Nhập môn Công nghệ học Phần mềm
(Introduction to Software Engineering)

Department of Software Engineering
Faculty of Information Technology
Hanoi University of Technology
TEL: 04-8682595 FAX: 04-8692906
Email: cnpm@it-hut.edu.vn

Cấu trúc môn học

- 45 tiết + 1 Đồ án môn học
- Cần những kiến thức căn bản về CNTT
- Cung cấp những nguyên lý chung về Công nghệ học Phần mềm (CNHPM)
- Cung cấp kiến thức để học các môn chuyên ngành hẹp như Phân tích và thiết kế phần mềm, Xây dựng và đánh giá phần mềm, Quản trị dự án phần mềm,...

Cấu trúc môn học (tiếp)

- **Nội dung: gồm 6 phần với 11 chương**
  - **Giới thiệu chung về CNHPM (3 buổi)**
  - **Quản lý dự án PM (2b)**
  - **Yêu cầu người dùng (1b)**
  - **Thiết kế và lập trình (2b)**
  - **Kiểm thử và bảo trì (2b)**
  - **Chủ đề nâng cao và tổng kết (1b+1b)**
- **Đánh giá: Thi hết môn + Đồ án môn học**

Tài liệu tham khảo

- R. Pressman, *Software Engineering: A Practioner's Approach*. 5th Ed., McGraw-Hill, 2001
- R. Pressman, Kỹ nghệ phần mềm. Tập 1, 2, 3. NXB Giáo dục, Hà Nội, 1997 (Người dịch: Ngô Trung Việt)
- I. Sommerville, *Software Engineering.* 5th Ed., Addison-Wesley, 1995
- K. Kawamura, *Nhập môn Công nghệ học Phần mềm*. NXB Kinki-Kagaku, Tokyo, 2001 (Tiếng Nhật)

Phần I
Giới thiệu chung về CNHPM

Chương 1: Bản chất phần mềm

1.1 Định nghĩa chung về phần mềm
1.2 Kiến trúc phần mềm
1.3 Các khái niệm
1.4 Đặc tính chung của phần mềm
1.5 Thế nào là phần mềm tốt ?
1.6 Các ứng dụng phần mềm

1.1. Định nghĩa chung về phần mềm

- **Phần mềm (Software - SW) như một khái niệm đối nghĩa với phần cứng (Hardware - HW), tuy nhiên, đây là 2 khái niệm tương đối**
- **Từ xưa, SW như thứ được cho không hoặc bán kèm theo máy (HW)**
- **Dần dần, giá thành SW ngày càng cao và nay cao hơn HW**

Các đặc tính của SW và HW

HW	SW
• Vật "cứng"	• Vật "mềm"
• Kim loại	• Kỹ thuật sử dụng
• Vật chất	• Trừu tượng
• Hữu hình	• Vô hình
• Sản xuất công nghiệp bởi máy móc là chính	• Sản xuất bởi con người là chính
• Định lượng là chính	• Định tính là chính
• Hỏng hóc, hao mòn	• Không hao mòn

Định nghĩa 1: Phần mềm là

- Các lệnh (chương trình máy tính) khi được thực hiện thì cung cấp những chức năng và kết quả mong muốn
- Các cấu trúc dữ liệu làm cho chương trình thao tác thông tin thích hợp
- Các tư liệu mô tả thao tác và cách sử dụng chương trình

SW đối nghĩa với HW

- Vai trò SW ngày càng thể hiện trội
- Máy tính là . . . chiếc hộp không có SW
- Ngày nay, SW quyết định chất lượng một hệ thống máy tính (HTMT), là chủ đề cốt lõi, trung tâm của HTMT
- Hệ thống máy tính gồm HW và SW

Định nghĩa 2

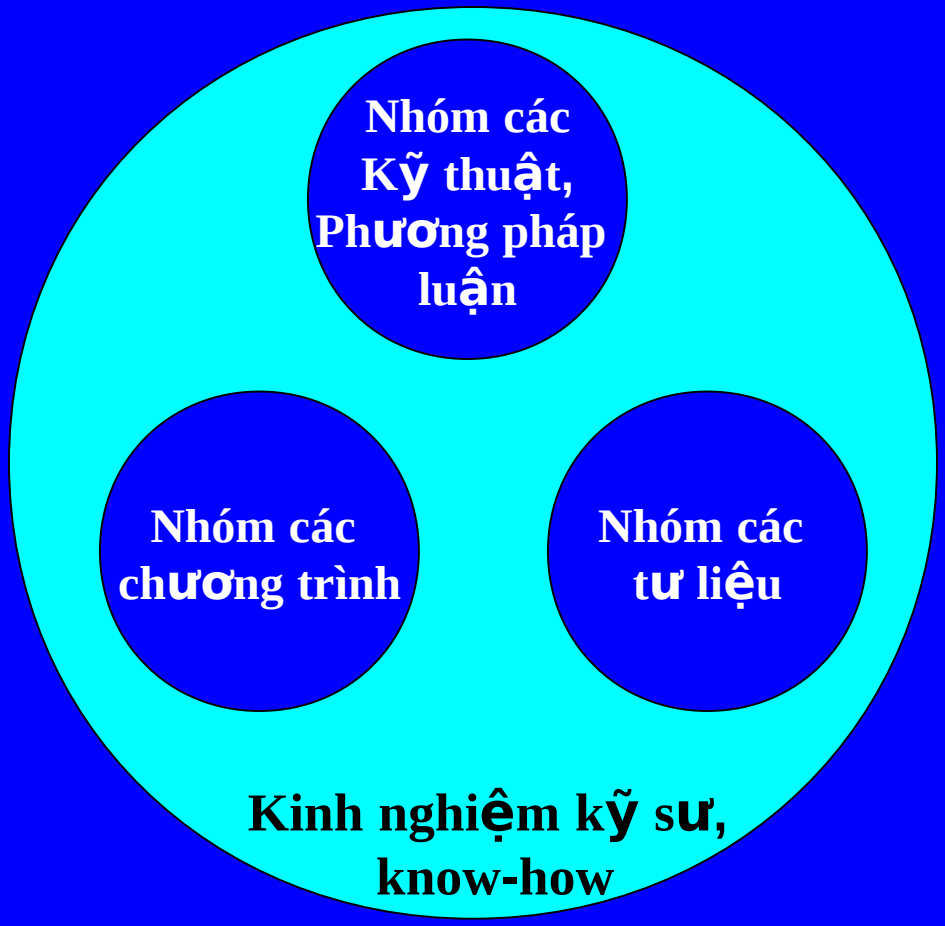
Trong một hệ thống máy tính, nếu trừ bỏ đi các thiết bị và các loại phụ kiện thì phần còn lại chính là phần mềm (SW)

- Nghĩa hẹp: SW là dịch vụ chương trình để tăng khả năng xử lý của phần cứng của máy tính (như hệ điều hành - OS)
- Nghĩa rộng: SW là tất cả các kỹ thuật ứng dụng để thực hiện những dịch vụ chức năng cho mục đích nào đó bằng phần cứng

SW theo nghĩa rộng

- Không chỉ SW cơ bản và SW ứng dụng
- Phải gồm cả khả năng, kinh nghiệm thực tiễn và kỹ năng của kỹ sư (người chế ra phần mềm): Know-how of Software Engineer
- Là tất cả các kỹ thuật làm cho sử dụng phần cứng máy tính đạt hiệu quả cao

Phần mềm là gì ?



Nhóm các kỹ thuật, phương pháp luận

- Các khái niệm và trình tự cụ thể hóa một hệ thống
- Các phương pháp tiếp cận giải quyết vấn đề
- Các trình tự thiết kế và phát triển được chuẩn hóa
- Các phương pháp đặc tả yêu cầu, thiết kế hệ thống, thiết kế chương trình, kiểm thử, toàn bộ quy trình quản lý phát triển phần mềm

Nhóm các chương trình

- Là phần giao diện với phần cứng, tạo thành từ các nhóm lệnh chỉ thị cho máy tính biết trình tự thao tác xử lý dữ liệu
- Phần mềm cơ bản: với chức năng cung cấp môi trường thao tác dễ dàng cho người sử dụng nhằm tăng hiệu năng xử lý của phần cứng (ví dụ như OS là chương trình hệ thống)
- Phần mềm ứng dụng: dùng để xử lý nghiệp vụ thích hợp nào đó (quản lý, kế toán, . . .), phần mềm đóng gói, phần mềm của người dùng, . . .

Nhóm các tư liệu

- Những tư liệu hữu ích, có giá trị cao và rất cần thiết để phát triển, vận hành và bảo trì phần mềm
- Để chế ra phần mềm với độ tin cậy cao cần tạo ra các tư liệu chất lượng cao: đặc tả yêu cầu, mô tả thiết kế từng loại, điều kiện kiểm thử, thủ tục vận hành, hướng dẫn thao tác

Những yếu tố khác

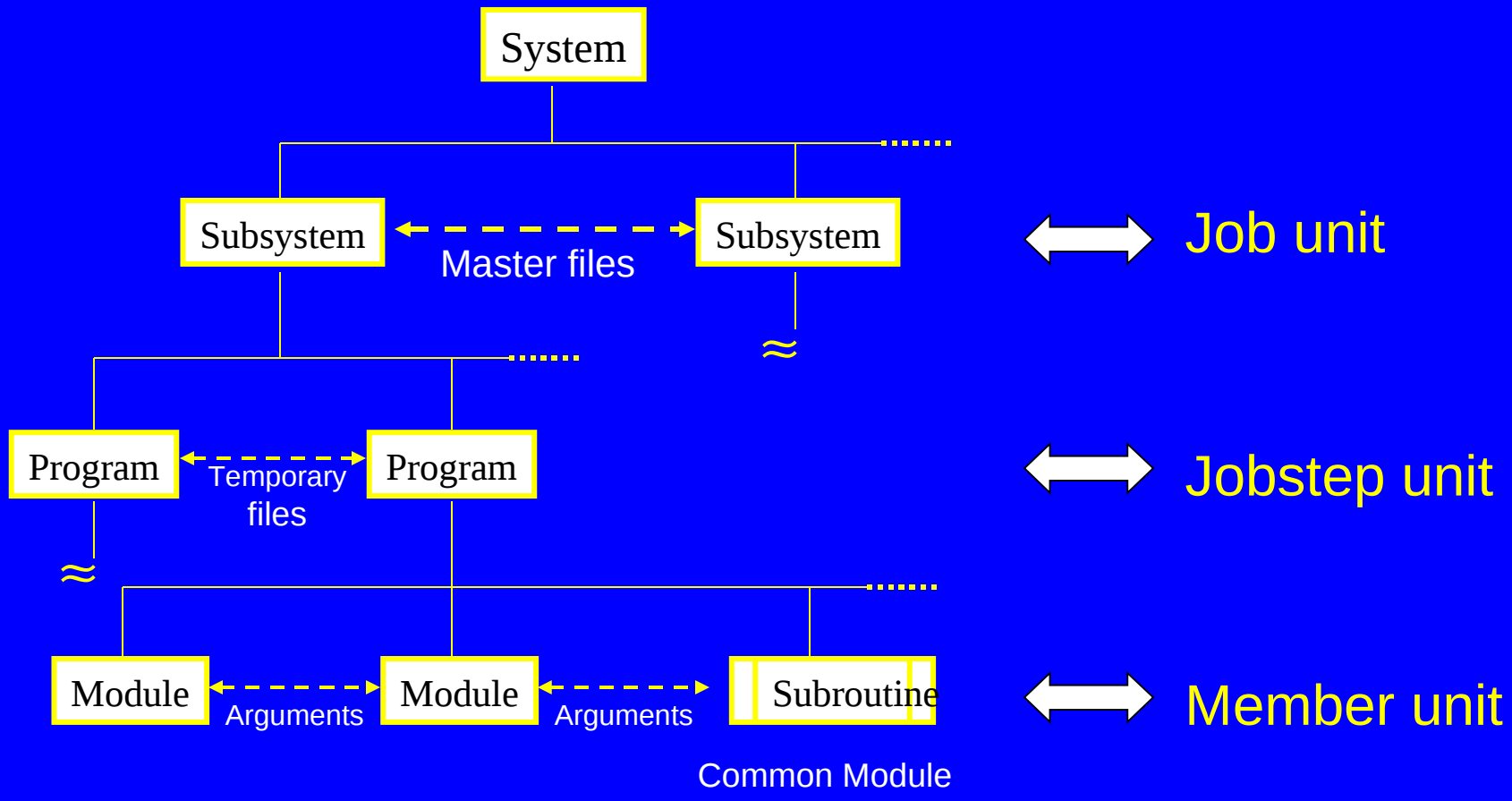
- Sản xuất phần mềm phụ thuộc rất nhiều vào con người (kỹ sư phần mềm). Khả năng hệ thống hóa trừu tượng, khả năng lập trình, kỹ năng công nghệ, kinh nghiệm làm việc, tầm bao quát, . . .: khác nhau ở từng người
- Phần mềm phụ thuộc nhiều vào ý tưởng (idea) và kỹ năng (know-how) của người/nhóm tác giả

1.2 Kiến trúc phần mềm

1.2.1 Phần mềm nhìn từ cấu trúc phân cấp

- Cấu trúc phần mềm là cấu trúc phân cấp (hierarchical structure): mức trên là hệ thống (system), dưới là các hệ thống con (subsystems)
- Dưới hệ thống con là các chương trình
- Dưới chương trình là các Modules hoặc Subroutines với các đối số (arguments)

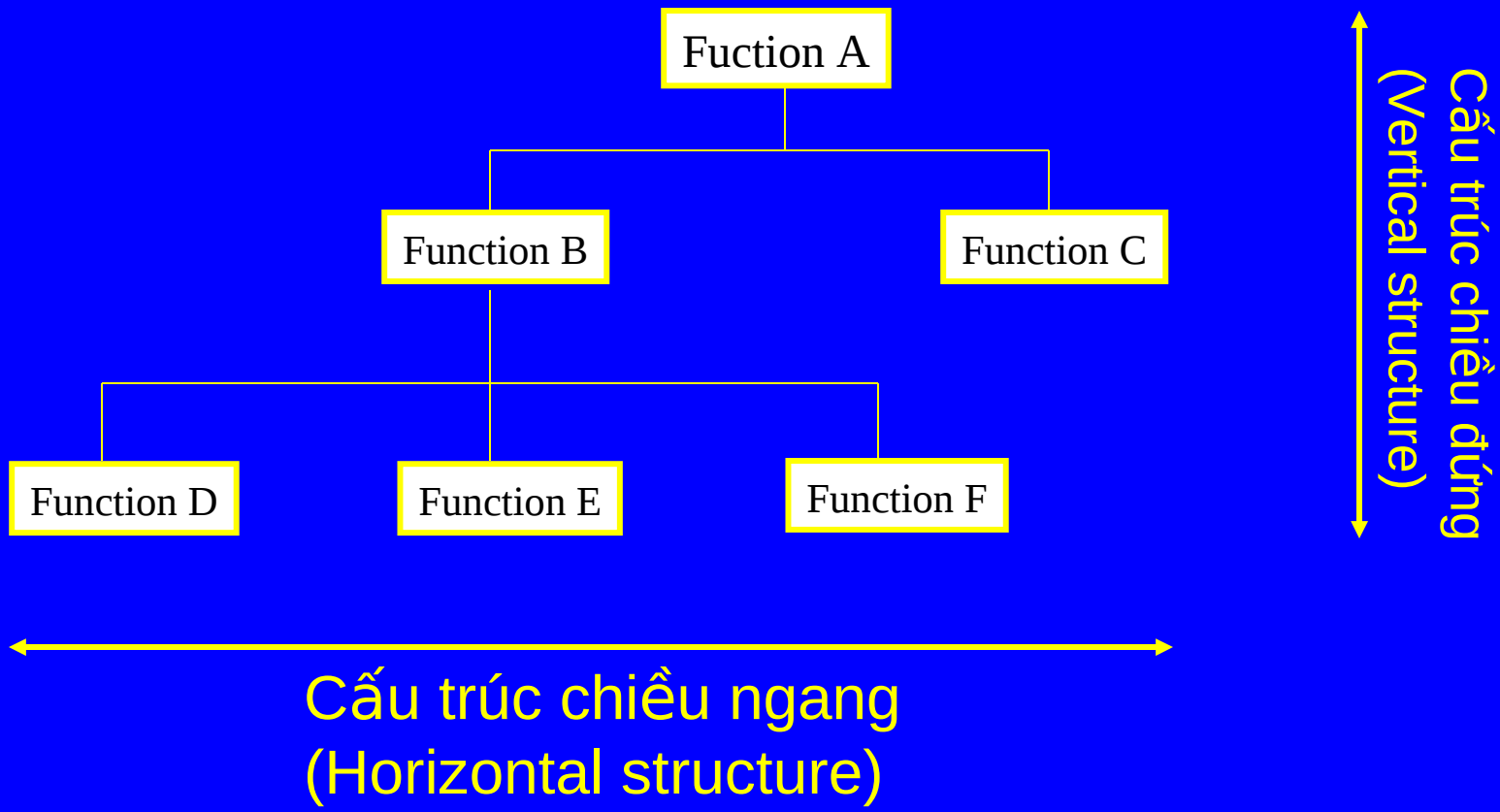
Kiến trúc phần mềm



1.2.2 Phần mềm nhìn từ cấu trúc và thủ tục

- Hai yếu tố cấu thành của phần mềm
  - Phương diện cấu trúc
  - Phương diện thủ tục
- Cấu trúc phần mềm: biểu thị kiến trúc các chức năng mà phần mềm đó có và điều kiện phân cấp các chức năng (thiết kế cấu trúc)
- Thiết kế chức năng: theo chiều đứng (càng sâu càng phức tạp) và chiều ngang (càng rộng càng nhiều chức năng, qui mô càng lớn)

Cấu trúc phần mềm



Thủ tục (procedure) phần mềm

- Là những quan hệ giữa các trình tự mà phần mềm đó có
- Thuật toán với những phép lặp, rẽ nhánh, điều khiển luồng xử lý (quay lui hay bỏ qua)
- Là cấu trúc lôgic biểu thị từng chức năng có trong phần mềm và trình tự thực hiện chúng
- Thiết kế cấu trúc trước rồi sang chức năng

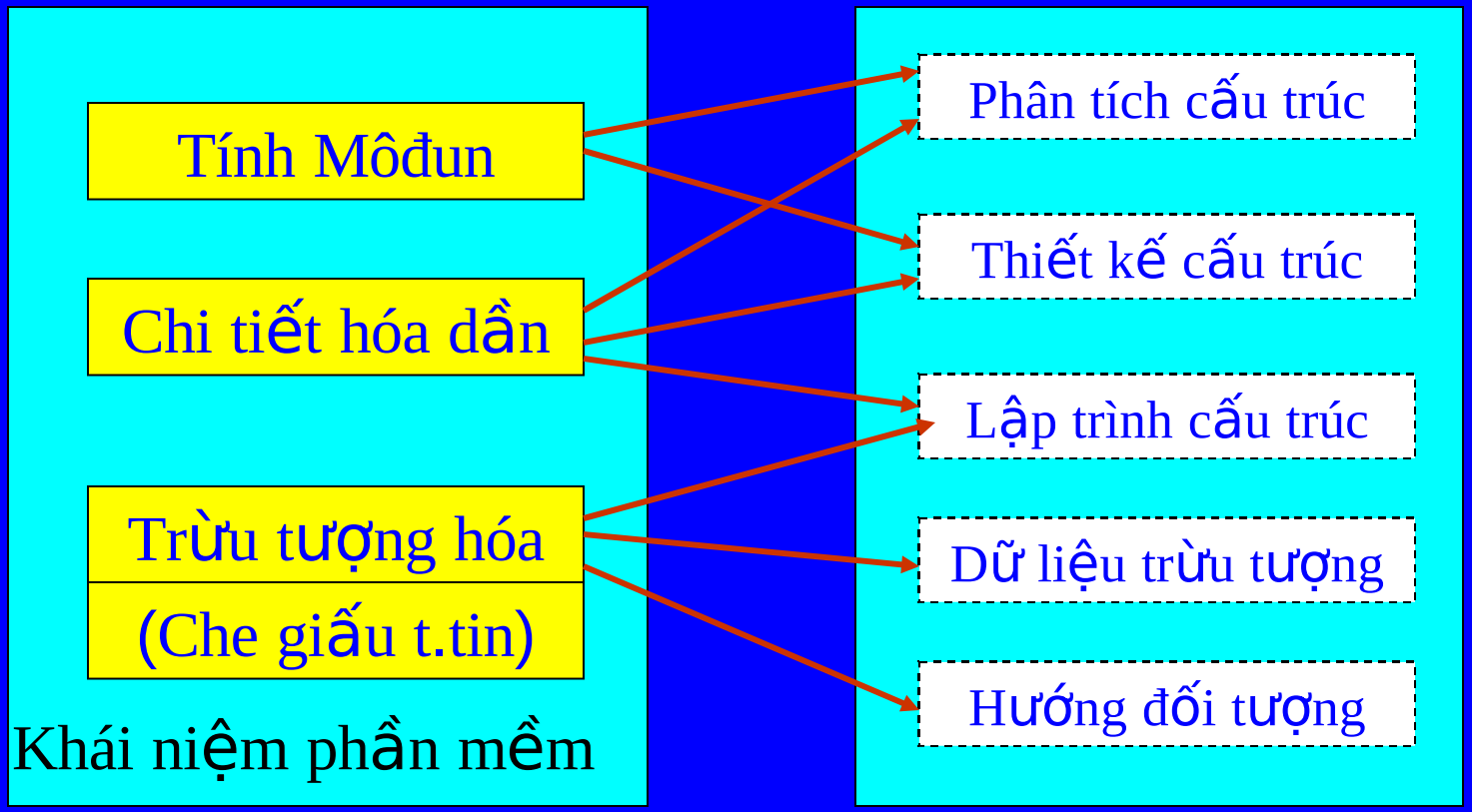
1.3 Các khái niệm

- Khi chế tác phần mềm cần nhiều kỹ thuật
  - Phương pháp luận (Methodology): những chuẩn mực cơ bản để chế tạo phần mềm với các chỉ tiêu định tính
  - Các phương pháp kỹ thuật (Techniques): những trình tự cụ thể để chế tạo phần mềm và là cách tiếp cận khoa học mang tính định lượng
- Từ phương pháp luận triển khai đến kỹ thuật

Các khái niệm (Software concepts)

- Khái niệm tính môđun (modularity concept)
- Khái niệm chi tiết hóa dần từng bước (stepwise refinement concept)
- Khái niệm trừu tượng hóa (abstraction concept): về thủ tục, điều khiển, dữ liệu
- Khái niệm che giấu thông tin (information hiding concept)
- Khái niệm hướng đối tượng (object oriented)

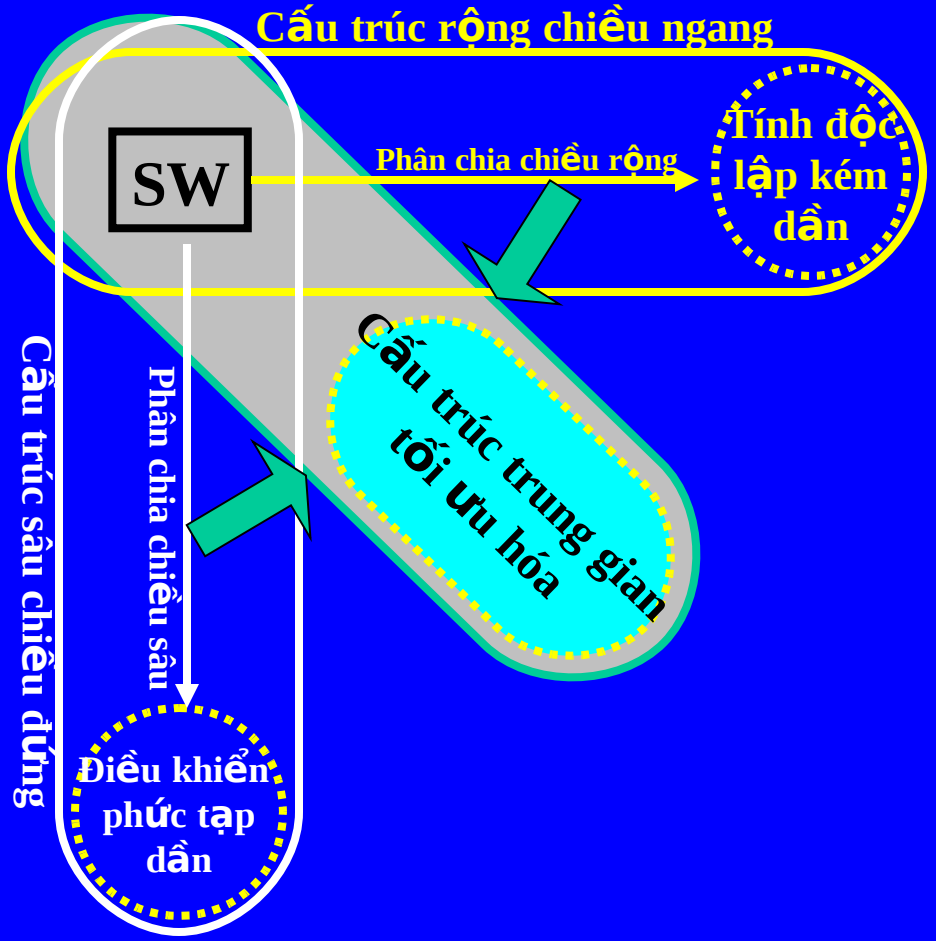
Từ phương pháp luận phần mềm sang kỹ thuật phần mềm



1.3.1 Tính môđun (Modularity)

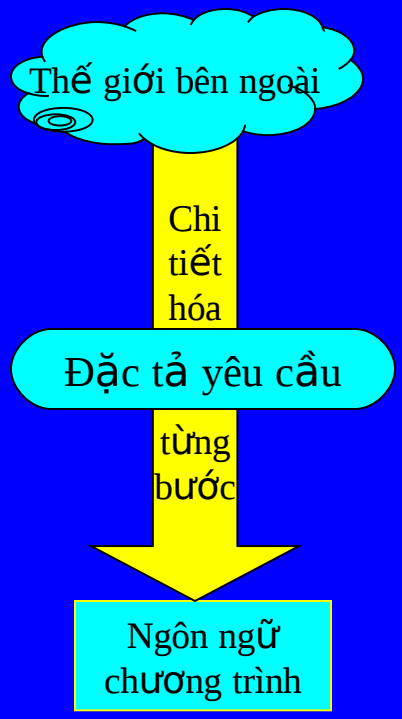
- Là khả năng phân chia phần mềm thành các môđun ứng với các chức năng, đồng thời cho phép quản lý tổng thể: khái niệm phân chia và trộn (partion and merge)
- Hai phương pháp phân chia môđun theo chiều
  - sâu (depth, thẳng đứng): điều khiển phức tạp dần
  - rộng (width, nằm ngang): môđun phụ thuộc dần
- Quan hệ giữa các môđun: qua các đối số (arguments)

Chuẩn phân chia môđun



1.3.2 Chi tiết hóa từng bước

Cách tiếp cận từ trên xuống (top-down approach)



Trừu tượng hóa mức cao:
Thế giới bên ngoài,
trạng thái chưa rõ ràng

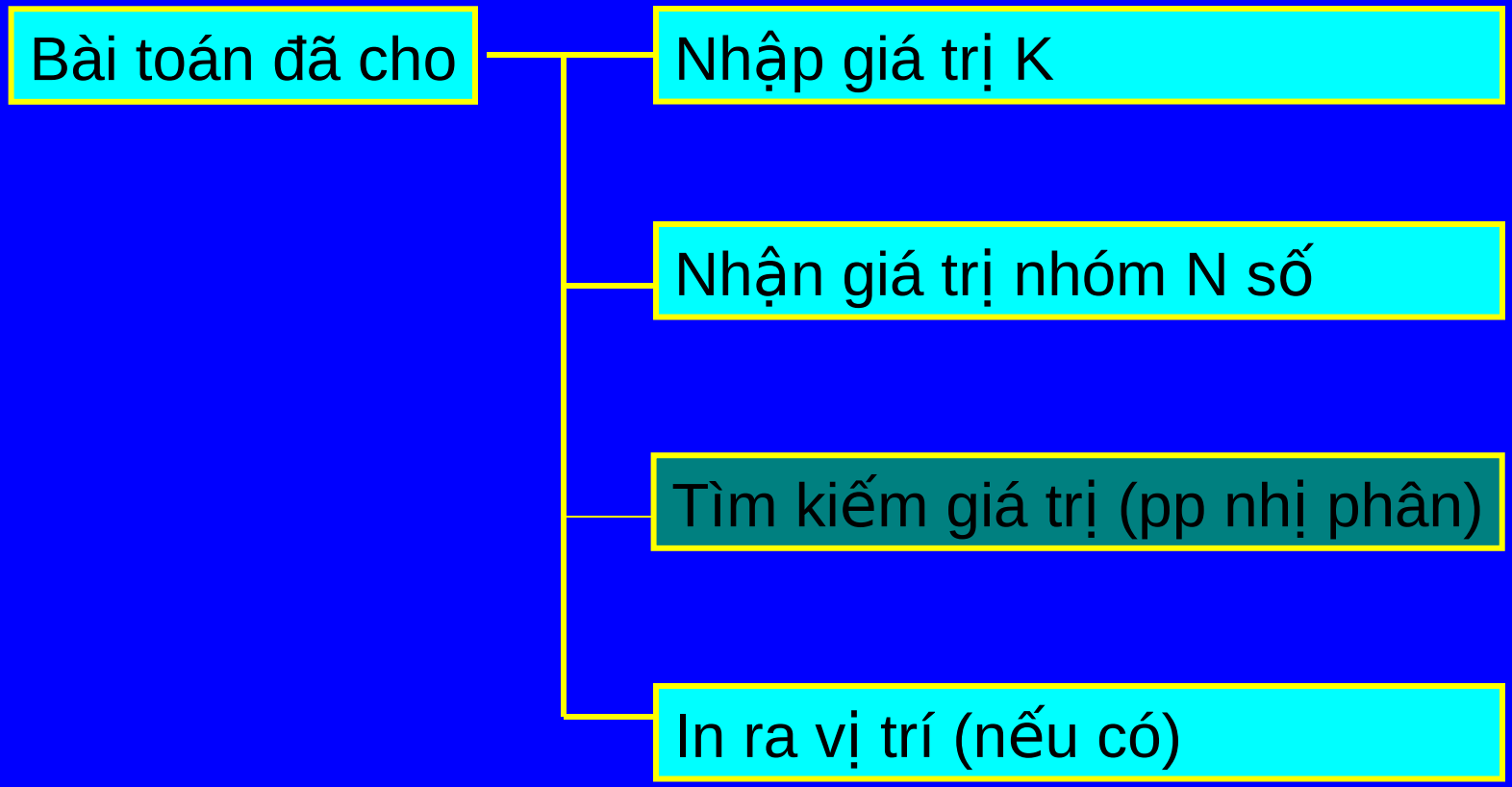
Trừu tượng hóa mức trung gian:
Xác định yêu cầu và đặc tả
những định nghĩa yêu cầu

Trừu tượng hóa mức thấp:
Từng lệnh của chương trình được
viết bởi ngôn ngữ thủ tục nào đó

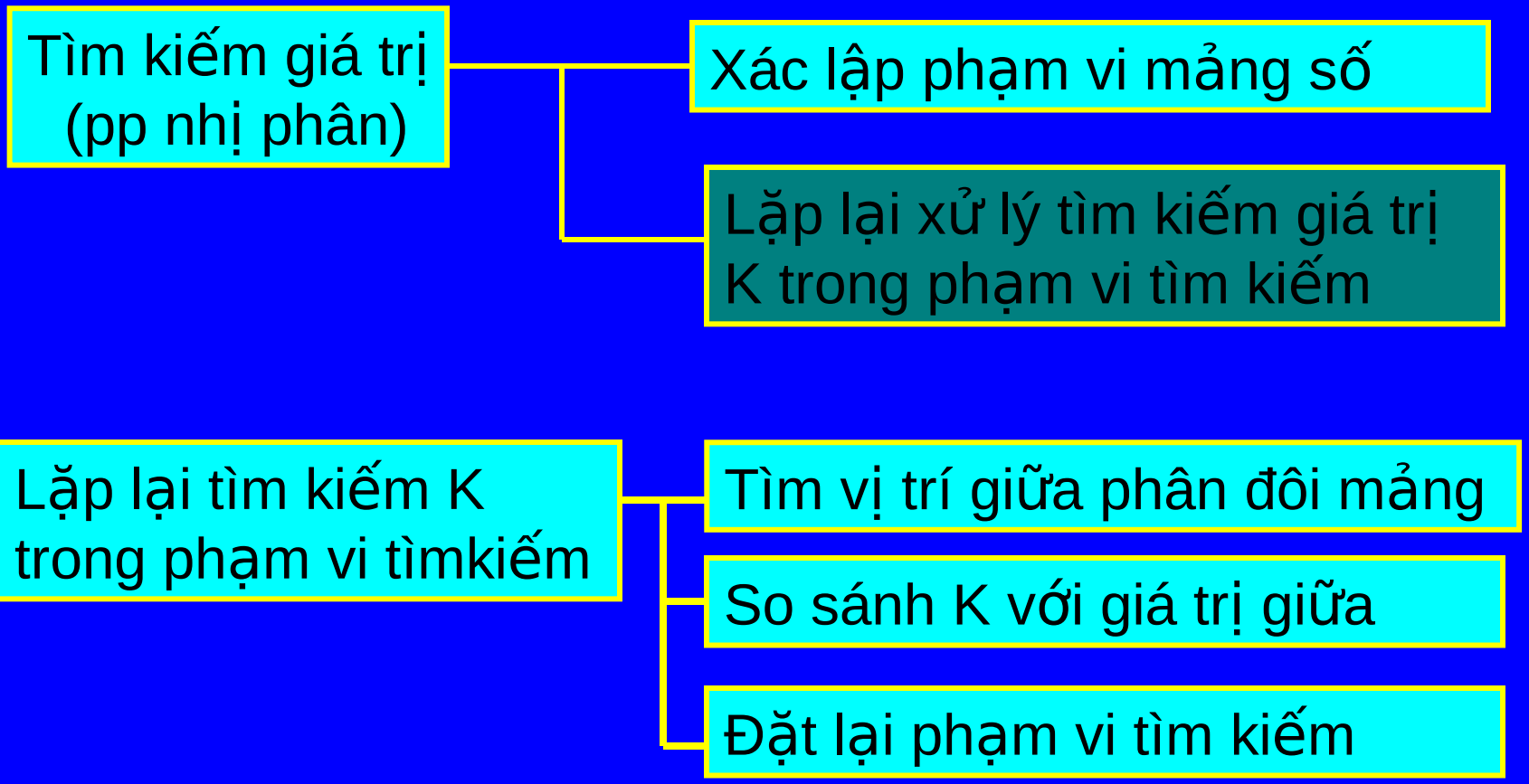
Ví dụ: Trình tự giải quyết vấn đề từ mức thiết kế chương trình đến mức lập trình

- Bài toán: từ một nhóm N số khác nhau tăng dần, hãy tìm số có giá trị bằng K (nhập từ ngoài vào) và in ra vị trí của nó
- Giải từng bước từ khái niệm đến chi tiết hóa từng câu lệnh bởi ngôn ngữ lập trình nào đó
- Chọn giải thuật tìm kiếm nhị phân (pp nhị phân)

Cụ thể hóa thủ tục qua các chức năng



Cụ thể hóa bước tiếp theo



Mức mô tả chương trình (bằng PDL)

```
Bắt đầu
    Đọc K
    Nhận giá trị cho mảng 1 chiều A(I), (I =1, 2,  . . . ,.N)
    MIN = 1
    MAX = N
    DO WHILE  (Có giá trị bằng K không, cho đến khi MIN > MAX)
        Lấy MID = (MIN + MAX) / 2
        IF  A(MID) > K THEN
                MAX = MID - 1
        ELSE
                IF A(MID) <  K THEN
                        MIN = MID + 1
                ELSE
                        In  giá trị  MID
                ENDIF
        ENDIF
    ENDDO
KếtThúc
```

1.3.3 Khái niệm Che giấu thông tin

- Để phân rã phần mềm thành các môđun một cách tốt nhất, cần tuân theo nguyên lý che giấu thông tin: “các môđun nên được đặc trưng bởi những quyết định thiết kế sao cho mỗi môđun ẩn kín đối với các môđun khác” [Parnas1972]
- Rất hữu ích cho kiểm thử và bảo trì phần mềm

Khái niệm Trừu tượng hóa

- Abstraction cho phép tập trung vấn đề ở mức tổng quát, gạt đi những chi tiết mức thấp ít liên quan
- 3 mức trừu tượng
  - Trừu tượng thủ tục: dãy các chỉ thị với chức năng đặc thù và giới hạn nào đó
  - Trừu tượng dữ liệu: tập hợp dữ liệu mô tả đối tượng dữ liệu nào đó
  - Trừu tượng điều khiển: Cơ chế điều khiển chương trình không cần đặc tả những chi tiết bên trong
- Ví dụ: *Mở cửa*. Thủ tục: Mở gồm . . .; Dữ liệu: Cửa là . . .

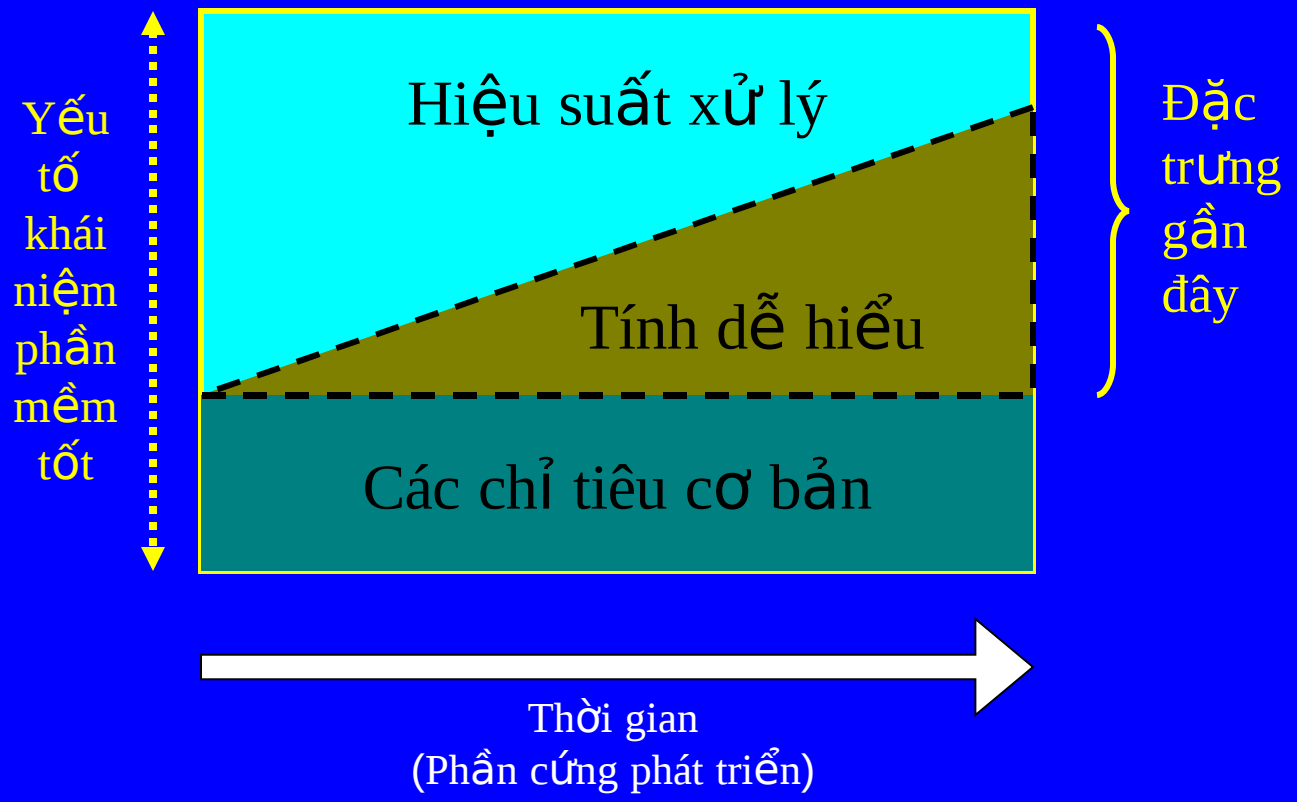
1.4 Đặc tính chung của phần mềm

- Là hàng hóa vô hình, không nhìn thấy được
- Chất lượng phần mềm: không mòn đi mà có xu hướng tốt lên sau mỗi lần có lỗi (error/bug) được phát hiện và sửa
- Phần mềm vốn chứa lỗi tiềm tàng, theo quy mô càng lớn thì khả năng chứa lỗi càng cao
- Lỗi phần mềm dễ được phát hiện bởi người ngoài

Đặc tính chung của phần mềm (tiếp)

- Chức năng của phần mềm thường biến hóa, thay đổi theo thời gian (theo nơi sử dụng)
- Hiệu ứng làn sóng trong thay đổi phần mềm
- Phần mềm vốn chứa ý tưởng và sáng tạo của tác giả/nhóm làm ra nó
- Cần khả năng "tư duy nhị phân" trong xây dựng, phát triển phần mềm
- Có thể sao chép rất đơn giản

1.5 Thế nào là phần mềm tốt ?



1.5.1 Các chỉ tiêu cơ bản

- Phản ánh đúng yêu cầu người dùng (tính hiệu quả - effectiveness)
- Chứa ít lỗi tiềm tàng
- Giá thành không vượt quá giá ước lượng ban đầu
- Dễ vận hành, sử dụng
- Tính an toàn và độ tin cậy cao

1.5.2 Hiệu suất xử lý cao

- **Hiệu suất thời gian tốt (efficiency):**
  - **Độ phức tạp tính toán thấp (Time complexity)**
  - **Thời gian quay vòng ngắn (Turn Around Time: TAT)**
  - **Thời gian hồi đáp nhanh (Response time)**
- **Sử dụng tài nguyên hữu hiệu: CPU, RAM, HDD, Internet resources, . . .**

1.5.3 Tính dễ hiểu

- Kiến trúc và cấu trúc thiết kế dễ hiểu
- Dễ kiểm tra, kiểm thử, kiểm chứng
- Dễ bảo trì
- Có tài liệu (mô tả yêu cầu, điều kiện kiểm thử, vận hành, bảo trì, FAQ, . . .) với chất lượng cao

Tính dễ hiểu: chỉ tiêu ngày càng quan trọng

1.6 Các ứng dụng phần mềm

- Phần mềm hệ thống (System SW)
- Phần mềm thời gian thực (Real-time SW)
- Phần mềm nghiệp vụ (Business SW)
- Phần mềm tính toán KH&KT (Eng.&Scie. SW)
- Phần mềm nhúng (Embedded SW)
- Phần mềm máy cá nhân (Personal computer SW)
- Phần mềm trên Web (Web-based SW)
- Phần mềm trí tuệ nhân tạo (AI SW)

Chương 2:
Khủng hoảng phần mềm
(Software Crisis)

2.1 Khủng hoảng phần mềm là gì ?

2.2 Những vấn đề (khó khăn) trong sản xuất phần mềm

2.1 Khủng hoảng phần mềm là gì?

- 10/1968 tại Hội nghị của NATO các chuyên gia phần mềm đã đưa ra thuật ngữ "Khủng hoảng phần mềm" (Software crisis). Qua hàng chục năm, thuật ngữ này vẫn được dùng và ngày càng mang tính cấp bách
- Khủng hoảng là gì ? [Webster's Dict.]
  - Điểm ngoặt trong tiến trình của bất kỳ cái gì; thời điểm, giai đoạn hoặc biến cố quyết định hay chủ chốt
  - Điểm ngoặt trong quá trình diễn biến bệnh khi trở nên rõ ràng bệnh nhân sẽ sống hay chết
- Trong phần mềm: Day dứt kinh niên (chronic affliation, by Prof. Tiechrow, Geneva, Arp. 1989)

Khủng hoảng phần mềm là gì? (tiếp)

Là sự day dứt kinh niên (kéo dài theo thời gian hoặc thường tái diễn, liên tục không kết thúc) gặp phải trong phát triển phần mềm máy tính, **như**

- **Phải làm thế nào với việc giảm chất lượng vì những lỗi tiềm tàng có trong phần mềm ?**
- **Phải xử lý ra sao khi bảo dưỡng phần mềm đã có ?**
- **Phải giải quyết thế nào khi thiếu kỹ thuật viên phần mềm?**
- **Phải chế tác phần mềm ra sao khi có yêu cầu phát triển theo qui cách mới xuất hiện ?**
- **Phải xử lý ra sao khi sự cố phần mềm gây ra những vấn đề xã hội ?**

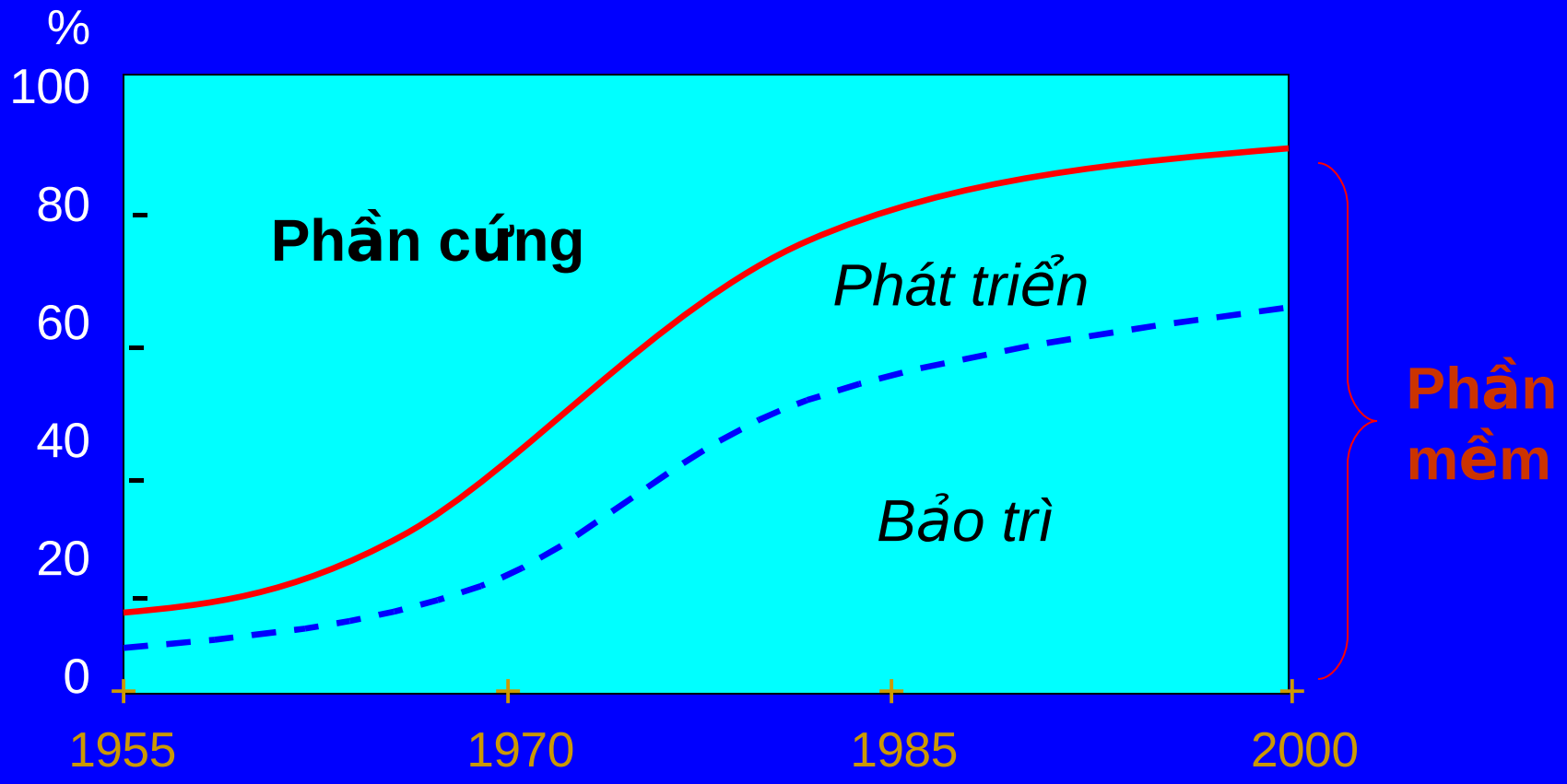
Một số yếu tố

- Phần mềm càng lớn sẽ kéo theo phức tạp hóa và tăng chi phí phát triển
- Đổi vai trò giá thành SW vs. HW
- Công sức cho bảo trì càng tăng thì chi phí cho Backlog càng lớn
- Nhân lực chưa đáp ứng được nhu cầu phần mềm
- Những phiền hà của phần mềm gây ra những vấn đề xã hội

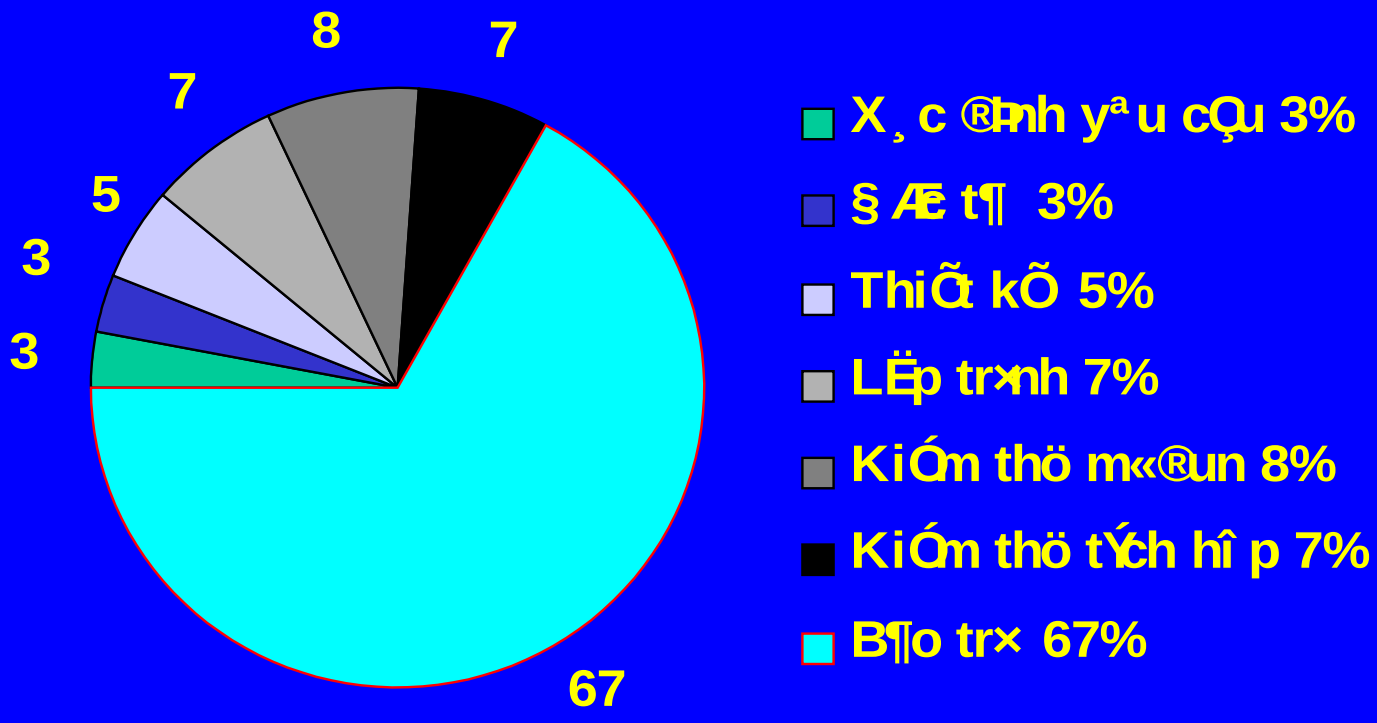
Những dự án lớn của NASA
(National Aeronautics and Space Administration)

Tên dự án	Thời gian phát triển	Tầm cỡ (người)
GEMINI	Giữa 1960	6
APOLLO (ĐỔ BỘ)	Sớm 1970	13
SPACE SHUTTLE	Giữa 1970	45

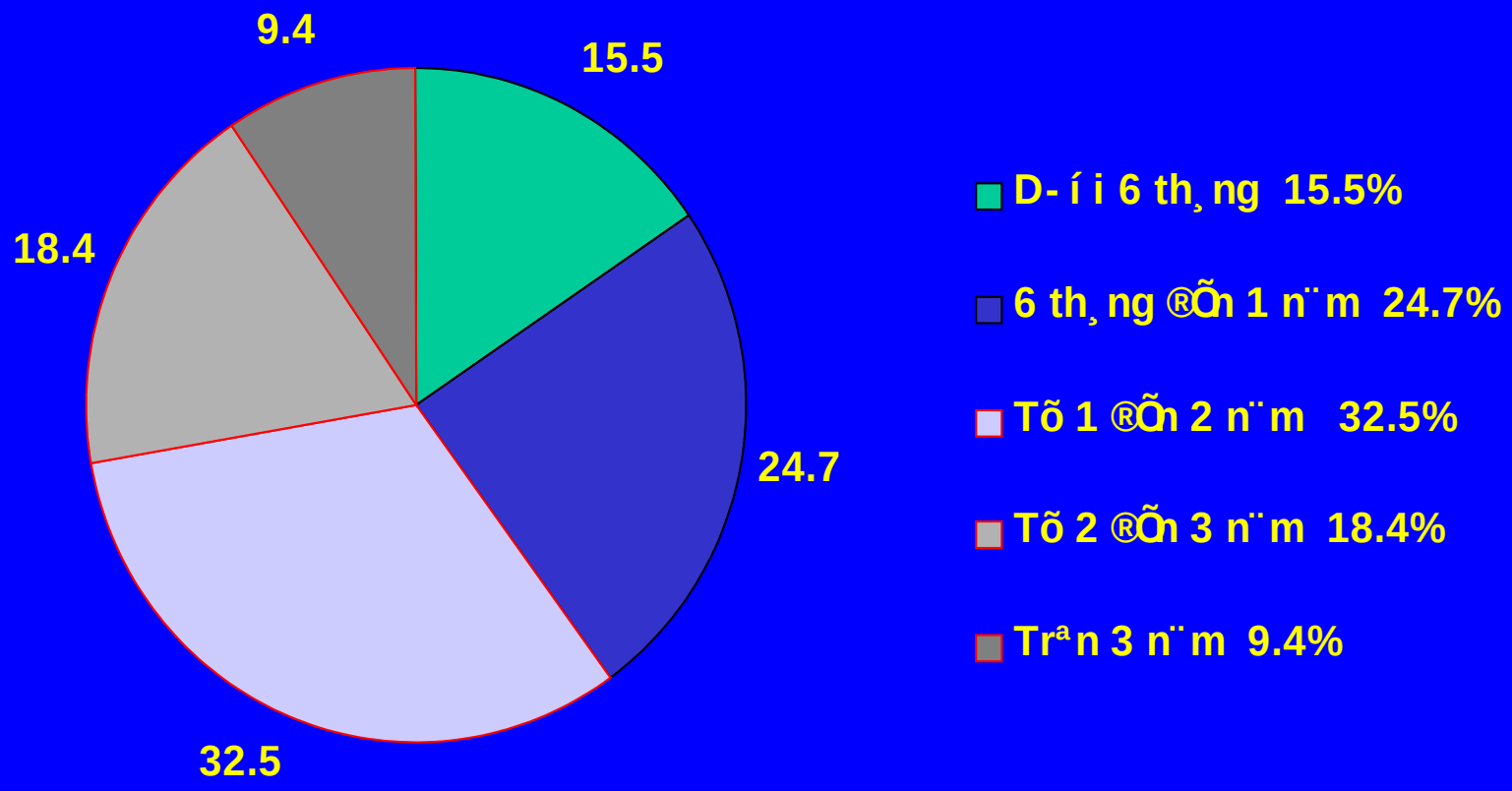
So sánh chi phí cho Phần cứng và Phần mềm



So sánh chi phí cho các pha



Backlog tại Nhật Bản năm 1985



Những vấn đề (khó khăn) trong sản xuất phần mềm

(1) Không có phương pháp mô tả rõ ràng định nghĩa yêu cầu của người dùng (khách hàng), sau khi bàn giao sản phẩm dễ phát sinh những trục trặc (troubles)

(2) Với những phần mềm quy mô lớn, tư liệu đặc tả đã cố định thời gian dài, do vậy khó đáp ứng nhu cầu thay đổi của người dùng một cách kịp thời trong thời gian đó

Những vấn đề trong sản xuất phần mềm (tiếp)

(3) Nếu không có Phương pháp luận thiết kế nhất quán mà thiết kế theo cách riêng (của công ty, nhóm), thì sẽ dẫn đến suy giảm chất lượng phần mềm (do phụ thuộc quá nhiều vào con người)

(4) Nếu không có chuẩn về làm tư liệu quy trình sản xuất phần mềm, thì những đặc tả không rõ ràng sẽ làm giảm chất lượng phần mềm

Những vấn đề trong sản xuất phần mềm (tiếp)

(5) Nếu không kiểm thử tính đúng đắn của phần mềm ở từng giai đoạn mà chỉ kiểm ở giai đoạn cuối và phát hiện ra lỗi, thì thường bàn giao sản phẩm không đúng hạn

(6) Nếu coi trọng việc lập trình hơn khâu thiết kế thì thường dẫn đến làm giảm chất lượng phần mềm

(7) Nếu coi thường việc tái sử dụng phần mềm (software reuse), thì năng suất lao động sẽ giảm

Những vấn đề trong sản xuất phần mềm (tiếp)

(8) Phần lớn trong quy trình phát triển phần mềm có nhiều thao tác do con người thực hiện, do vậy năng suất lao động thường bị giảm

(9) Không chứng minh được tính đúng đắn của phần mềm, do vậy độ tin cậy của phần mềm sẽ giảm

(10) Chuẩn về một phần mềm tốt không thể đo được một cách định lượng, do vậy không thể đánh giá được một hệ thống đúng đắn hay không

Những vấn đề trong sản xuất phần mềm (tiếp)

(11) Khi đầu tư nhân lực lớn vào bảo trì sẽ làm giảm hiệu suất lao động của nhân viên

(12) Công việc bảo trì kéo dài làm giảm chất lượng của tư liệu và ảnh hưởng xấu đến những việc khác

Những vấn đề trong sản xuất phần mềm (tiếp)

(13) Quản lý dự án lỏng lẻo kéo theo quản lý lịch trình cũng không rõ ràng

(14) Không có tiêu chuẩn để ước lượng nhân lực và dự toán sẽ làm kéo dài thời hạn và vượt kinh phí của dự án

Đây là những vấn đề phản ánh các khía cạnh khủng hoảng phần mềm, hãy tìm cách nỗ lực vượt qua để tạo ra phần mềm tốt!

Chương 3
Công nghệ học Phần mềm
(Software Engineering)

3.1 Lịch sử tiến triển Công nghệ học phần mềm

3.2 Sự tiến triển của các phương pháp thiết kế phần mềm

3.3 Định nghĩa Công nghệ học phần mềm

3.4 Vòng đời của phần mềm

3.5 Quy trình phát triển phần mềm

3.1 Lịch sử tiến triển của CNHPM

- **Nửa đầu 1960**: ít quan tâm đến phần mềm, chủ yếu tập trung nâng cao tính năng và độ tin cậy của phần cứng
- **Giữa những năm 1960**: Phát triển hệ điều hành như phần mềm lớn (IBM OS/360, EC OS). Xuất hiện nhu cầu về quy trình phát triển phần mềm lớn và quy trình gỡ lỗi, kiểm thử trong phạm vi giới hạn

Lịch sử tiến triển của CNHPM (tiếp)

- **Năm 1968: Tại Tây Đức, Hội nghị khoa học của NATO đã đưa ra từ "Software Engineering". Bắt đầu bàn luận về khủng khoảng phần mềm và xu hướng hình thành CNHPM như một chuyên môn riêng**
- **Nửa cuối 1960: IBM đưa ra chính sách phân biệt giá cả giữa phần cứng và phần mềm. Từ đó, ý thức về phần mềm ngày càng cao. Bắt đầu những nghiên cứu cơ bản về phương pháp luận lập trình**

Lịch sử tiến triển của CNHPM (tiếp)

- **Nửa đầu những năm 1970**: Nhằm nâng cao chất lượng phần mềm, không chỉ có các nghiên cứu về lập trình, kiểm thử, mà có cả những nghiên cứu đảm bảo tính tin cậy trong quy trình sản xuất phần mềm. Kỹ thuật: lập trình cấu trúc hóa, lập trình môđun, thiết kế cấu trúc hóa, vv
- **Giữa những năm 1970**: Hội nghị quốc tế đầu tiên về CNHPM được tổ chức (1975): International Conference on SE (ICSE)

Lịch sử tiến triển của CNHPM (tiếp)

- **Nửa sau những năm 1970: Quan tâm đến mọi pha trong quy trình phát triển phần mềm, nhưng tập trung chính ở những pha đầu. ICSE tổ chức lần 2, 3 và 4 vào 1976, 1978 và 1979**
  - **Nhật Bản có "Kế hoạch phát triển kỹ thuật sản xuất phần mềm" từ năm 1981**
  - **Cuộc "cách tân sản xuất phần mềm" đã bắt đầu trên phạm vi các nước công nghiệp**

Lịch sử tiến triển của CNHPM (tiếp)

- **Nửa đầu những năm 1980: Trình độ học vấn và ứng dụng CNHPM được nâng cao, các công nghệ được chuyển vào thực tế. Xuất hiện các sản phẩm phần mềm và các công cụ khác nhau làm tăng năng suất sản xuất phần mềm đáng kể**
  - **ICSE tổ chức lần 5 và 6 năm 1981 và 1982 với trên 1000 người tham dự mỗi năm**
  - **Nhật Bản sang "Kế hoạch phát triển các kỹ thuật bảo trì phần mềm" (1981-1985)**

Lịch sử tiến triển của CNHPM (tiếp)

- **Nửa cuối những năm 1980 đến nay: Từ học vấn sang nghiệp vụ! Chất lượng phần mềm tập trung chủ yếu ở *tính năng suất, độ tin cậy và tính bảo trì.* Nghiên cứa hỗ trợ tự động hóa sản xuất phần mềm**
  - **Nhật Bản có "Kế hoạch hệ thống công nghiệp hóa sản xuất phần mềm"(SIGMA: Software Industrialized Generator & Maintenance Aids, 1985-1990)**
  - **Nhiều trung tâm, viện nghiên cứu CNHPM ra đời. Các trường đưa vào giảng dạy SE**

Hiện nay

- Công nghiệp hóa sản xuất phần mềm bằng cách đưa những kỹ thuật công nghệ học (Engineering techniques) thành cơ sở khoa học của CNHPM
- Thể chế hóa lý luận trong sản xuất phần mềm và ứng dụng những phương pháp luận một cách nhất quán
- Tăng cường nghiên cứu và tạo công cụ trợ giúp sản xuất phần mềm

3.2 Sự tiến triển của các phương pháp thiết kế phần mềm

- Phương pháp luận trong CNHPM: bắt đầu từ những năm 1970
- Trong phát triển phần mềm: nâng cao năng suất, độ tin cậy, giá thành - tính năng (productivity, reliability, cost-performance)
- Tiến triển phương pháp thiết kế: Sơ khởi, Trưởng thành, Phát triển và Biến đổi

Sơ khởi: nửa đầu 1970

- Khái niệm về tính môđun, cụ thể hóa từng bước trong phương pháp luận thiết kế
- N. Wirth: Chi tiết hóa từng giai đoạn. Thiết kế trên xuống. Lập trình môđun

Trưởng thành: nửa cuối 1970

- Phương pháp luận về quy trình thiết kế phần mềm với phương pháp phân chia môđun và thiết kế trong từng môđun.
- L.L. Constantine, 1974: Thiết kế cấu trúc hóa (phân chia môđun);
- E.W. Dijkstra, 1972: Lập trình cấu trúc hóa (trong môđun) . Phương pháp M.A. Jackson (1975) và J.D. Warnier (1974)
- Trừu tượng hóa dữ liệu: B.H. Liskov (1974);D.L. Parnas (1972)

Phát triển: nửa đầu 1980

- Triển khai các công cụ hỗ trợ phát triển phần mềm dựa trên các phương pháp và kỹ thuật đưa ra những năm 1970
- Bộ khởi tạo chương trình (program generators: pre-compiler; graphics-input editors, etc.)
- Ngôn ngữ đối thoại đơn giản (4GL, DB SQL)
- Hệ trợ giúp: Hệ trợ giúp kiểm thử; Hệ trợ giúp quản lý thư viện; Hệ trợ giúp tái sử dụng

Biến đổi: nửa cuối 1980 đến nay

- Đưa ra các môi trường mới về phát triển phần mềm. Triển khai mới về kết hợp giữa CNHPM và CNH Tri thức (Knowledge Engineering)
- Triển khai những môi trường bậc cao về phát triển phần mềm; Tự động hóa sản xuất phần mềm; Chế phần mềm theo kỹ thuật chế thử (Prototyping); Lập trình hướng đối tượng - OOP; Hướng thành phần; Hỗ trợ phát triển phần mềm từ các hệ chuyên gia, vv

Hình thái sản xuất Phần mềm

Đưa ra các kỹ thuật, phương pháp luận

↓

ứng dụng thực tế vào từng quy trình

↓

Cải biên, biến đổi vào từng sản phẩm và công cụ phần mềm (máy tính hóa từng phần)

↓

Tổng hợp, hệ thống hóa cho từng loại công cụ (Máy tính hóa toàn bộ quy trình sản xuất phần mềm)

↓

Hướng tới sản xuất phần mềm tự động

3.3 Định nghĩa Công nghệ học phần mềm

- **Bauer [1969]:** CNHPM là việc thiết lập và sử dụng các nguyên tắc công nghệ học đúng đắn dùng để thu được phần mềm một cách kinh tế vừa tin cậy vừa làm việc hiệu quả trên các máy thực
- **Parnas [1987]:** CNHPM là việc xây dựng phần mềm nhiều phiên bản bởi nhiều người
- **Ghezzi [1991]:** CNHPM là một lĩnh vực của khoa học máy tính, liên quan đến xây dựng các hệ thống phần mềm vừa lớn vừa phức tạp bởi một hay một số nhóm kỹ sư

Định nghĩa CNHPM (tiếp)

- **IEEE [1993]:** CNHPM là
  (1) việc áp dụng phương pháp tiếp cận có hệ thống, bài bản và được lượng hóa trong phát triển, vận hành và bảo trì phần mềm;
  (2) nghiên cứu các phương pháp tiếp cận được dùng trong (1)
- **Pressman [1995]:** CNHPM là bộ môn tích hợp cả quy trình, các phương pháp, các công cụ để phát triển phần mềm máy tính

Định nghĩa CNHPM (tiếp)

- **Sommerville [1995]:** CNHPM là lĩnh vực liên quan đến lý thuyết, phương pháp và công cụ dùng cho phát triển phần mềm
- **K. Kawamura [1995]:** CNHPM là lĩnh vực học vấn về các kỹ thuật, phương pháp luận công nghệ học (lý luận và kỹ thuật được hiện thực hóa trên những nguyên tắc, nguyên lý nào đó) trong toàn bộ quy trình phát triển phần mềm nhằm nâng cao cả chất và lượng của sản xuất phần mềm

Định nghĩa CNHPM (tiếp)

Công nghệ học phần mềm là lĩnh vực khoa học về các phương pháp luận, kỹ thuật và công cụ tích hợp trong quy trình sản xuất và vận hành phần mềm nhằm tạo ra phần mềm với những chất lượng mong muốn **[Software Engineering is a scientìic field to deal with methodologies, techniques and tools integrated in software production-maintenance process to obtain software with desired qualities]**

Công nghệ học trong CNHPM ?

(1) Như các ngành công nghệ học khác, CNHPM cũng lấy các phương pháp khoa học làm cơ sở

(2) Các kỹ thuật về thiết kế, chế tạo, kiểm thử và bảo trì phần mềm đã được hệ thống hóa hóa thành phương pháp luận và hình thành nên CNHPM

(3) Toàn bộ quy trình quản lý phát triển phần mềm gắn với khái niệm vòng đời phần mềm, được mô hình hóa với những kỹ thuật và phương pháp luận trở thành các chủ đề khác nhau trong CNHPM

Công nghệ học trong CNHPM ? (tiếp)

(4) Trong vòng đời phần mềm không chỉ có chế tạo mà bao gồm cả thiết kế, vận hành và bảo dưỡng (tính quan trọng của thiết kế và bảo dưỡng)

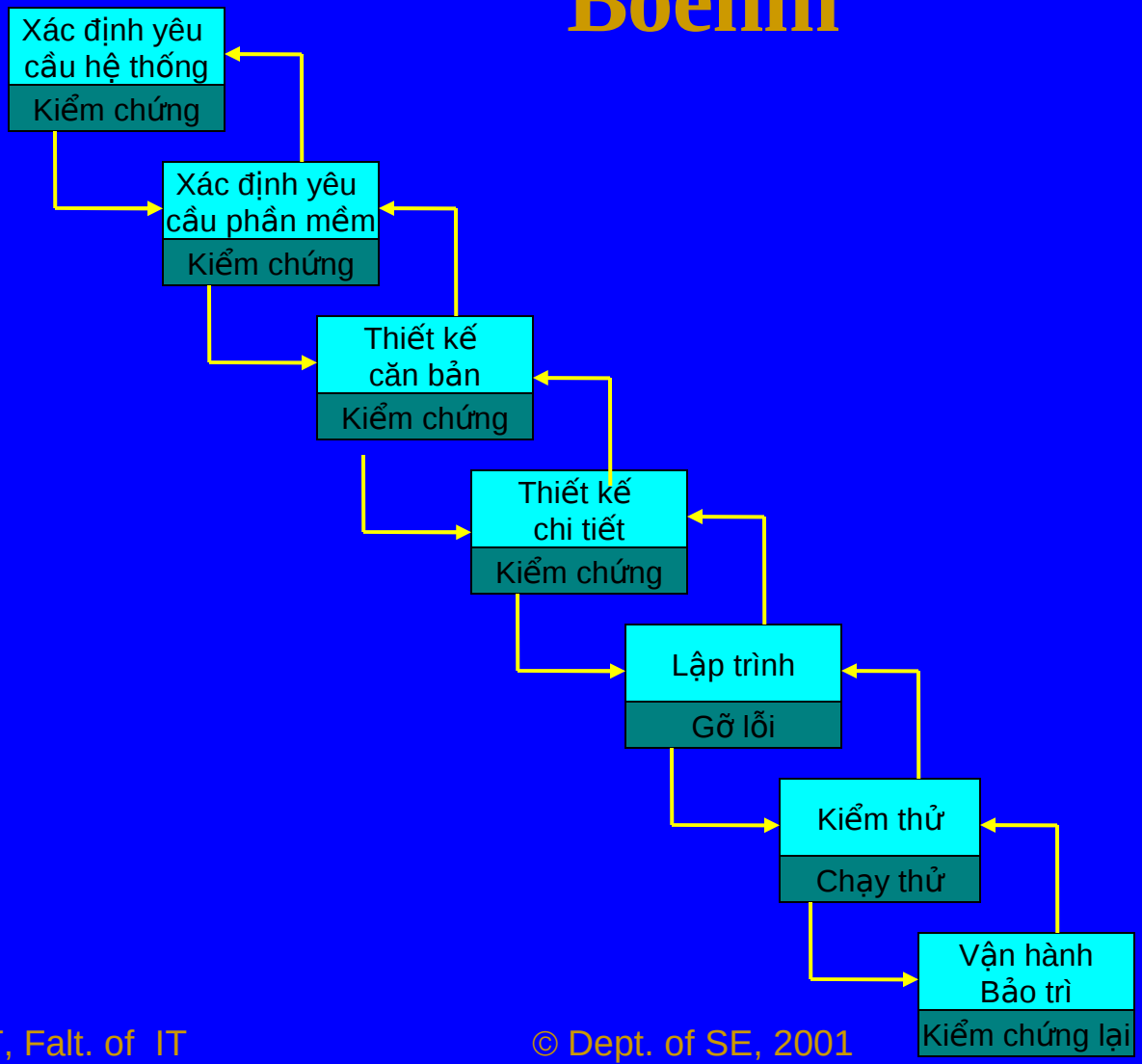
(5) Trong khái niệm phần mềm, không chỉ có chương trình mà cả tư liệu về phần mềm

(6) Cách tiếp cận công nghệ học (khái niệm công nghiệp hóa) thể hiện ở chỗ nhằm nâng cao năng suất (tính năng suất) và độ tin cậy của phần mềm, đồng thời giảm chi phí giá thành

3.4 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

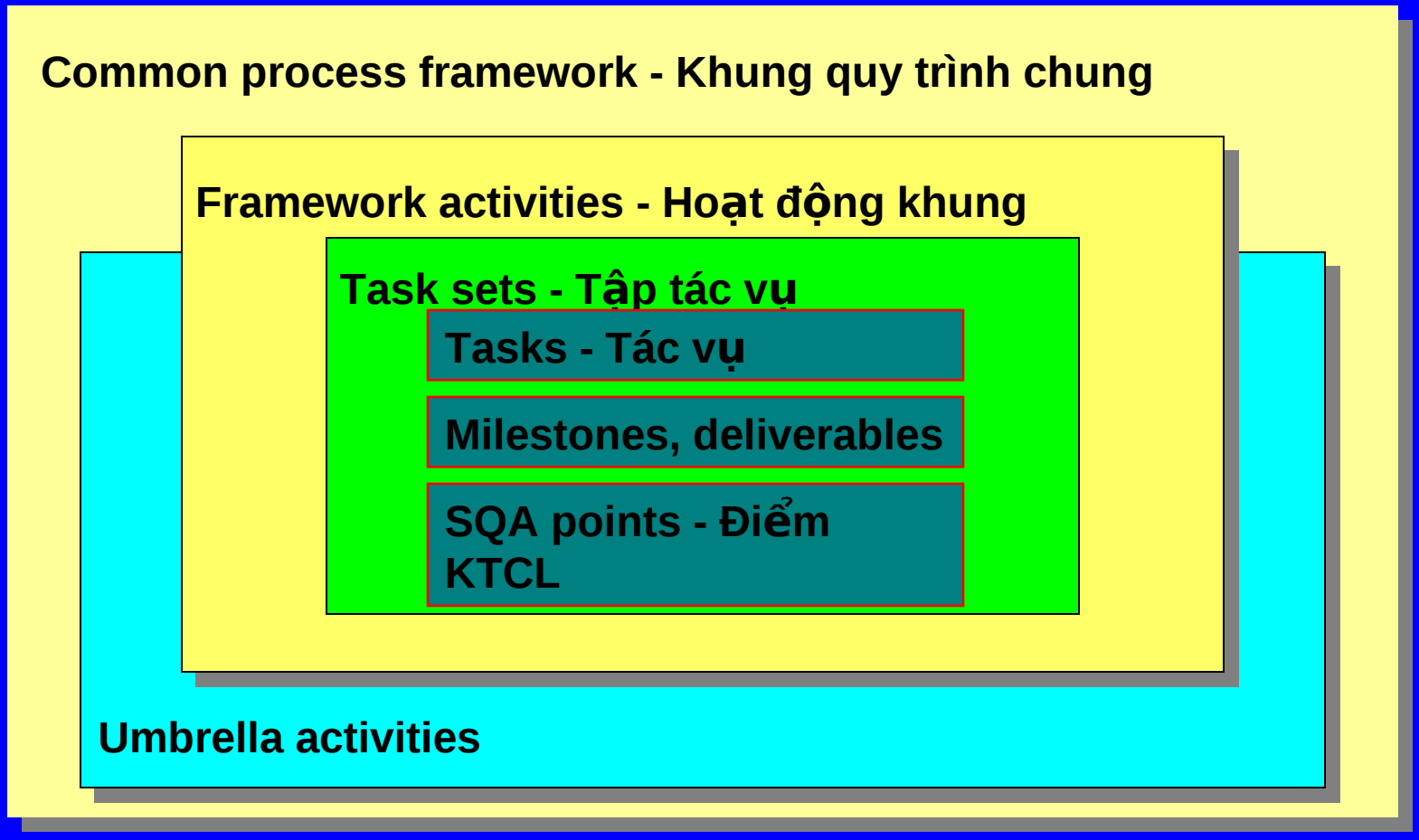
(7) Cần chuẩn hóa mẫu biểu, cách ghi chép tạo tư liệu cho từng pha, nhằm đảm bảo chất lượng phần mềm

(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

Pha	Nội dung giới thiệu	Phương pháp kỹ thuật
Xác định yêu cầu	Sản phẩm yêu cầu người dùng Xác định yêu cầu phần mềm	Phương pháp hướng cấu trúc Phương pháp hướng đối tượng
Thiết kế kiến trúc	Thiết kế cấu trúc phần mềm Thiết kế giao diện người dùng	Thiết kế hướng cấu trúc Thiết kế hướng đối tượng
Thiết kế chi tiết	Lựa chọn cấu trúc dữ liệu thiết kế thuật toán Thiết kế giao diện (nếu có)	Phương pháp hướng cấu trúc Phương pháp hướng đối tượng Visual
Lập trình	Mã hóa theo ngôn ngữ lập trình	Mã hóa theo cấu trúc
Sửa lỗi và phát triển	Kiểm tra để sửa lỗi và phát triển	Phương pháp kiểm thử hộp trắng
Vận hành Bảo trì	Sử dụng vận hành phần mềm Phát triển Bảo trì phần mềm	Chưa có

3.5 Quy trình phát triển phần mềm



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

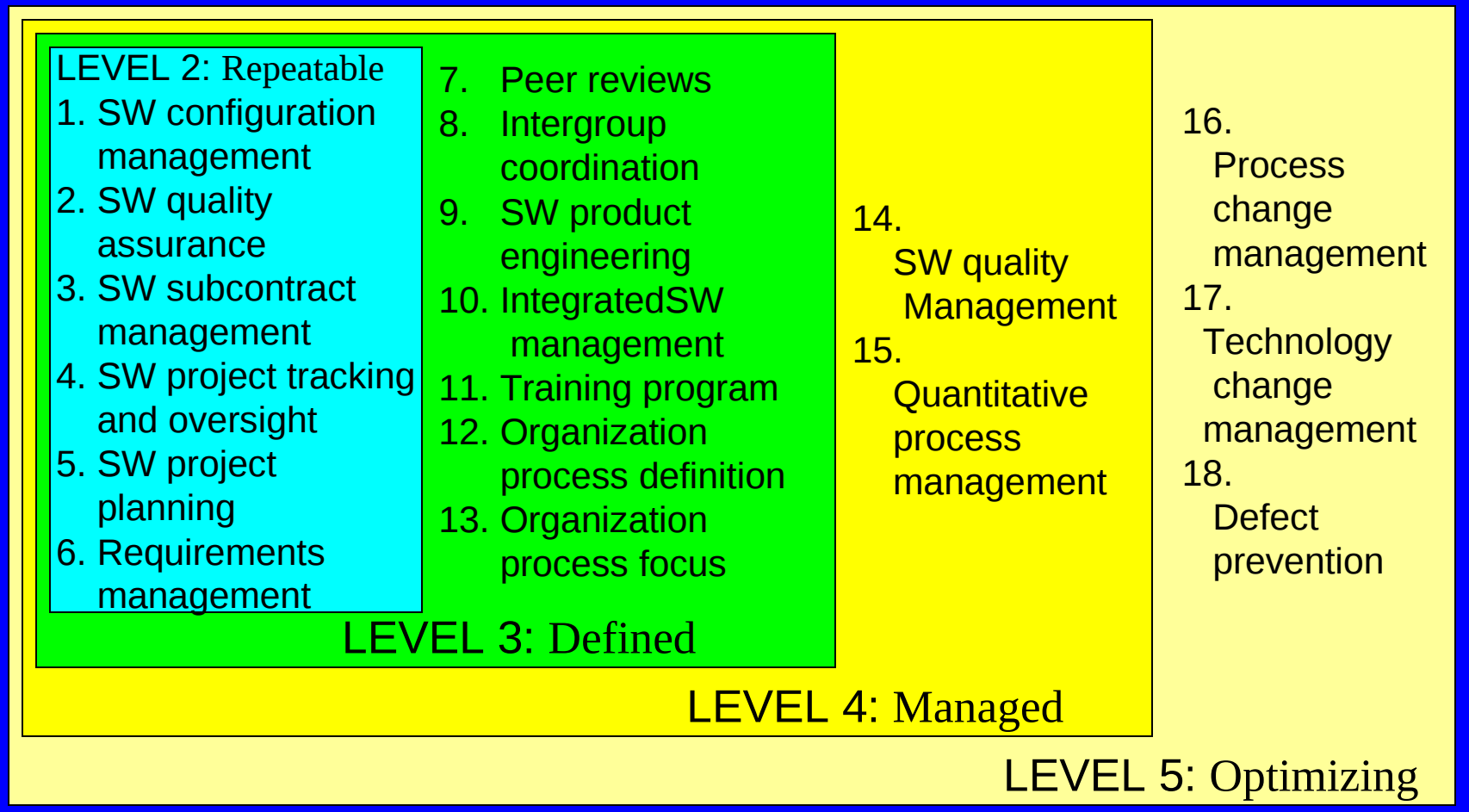
- **Level 1: Initial (Khởi đầu). Few processes are defined. Success depends on individual effort**
- **Level 2: Repeatable (Lặp lại). Basic project management processes. Repeat earlier succeses on projects with similar applications**
- **Level 3: Defined (Xác định). Use a documented and approved version of the organization's process for developing and supporting software**

CMM (cont.)

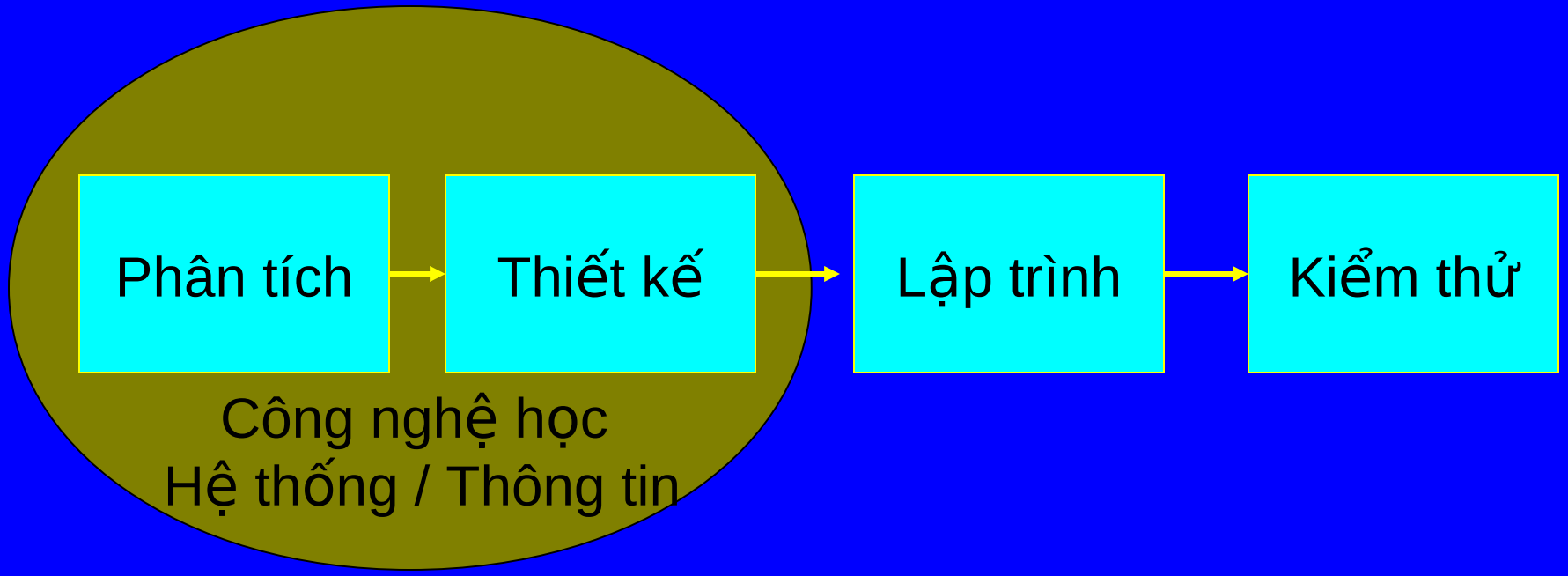
- **Level 4: Managed (Quản trị). Both SW process and products are quantitatively understood and controlled using detailed measures**
- **Level 5: Optimizing (Tối ưu). Continuous process improvement is enabled by quantitative feedback from the process and from testing innovative ideas and technologies**

18 key process areas (KPAs) for CMM

18 KPAs of CMM



3.5.2 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 được 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

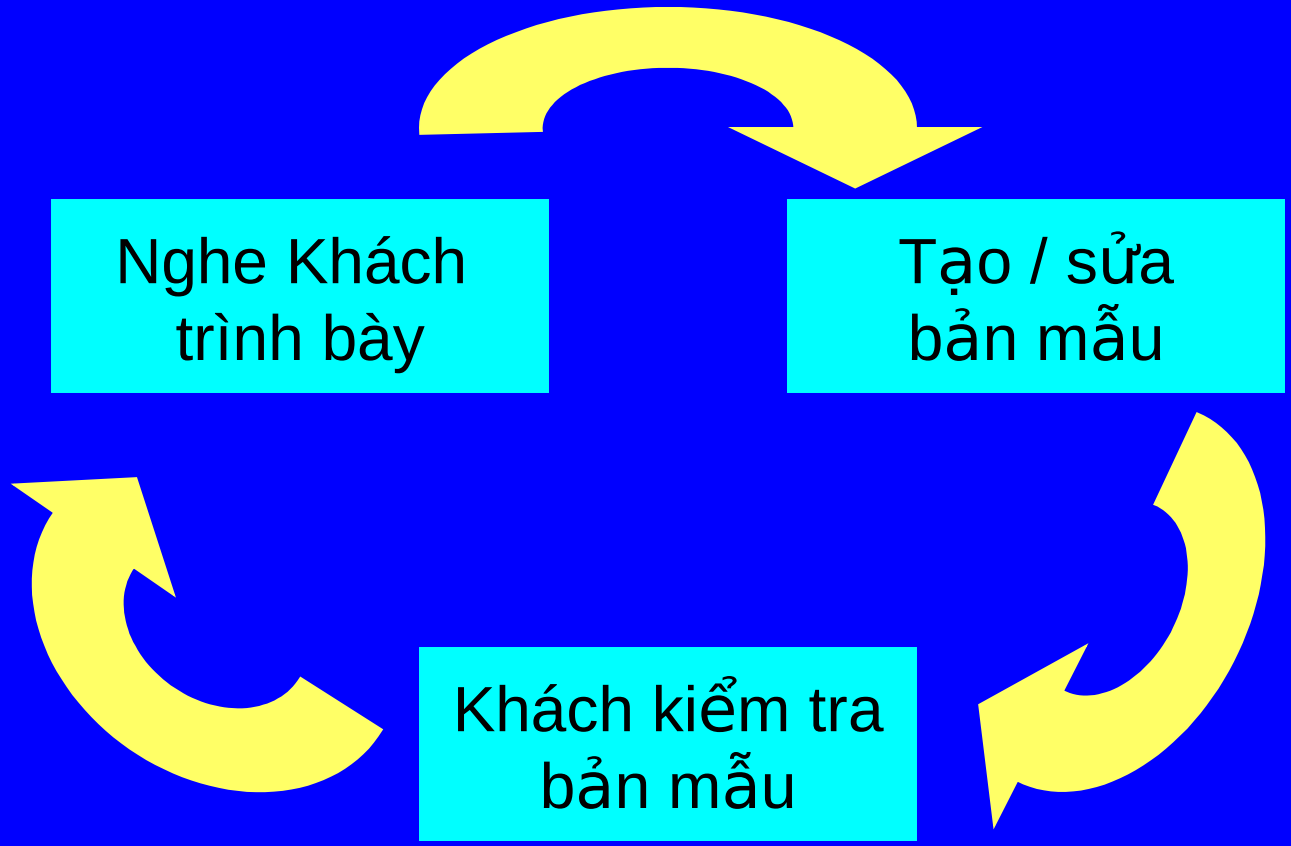
Mô hình tuyến tính

- **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!

3.5.3 Mô hình chế thử (Prototyping model)



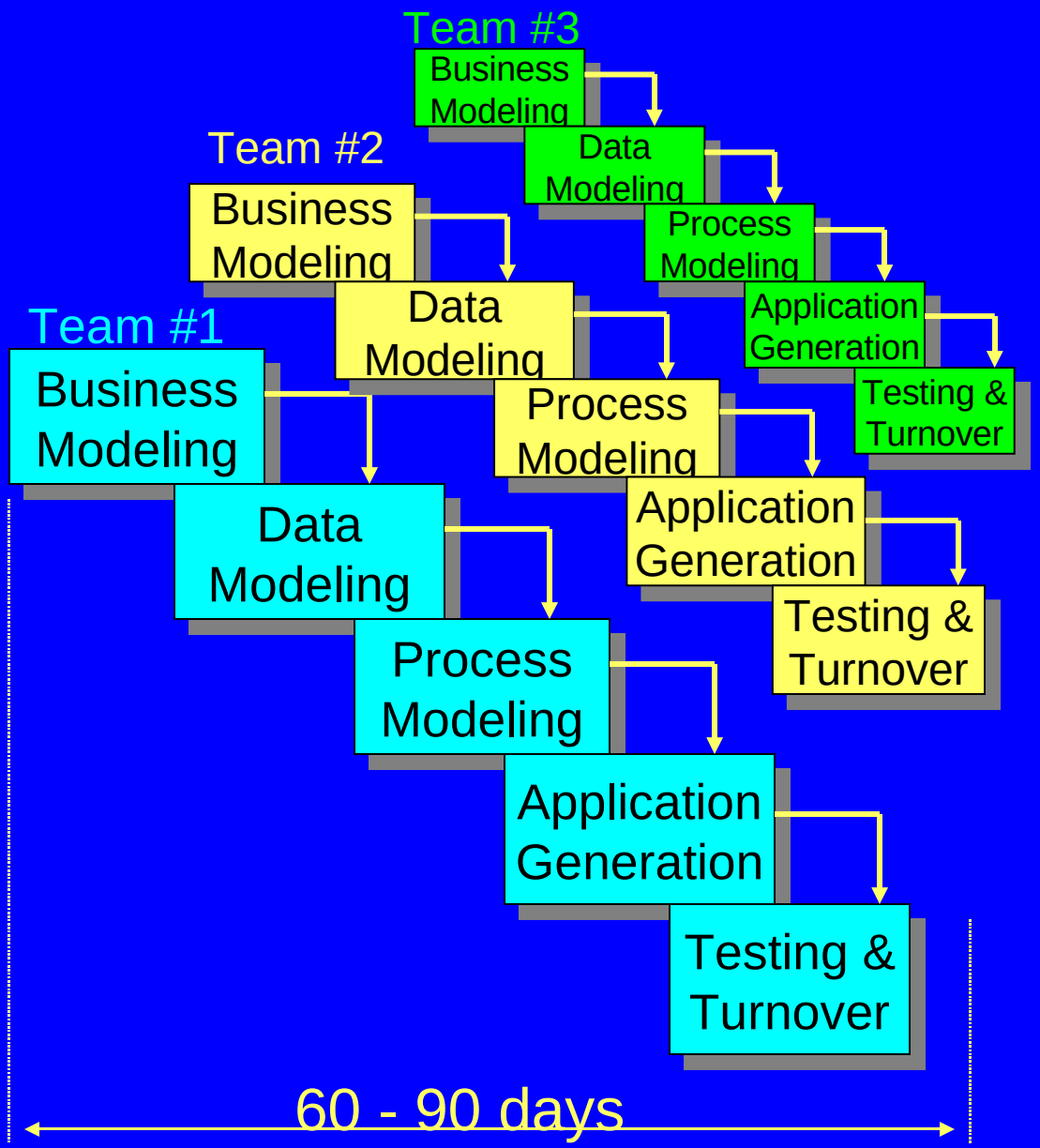
Mô hình chế thử: Khi nào ?

- 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

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

- Là quy trình phát triển phần mềm gia tăng, tăng dần từng bước (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

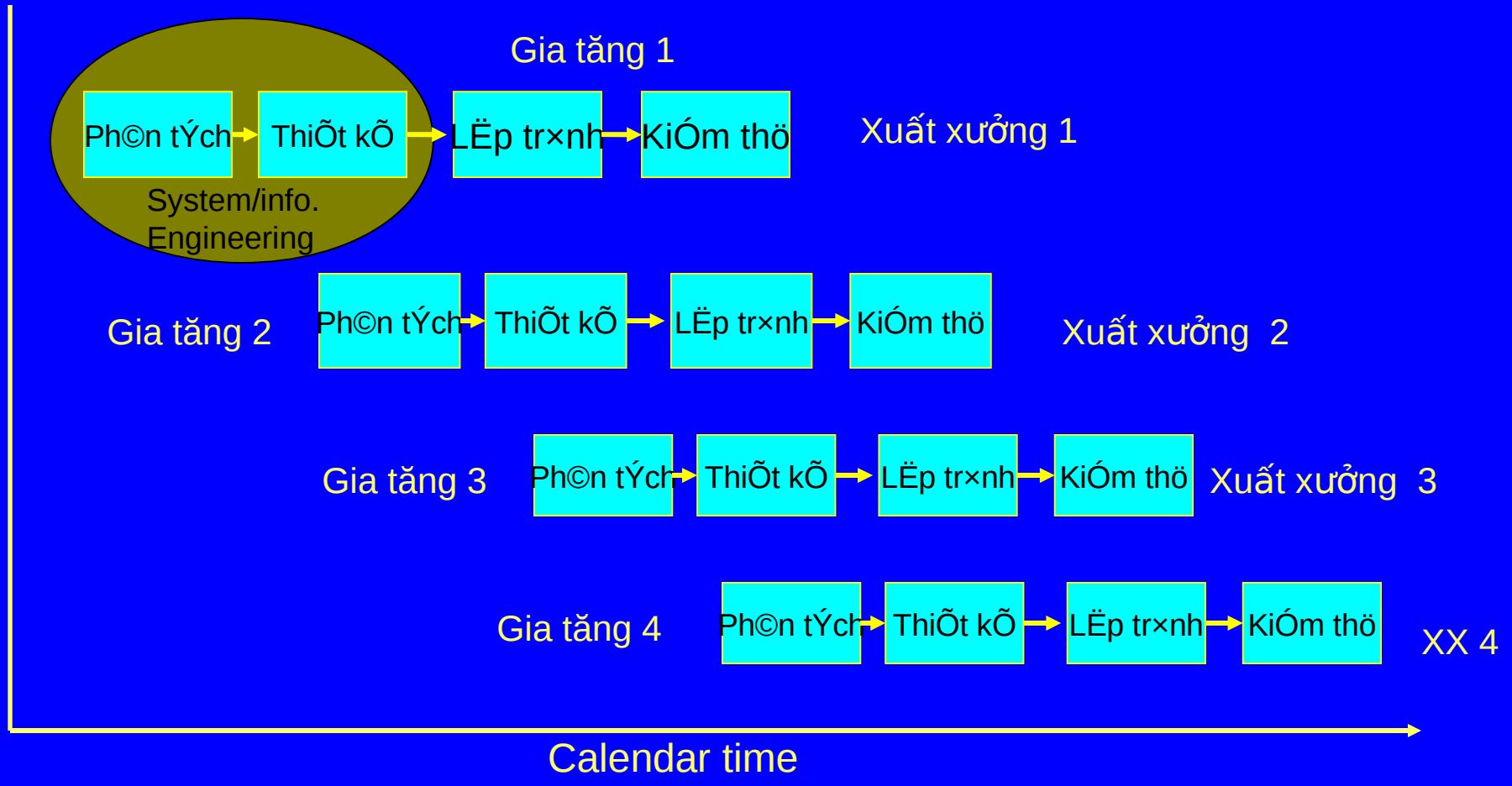
3.5.5 Các mô hình tiến hóa: gia tăng, xoắn ốc, xoắn WINWIN, ...

- 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

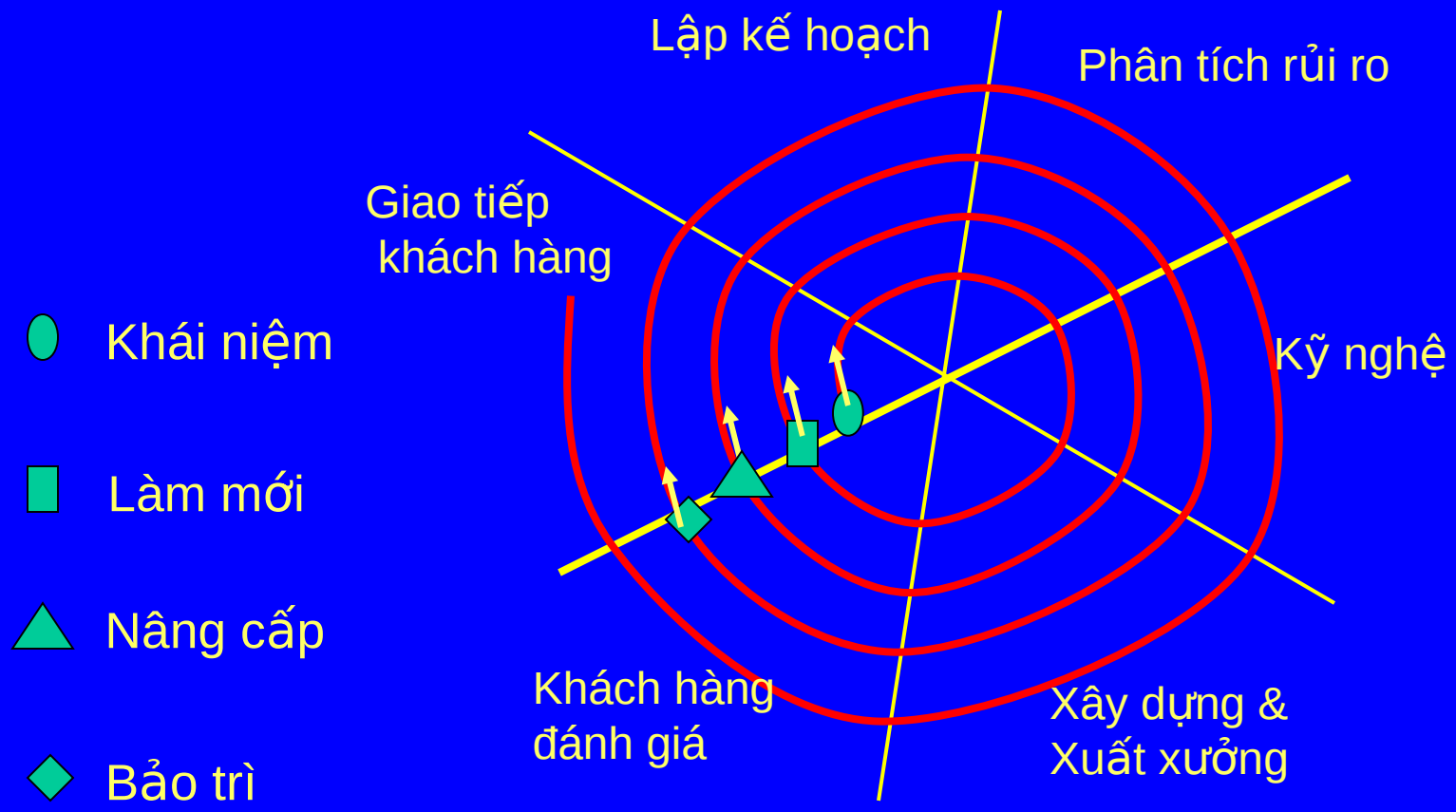
Mô hình gia tăng (The incremental model)

- 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 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

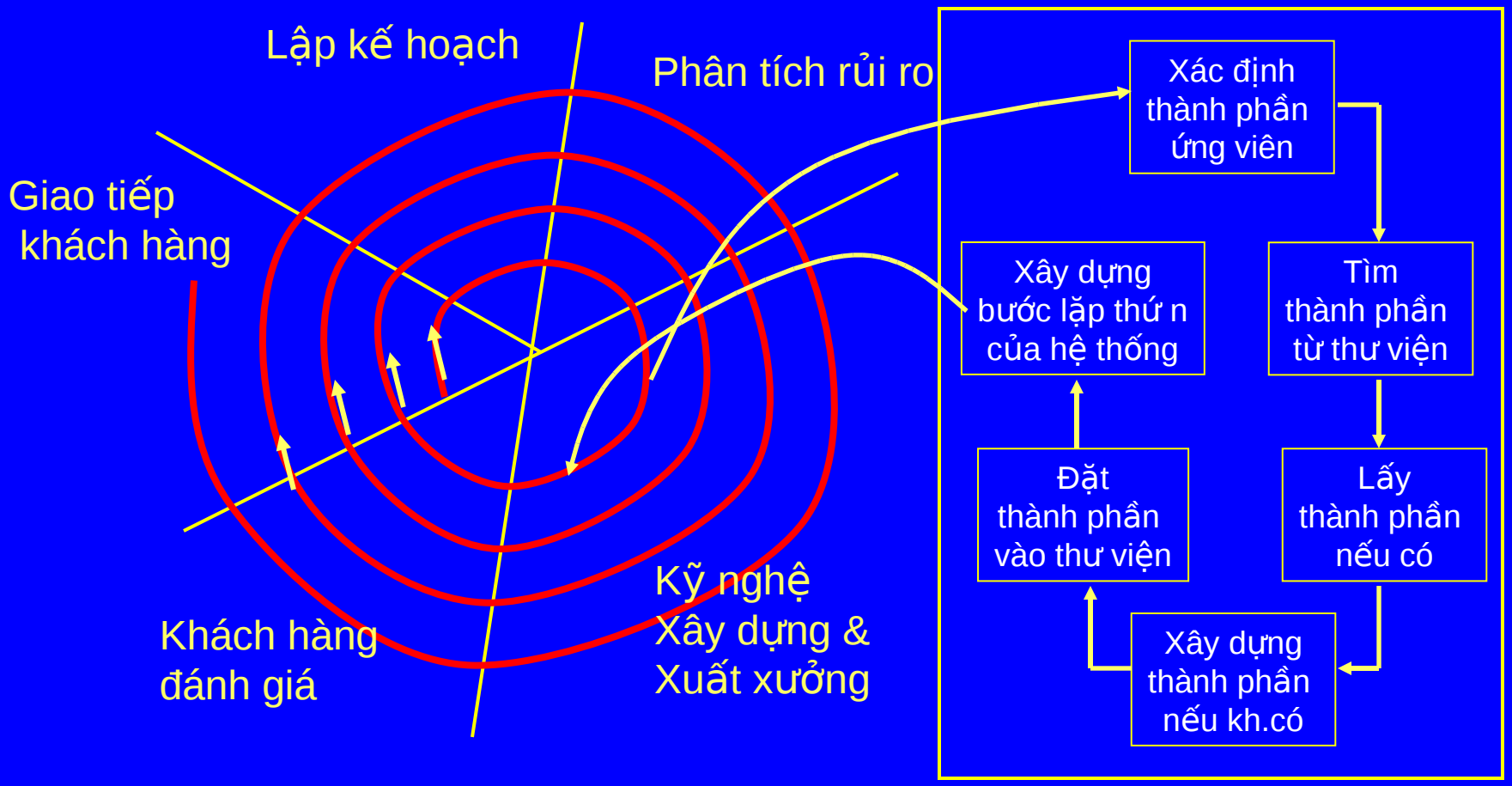
Mô hình phát triển đồng thời (The concurrent development model)

- 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

3.5.6 Mô hình theo 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 theo thành phần



3.5.7 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

3.5.8 Các kỹ thuật thế hệ 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: How ?

- 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 ⇒ cần kỹ năng của kỹ sư phần mềm
- Tương lai: 4GT với mô hình theo thành phần

3.5.9 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

Bài tập Phần I và Đồ án I

- Xem lại các khái niệm, mô hình của phần mềm và CNHPM
- Đồ án môn học I (cho 13 nhóm, nạp báo cáo, tư liệu tìm được trên Web và thư viện):
  - Tìm hiểu và viết báo cáo, trình bày về mô hình phát triển phần mềm (10 mô hình / 10 nhóm)
  - Chuẩn ISO 9001 cho SE
  - Chuẩn CMM (www.sei.com)
  - Các kỹ thuật lập trình (cấu trúc, mô đun, . . .)