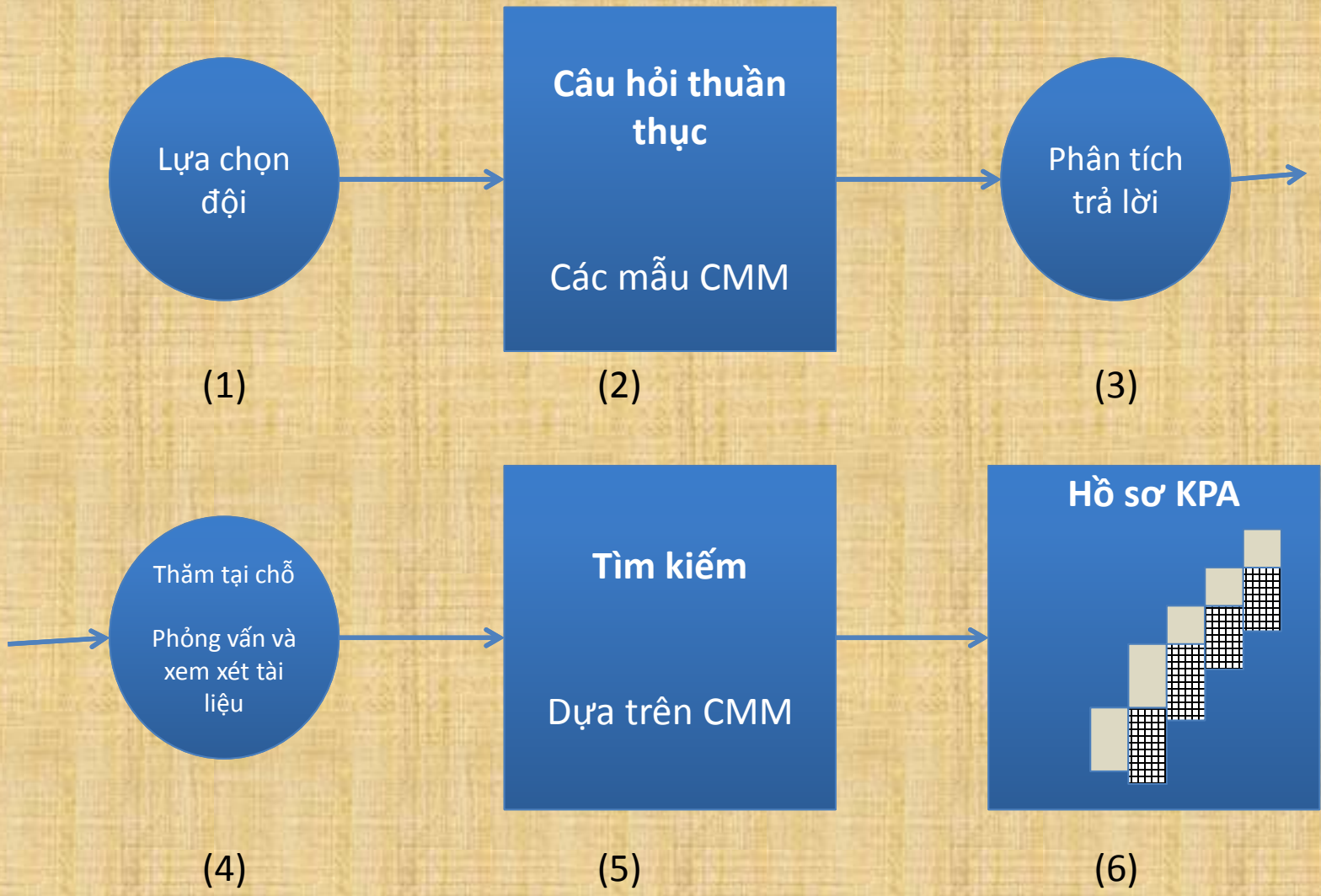
Khả năng tiến trình và dự đoán theo các mức của CMM

- Khi mức độ thuần thục tăng, sự sai khác giữa kết quả đạt được và kết quả dự tính giảm xuống.
- Khi mức độ thuần thục tăng, độ biến động của kết quả thực tế so với kết quả đề ra giảm xuống.
- Khi mức độ thuần thục tăng thì các kết quả sẽ được cải thiện. Đó là, chi phí giảm, thời gian phát triển ngắn hơn, chất lượng và năng suất tăng.

Cách thức sử dụng mô hình CMM

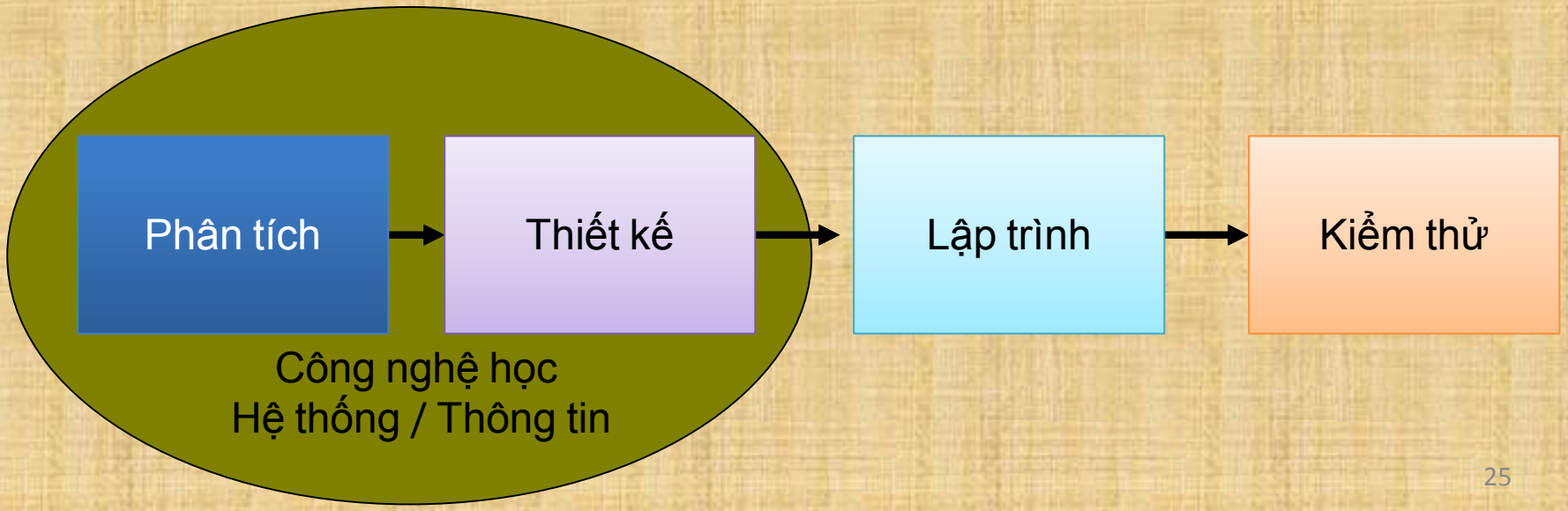
- Định giá tiến trình phần mềm (Software process assessments) xác định trạng thái của tiến trình phần mềm hiện tại của tổ chức, xác định mức độ ưu tiên đối với các vấn đề có liên quan tới tiến trình phần mềm khi xử lý chúng và xây dựng hệ thống hỗ trợ phát triển tiến trình phần mềm.
- Đánh giá khả năng phần mềm (Software capability evaluations) xác định các nhà thầu có đủ tư cách triển khai một dự án phần mềm hoặc quản lý hiện trạng của một hệ thống phần mềm đã có sẵn.

Những điểm chung của 2 phương thức sử dụng CMM



3.2. Mô hình tuyến tính

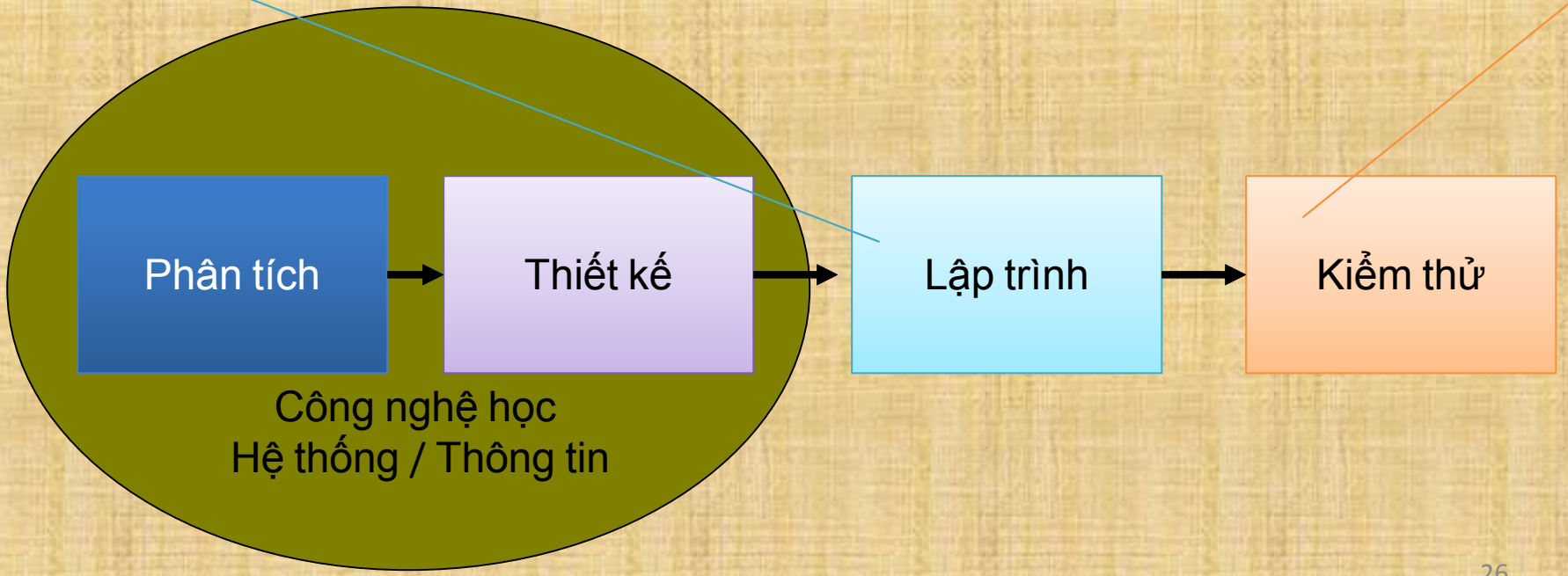
- Công nghệ học Hệ thống / Thông tin và mô hình hóa (System / Information engineering and modeling): thiết lập các yêu cầu, ánh xạ một số tập con các yêu cầu sang phần mềm trong quá trình tương tác giữa phần cứng, người và CSDL



Mô hình tuyến tính

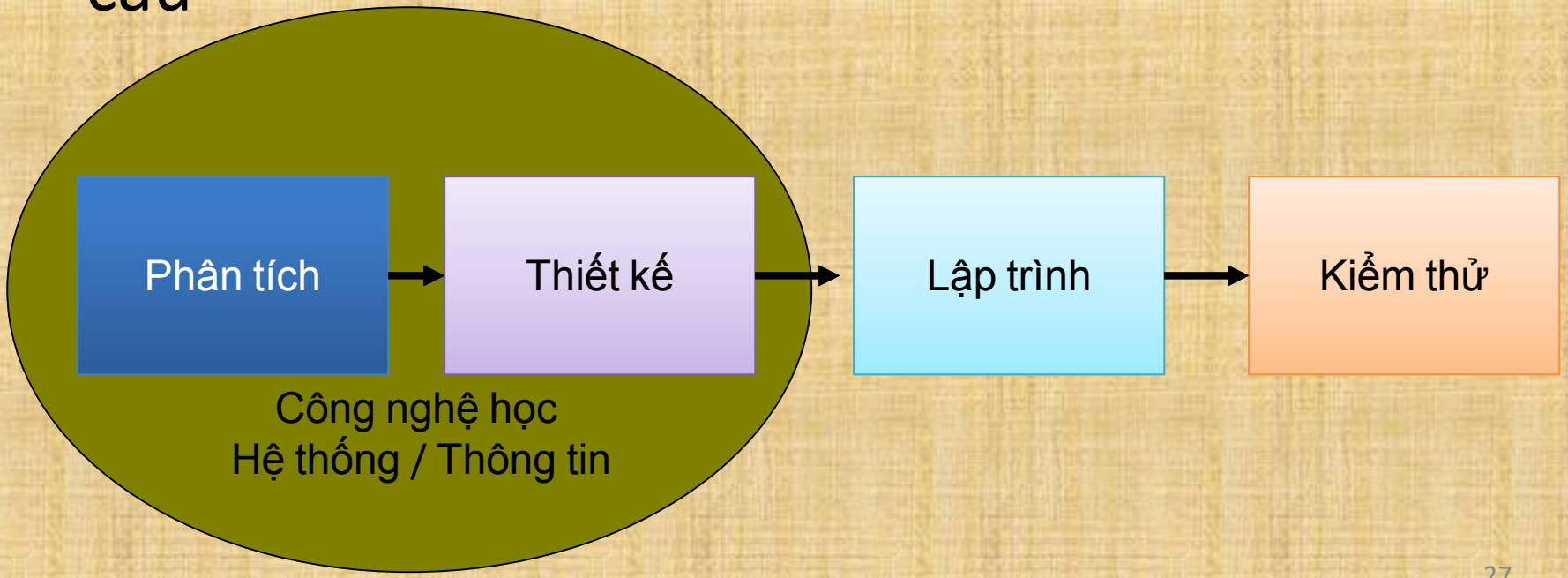
Tạo mã / lập trình (Code generation / programming): Chuyển thiết kế thành chương trình máy tính bởi ngôn ngữ nào đó. Nếu thiết kế đã được chi tiết hóa thì lập trình có thể chỉ thuần túy cơ học

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



Mô hình tuyến tính

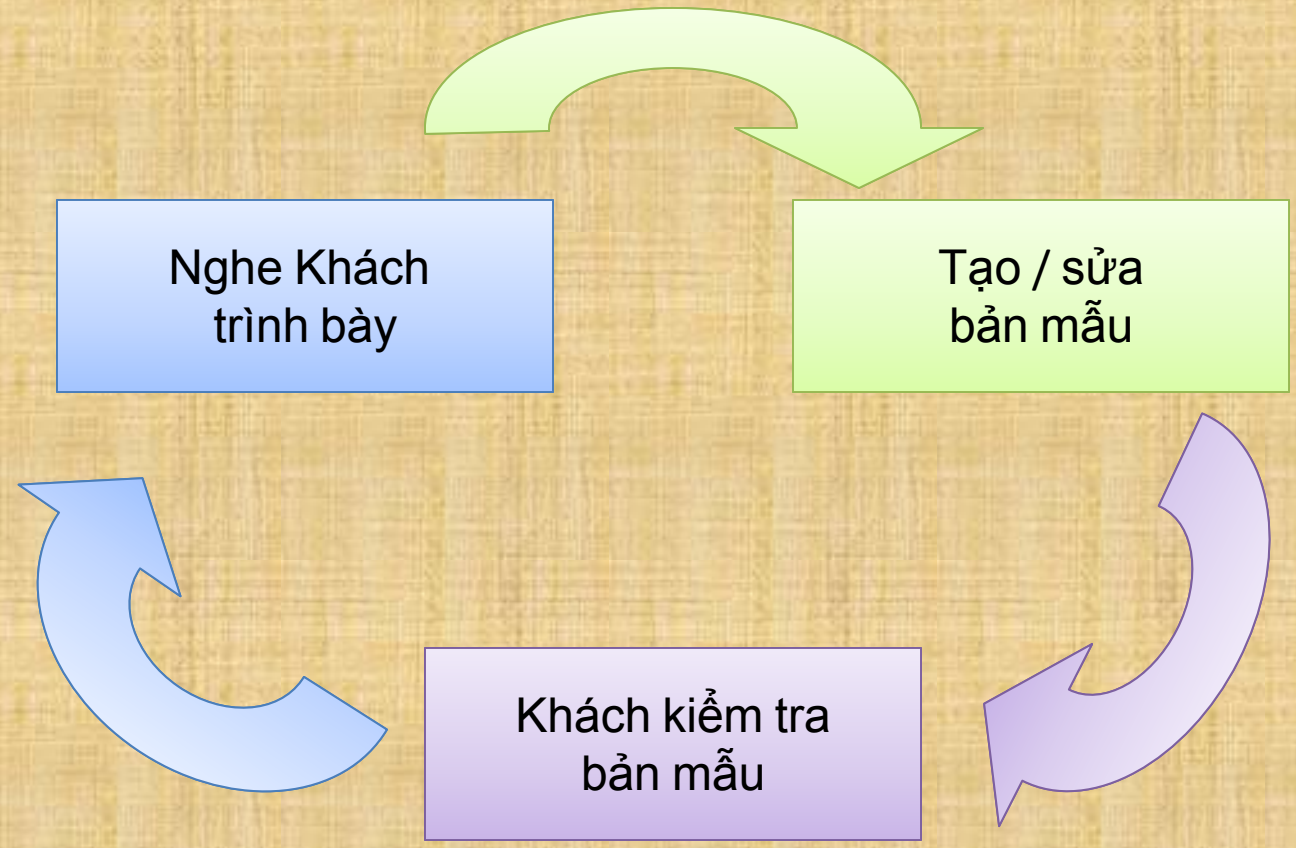
- Hỗ trợ / Bảo trì (Support / Maintenance): Đáp ứng những thay đổi, nâng cấp phần mềm đã phát triển do sự thay đổi của môi trường, nhu cầu



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!

3.3. 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

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

- Là quy trình phát triển phần mềm gia tăng, tăng dần từng bước (Incremental 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: Mô hình dữ liệu và tiến trình

- Mô hình dữ liệu (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
- Mô hình tiến trình (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: Tạo ứng dụng và kiểm thử

- Tạo ứng dụng (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
- Kiểm thử (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

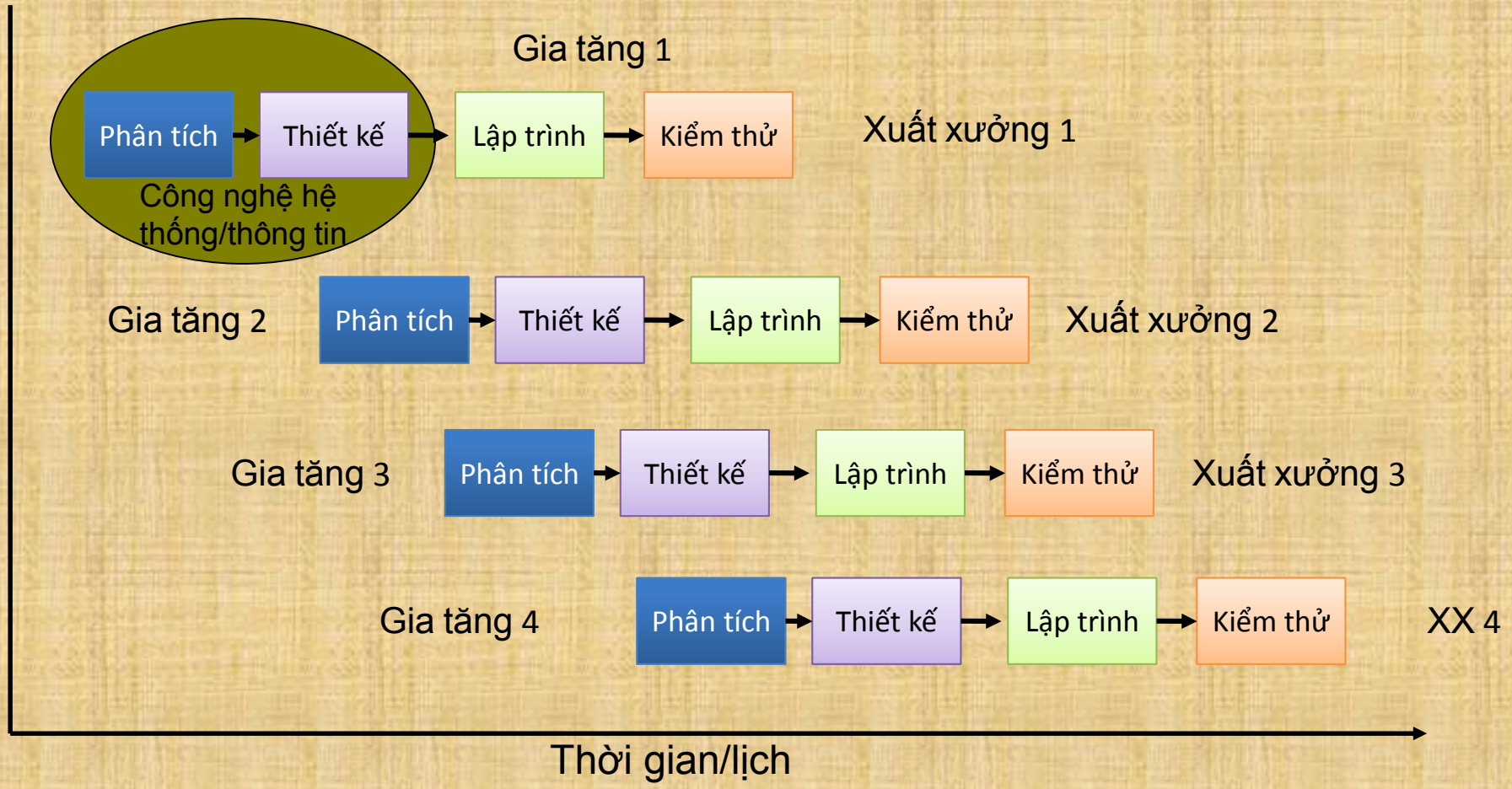
3.5. Các mô hình tiến hóa

- 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 tiêu biểu:
  - Gia tăng (Incremental)
  - Xoắn ốc (Spiral)
  - Xoắn ốc WINWIN (WINWIN spiral)
  - Phát triển đồng thời (Concurrent development)

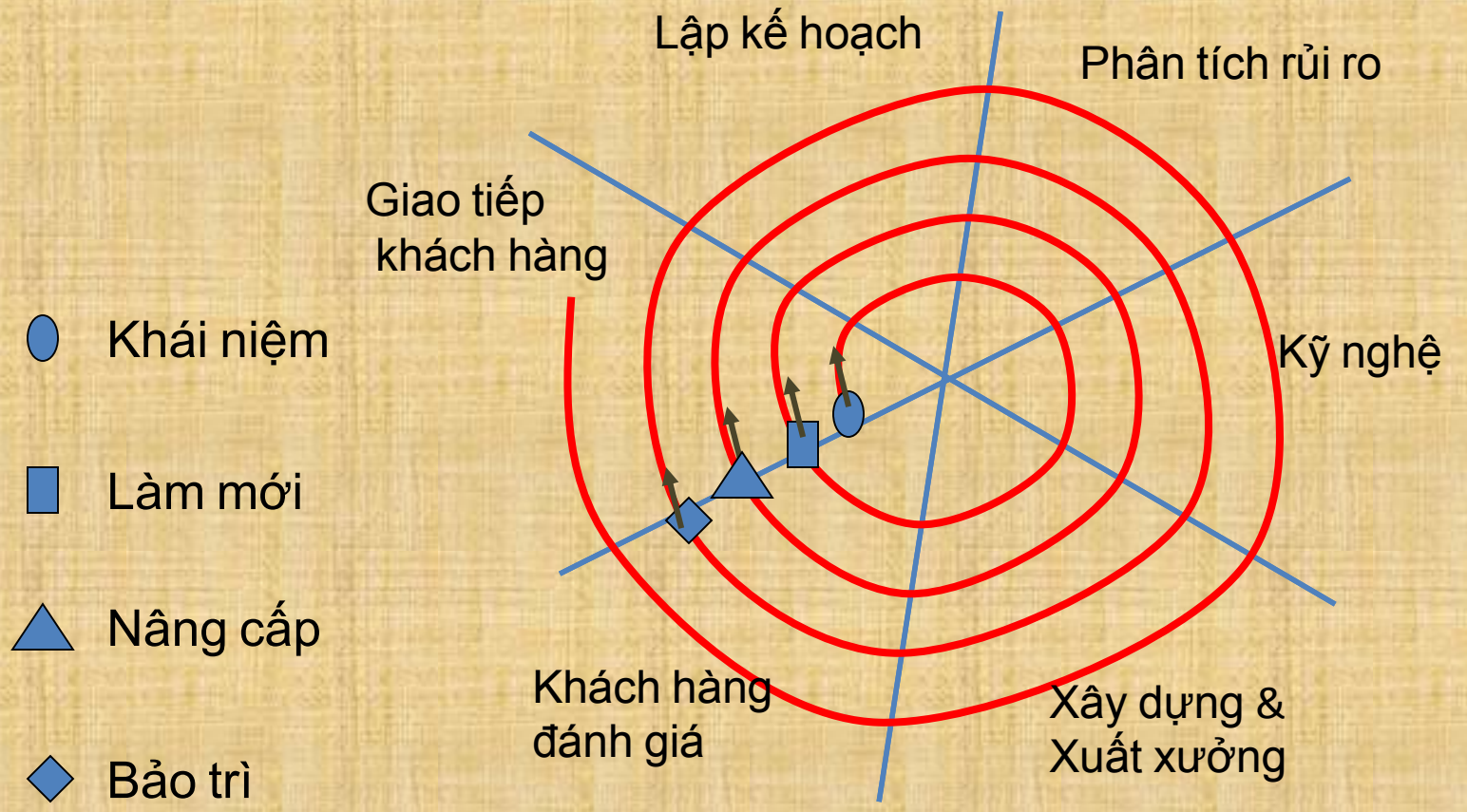
Mô hình gia tăng (The incremental model)

- 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

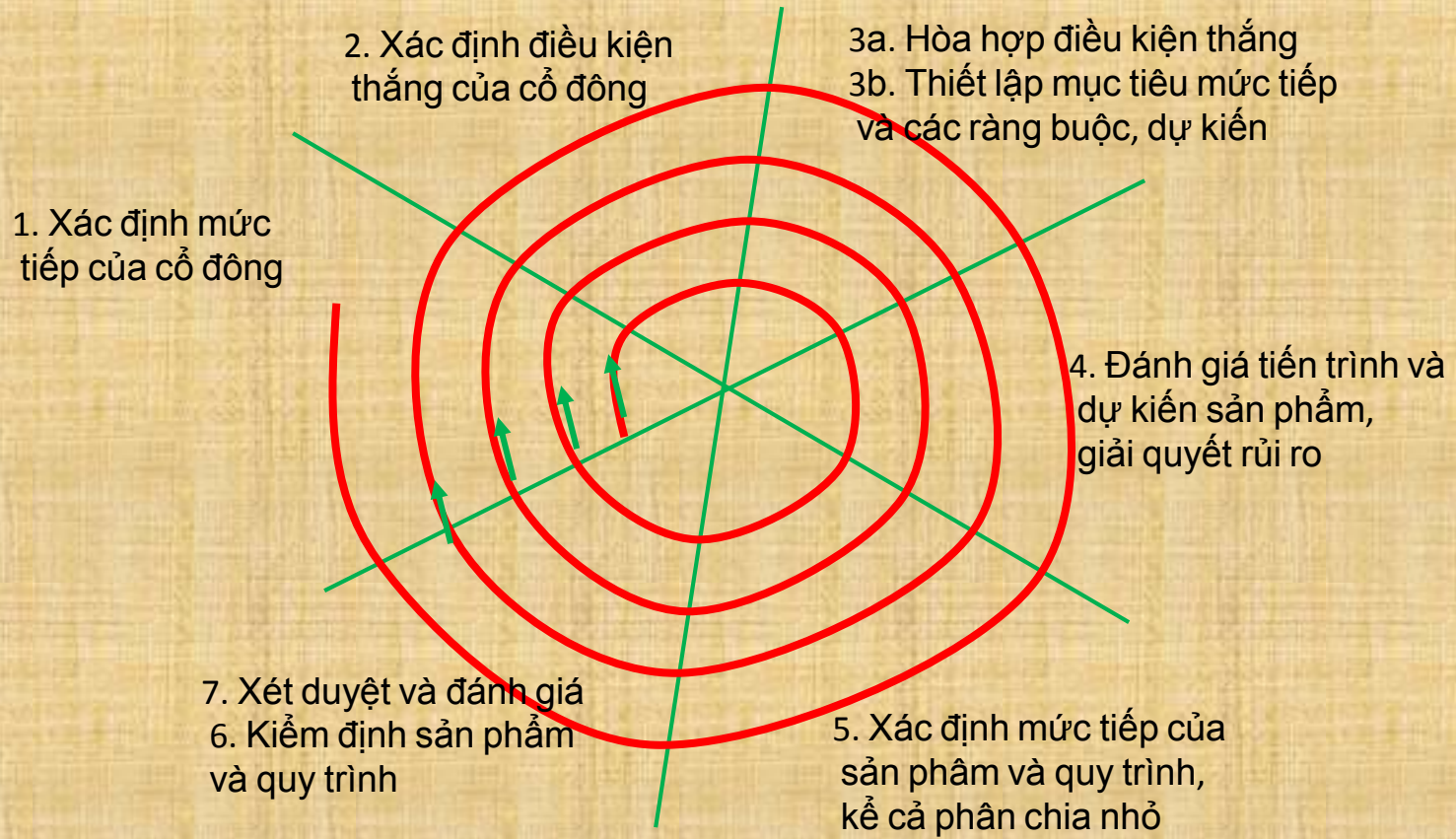
Mô hình xoắn ốc: Mạnh và yếu?

- 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 xoắn ốc WINWIN

- Nhằm thỏa hiệp giữa người phát triển và khách hàng, cả hai cùng "Thắng" (win-win)
  - Khách thì có phần mềm thỏa mãn yêu cầu chính
  - Người phát triển thì có kinh phí thỏa đáng và thời gian hợp lý
- Các hoạt động chính trong xác định hệ thống:
  - Xác định cổ đông (stakeholders)
  - Xác định điều kiện thắng của cổ đông
  - Thỏa hiệp điều kiện thắng của các bên liên quan

Mô hình xoắn ốc WINWIN



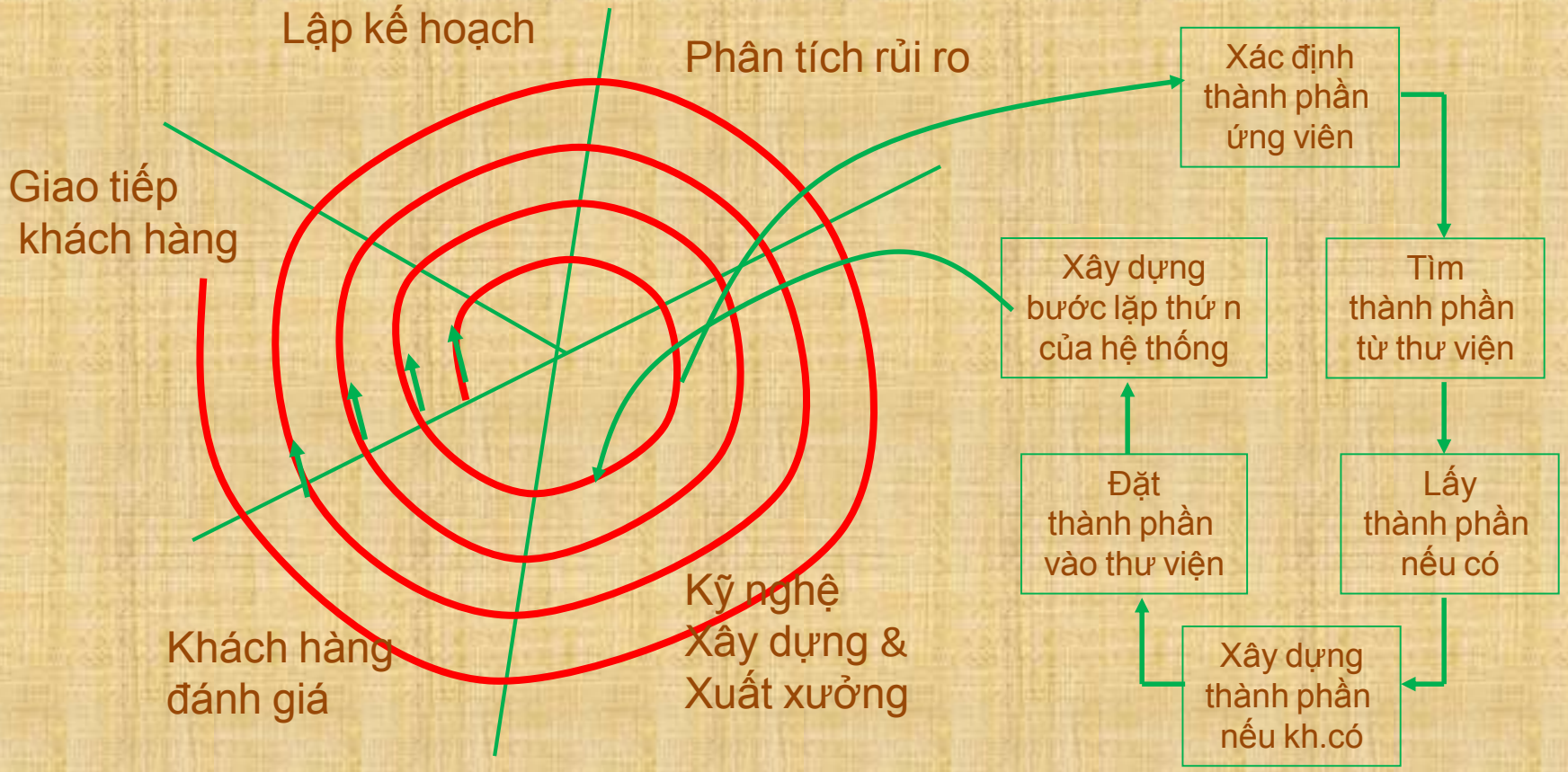
Mô hình phát triển đồng thời (concurrent development)

- 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): hệ thống và các thành phần cấu thành hệ thống được phát triển đồng thời

3.6. 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 dựa thành phần



2.3.7. Mô hình RUP (Rational Unified Process)

- SV tự nghiên cứu

Tổng kết các mô hình

- Thác nước: mô hình tuyến tính
- Chế thử: mô hình lặp đi lặp lại
- Gia tăng: kết hợp giữa mô hình tuyến tính và lặp đi lặp lại
- Xoăn ốc: kết hợp giữa mô hình tuyến tính và lặp đi lặp lại
- Phát triển nhanh: mô hình lặp đi lặp lại

3.8. Các kỹ thuật thế hệ thứ 4 (Fourth generation techniques)

- Tập hợp các công cụ cho phép xác định đặc tính phần mềm ở mức cao, sau đó sinh tự động mã nguồn dựa theo đặc tả đó
- Các công cụ 4GT điển hình: ngôn ngữ phi thủ tục cho truy vấn CSDL; tạo báo cáo; xử lý dữ liệu; tương tác màn hình; tạo mã nguồn; khả năng đồ họa bậc cao; khả năng bảng tính; khả năng giao diện Web; vv

4GT: Tại sao ?

- Từ thu thập yêu cầu cho đến sản phẩm: đối thoại giữa khách và người phát triển là quan trọng
- Không nên bỏ qua khâu thiết kế. 4GT chỉ áp dụng để triển khai thiết kế qua 4GL
- Mạnh: giảm thời gian phát triển và tăng năng suất
- Yếu: 4GT khó dùng hơn ngôn ngữ lập trình, mã khó tối ưu và khó bảo trì cho hệ thống lớn $\Rightarrow$ cần kỹ năng của kỹ sư phần mềm
- Tương lai: 4GT với mô hình theo thành phần

4. Đánh giá: Sản phẩm và quy trình (Product and process)

- Quy trình yếu thì sản phẩm khó mà tốt, song không nên coi trọng quá mức vào quy trình hoặc quá mức vào sản phẩm
- Sản phẩm và quy trình cần được coi trọng như nhau