

PHẦN I: GIỚI THIỆU CHUNG VỀ CÔNG NGHỆ PHẦN MỀM

- I. Bản chất phần mềm
- II. Những vấn đề trong phát triển phần mềm
- III. Quy trình phát triển phần mềm

1

PHẦN I: GIỚI THIỆU CHUNG VỀ CÔNG NGHỆ PHẦN MỀM

- I. Bản chất phần mềm
 1. Định nghĩa chung về phần mềm
 2. Kiến trúc phần mềm
 3. Các khái niệm
 4. Đặc tính chung của phần mềm
 5. Thế nào là phần mềm tốt ?
 6. Các ứng dụng phần mềm
- II. Những vấn đề trong phát triển phần mềm
- III. Quy trình phát triển phần mềm

2

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

3

Các đặc tính của SW và HW

Hardware

- Vật "cứng"
- Kim loại
- Vật chất
- Hữu hình
- Sản xuất công nghiệp bởi máy móc là chính
- Định lượng là chính
- Hỏng hóc, hao mòn

Software

- Vật "mềm"
- Kỹ thuật sử dụng
- Trừu tượng
- Vô hình
- Sản xuất bởi con người là chính
- Định tính là chính
- Không hao mòn

4

Định nghĩa 1

- IEEE: Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system.
- 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

5

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

6

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

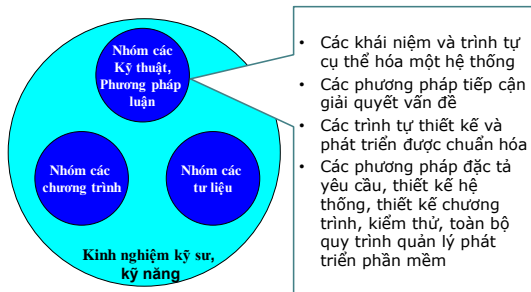
7

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): Kỹ năng của kỹ sư 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

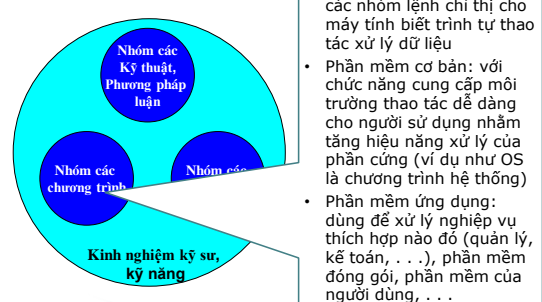
8

Phần mềm là gì ?

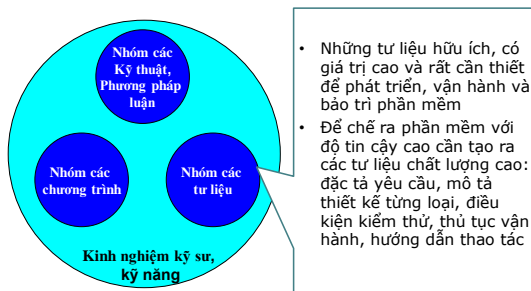


9

Phần mềm là gì ?

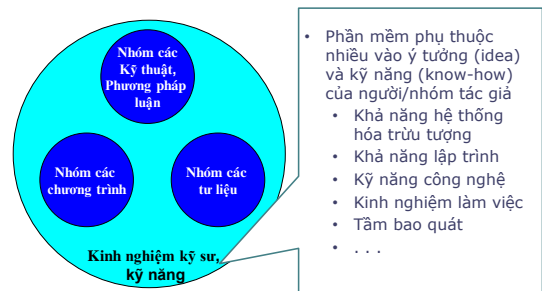


Phần mềm là gì ?



11

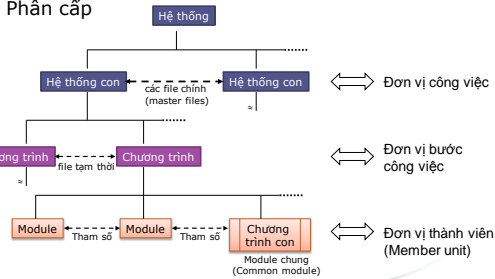
Phần mềm là gì ?



12

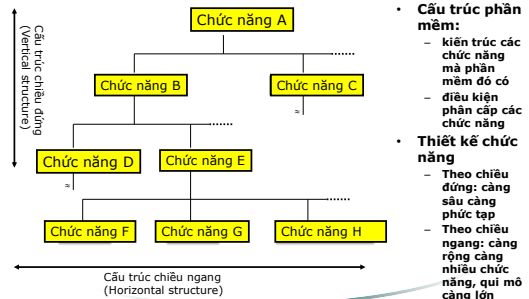
2. Kiến trúc phần mềm

• Phân cấp



Phần mềm

Nhìn từ phương diện cấu trúc



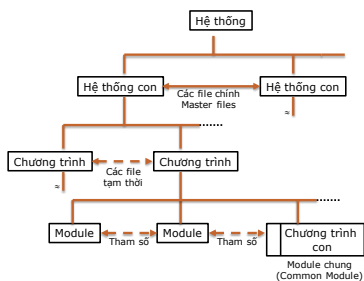
• Cấu trúc phần mềm:

- kiến trúc các chức năng mà phần mềm đó có
- điều kiện phân cấp các chức năng

• Thiết kế chức năng

- Theo chiều đứng: càng sâu càng phức tạp
- Theo chiều ngang: càng rộng càng nhiều chức năng, qui mô càng lớn

Phần mềm Nhìn từ phương diện thủ tục



- Quan hệ thứ tự giữa các thành phần cấu thành phần mềm
- Thuật toán với những phép lập, rẽ nhánh, điều khiển luồng xử lý (quay lui hay bỏ qua)
- Cấu trúc logic biểu thị từng chức năng có trong phần mềm và trình tự thực hiện chúng
- Thiết kế cấu trúc trước rồi sang chức năng

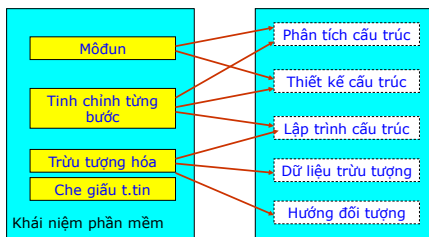
Từ phương pháp luận phần mềm sang kỹ thuật phần mềm

• Khi chế tác phần mềm cần nhiều phương pháp:

- 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

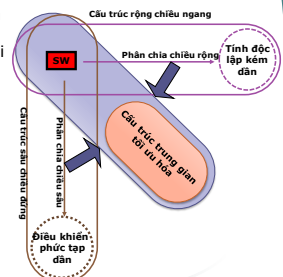
1. Định nghĩa chung về phần mềm
2. Kiến trúc phần mềm
3. Các khái niệm

Từ phương pháp luận phần mềm sang kỹ thuật phần mềm



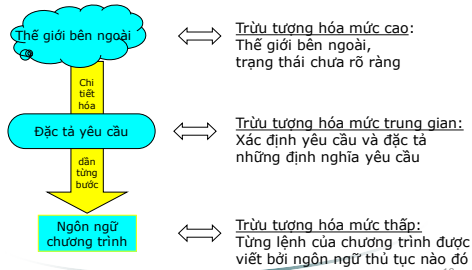
3.1 Tính module (Modularity)

- Là khả năng phân chia phần mềm thành các module ứng với các chức năng, đồng thời cho phép quản lý tổng thể: khái niệm phân chia và trộn (partition and merge)
- Hai phương pháp phân chia module theo chiều
 - Theo chiều sâu
 - Theo chiều rộng
- Quan hệ giữa các module ? qua các đối số (arguments)



3.2 Tinh chỉnh từng bước (Step refinement)

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



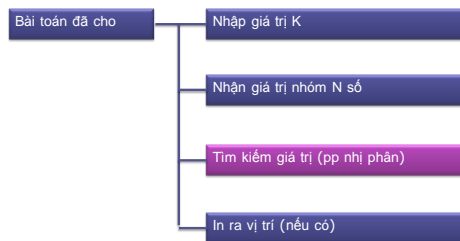
19

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)

20

Cụ thể hóa thủ tục qua các chức năng



21

Cụ thể hóa bước tiếp theo



22

Mức mô tả chương trình (bằng PDL)

```

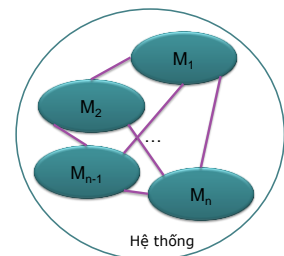
Bắt Đầu
  Đọc K
  Nhập giá trị cho mảng 1 chiều A(I), (I = 1, 2, ..., N)
  MIN = 1
  MAX = N
  DO WHILE (Có giá trị bằng K không, cho đến khi MIN > MAX)
    Lấy MID = (MIN + MAX) / 2
    IF A(MID) > K THEN
      MAX = MID - 1
    ELSE
      IF A(MID) < K THEN
        MIN = MID + 1
      ELSE
       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

23

Câu hỏi

Làm thế nào để định nghĩa cấu trúc của một hệ thống được thiết kế dựa trên các module?

Đâu là các đặc tính cần có của cấu trúc này?



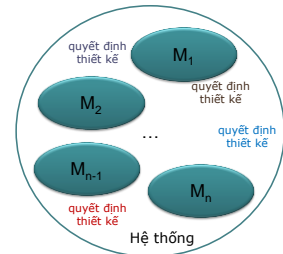
3.3. Che giấu thông tin (Information hiding) [Parnas72]

- Các module nên được đặc trưng bởi những quyết định thiết kế (design decision) sao cho mỗi module đều là bí mật đối với các module khác
- Rất hữu ích cho kiểm thử và bảo trì phần mềm

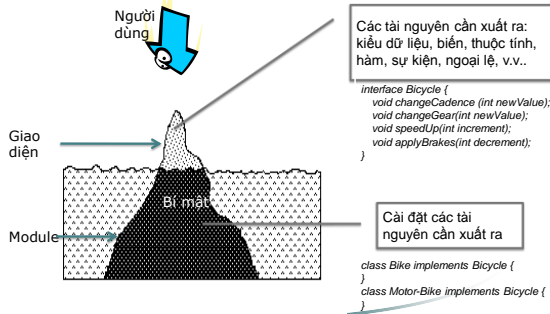
25

3.3. Che giấu thông tin (Information hiding) [Parnas72]

Cố định tất cả các quyết định thiết kế (design decision) có khả năng bị thay đổi
Gán mỗi quyết định thiết kế vào một module mới; lúc này quyết định thiết kế sẽ là phần bí mật của module (module secret)
Thiết kế giao diện của module (**module interface**), giao diện này sẽ không thay đổi khi phần bí mật của module thay đổi



3.3. Che giấu thông tin (Information hiding) [Parnas72]



3.4. Trừu tượng hóa (Abstraction)

- Cho phép tập trung xem xét 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 . . .

28

1. Định nghĩa chung về phần mềm
2. Kiến trúc phần mềm
3. Các khái niệm
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

29

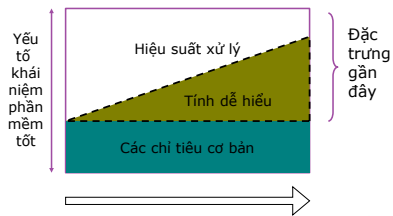
4. Đặc tính chung của phần mềm

- Chức năng của phần mềm thường biến hóa, thay đổi theo thời gian (theo nơi sử dụng)
- Hiệu ứng lan 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

30

5. Thế nào là phần mềm tốt ?

1. Định nghĩa chung về phần mềm
2. Kiến trúc phần mềm
3. Các khái niệm
4. Đặc tính chung của phần mềm



31

5.1. Các chỉ tiêu cơ bản

1. Định nghĩa chung về phần mềm
2. Kiến trúc phần mềm
3. Các khái niệm
4. Đặc tính chung của phần mềm
5. Thế nào là phần mềm tốt ?

- Phản ánh đúng yêu cầu người dùng (tính hiệu quả - effectiveness)
- Chưa 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

32

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, . . .

33

5.3. 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

34

Tính dễ hiểu: chỉ tiêu ngày càng quan trọng

Ví dụ cụ thể ???

1. Định nghĩa chung về phần mềm
2. Kiến trúc phần mềm
3. Các khái niệm
4. Đặc tính chung của phần mềm
5. Thế nào là phần mềm tốt ?
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 (Engineering & Science 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 (Artificial Intelligent SW)

35

Phân loại phần mềm

1. Định nghĩa chung về phần mềm
2. Kiến trúc phần mềm
3. Các khái niệm
4. Đặc tính chung của phần mềm
5. Thế nào là phần mềm tốt ?
6. Các ứng dụng phần mềm



36

Bài tập về nhà:

Phân biệt các khái niệm sau

- Hệ thống, phần mềm, ứng dụng
- Lập trình, phát triển phần mềm
- Lập trình viên và kỹ sư phần mềm

37

PHẦN I: GIỚI THIỆU CHUNG VỀ CÔNG NGHỆ PHẦN MỀM

I. Bản chất phần mềm

II. Những vấn đề trong phát triển phần mềm

1. Khủng hoảng phần mềm là gì ?

2. Những khó khăn trong sản xuất phần mềm

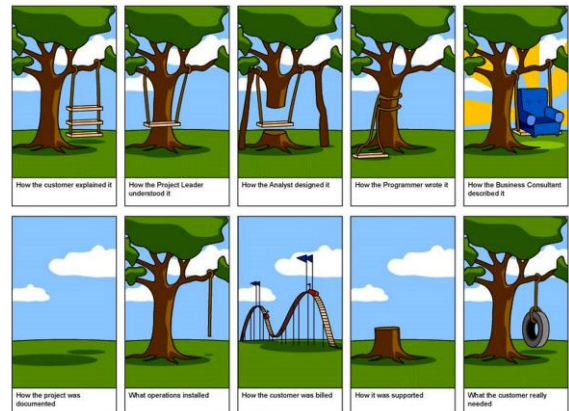
III. Quy trình phát triển phần mềm (CNPМ)

38

Một số yếu tố

- Phần mềm càng lớn sẽ kéo theo phức tạp hóa và tăng chi phí phát triển
- Đối 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

39



2. Những khó khăn trong sản xuất phần mềm

- Không có phương pháp mô tả rõ ràng định nghĩa yêu cầu của người dùng (khách hàng)
→ **Sau khi bàn giao sản phẩm để phát sinh những trục trặc (troubles)**
- Với những phần mềm quy mô lớn, tư liệu đặc tả đã cố định thời gian dài
→ **Khó đáp ứng nhu cầu thay đổi của người dùng một cách kịp thời trong thời gian đó**
- Phương pháp luận thiết kế không nhất quán
→ **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)**
- Không có chuẩn về việc tạo tư liệu quy trình sản xuất phần mềm
→ **Đặc tả không rõ ràng sẽ làm giảm chất lượng phần mềm**

41

2. Những khó khăn trong sản xuất phần mềm

- 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
→ **thường bàn giao sản phẩm không đúng hạn**
- Coi trọng việc lập trình hơn khâu thiết kế
→ **giảm chất lượng phần mềm**
- Coi thường việc tái sử dụng phần mềm (software reuse)
→ **giảm năng suất lao động**
- Phần lớn các thao tác trong quy trình phát triển phần mềm do con người thực hiện
→ **giảm năng suất lao động**
- Không chứng minh được tính đúng đắn của phần mềm
→ **giảm độ tin cậy của phần mềm**

42

Những vấn đề trong sản xuất phần mềm (tiếp)

- Chuẩn về phần mềm tốt không thể đo được một cách định lượng
→ **Không thể đánh giá được một hệ thống đúng đắn hay không**
- Đầu tư nhân lực lớn vào bảo trì
→ **giảm hiệu suất lao động của nhân viên**
- Công việc bảo trì kéo dài
→ **giảm chất lượng của tư liệu và ảnh hưởng xấu đến những việc khác**
- Quản lý dự án lỏng lẻo
→ **quản lý lịch trình sản xuất phần mềm không rõ ràng**
- Không có tiêu chuẩn để ước lượng nhân lực và dự toán
→ **làm kéo dài thời hạn và vượt kinh phí của dự án**

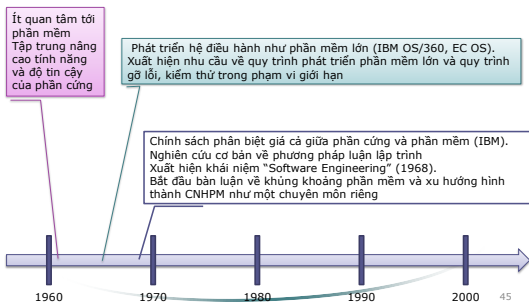
43

PHẦN I: GIỚI THIỆU CHUNG VỀ CÔNG NGHỆ PHẦN MỀM

- I. Bản chất phần mềm
- II. Những vấn đề trong phát triển phần mềm
- III. Quy trình phát triển phần mềm
 1. Sự tiến triển của các phương pháp thiết kế phần mềm
 2. Định nghĩa Công nghệ học phần mềm
 3. Vòng đời của phần mềm
 4. Một số quy trình phát triển phần mềm

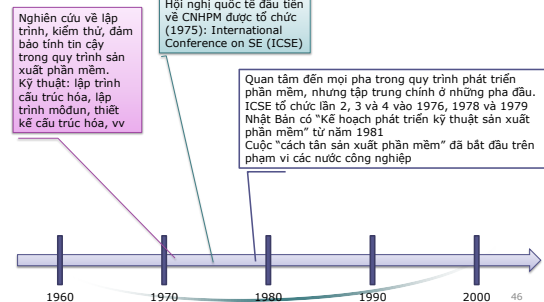
44

1. Sự tiến triển của các phương pháp thiết kế phần mềm



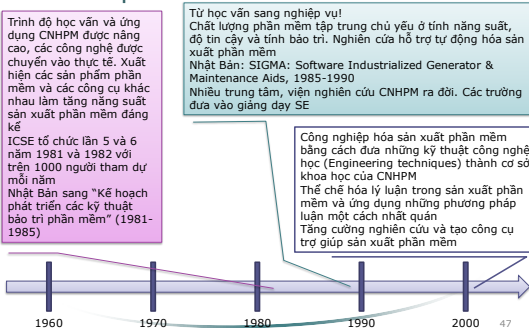
45

1. Sự tiến triển của các phương pháp thiết kế phần mềm



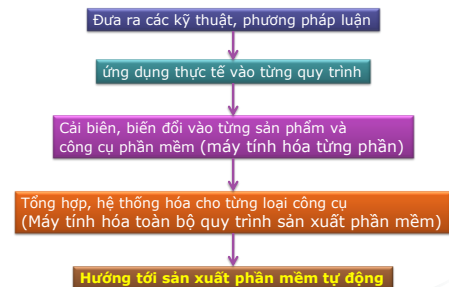
46

1. Sự tiến triển của các phương pháp thiết kế phần mềm



47

Hình thái sản xuất Phần mềm



48

2. Công nghệ phần mềm (Software Engineering)

- Bauer [1969]: CNPM 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]: CNPM là việc xây dựng phần mềm nhiều phiên bản bởi nhiều người
- Ghezzi [1991]: CNPM 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

49

2. Công nghệ phần mềm (Software Engineering)

- IEEE [1993]: CNPM 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]: CNPM là bộ môn tích hợp cả quy trình, các phương pháp, các công cụ để phát triển phần mềm máy tính

50

2. Công nghệ phần mềm (Software Engineering)

- Sommerville [1995]: CNPM là lĩnh vực liên quan đến lý thuyết, phương pháp và công cụ dùng cho phát triển phần mềm
- K. Kawamura [1995]: CNPM 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

51

2. Công nghệ phần mềm (Software Engineering)

- Công nghệ 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 scientific field to deal with methodologies, techniques and tools integrated in software production-maintenance process to obtain software with desired qualities]

52

Công nghệ trong CNPM ?

- Như các ngành công nghệ học khác, CNPM cũng lấy các phương pháp khoa học làm cơ sở
- Các kỹ thuật về thiết kế, chế tạo, kiểm thử và bảo trì phần mềm đã được hệ thống hóa thành phương pháp luận và hình thành nên CNPM
- 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 CNPM

53

Công nghệ trong CNPM ? (tiếp)

- 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)
- Trong khái niệm phần mềm, không chỉ có chương trình mà cả tư liệu về phần mềm
- Cách tiếp cận công nghệ (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

54

Vòng đời phần mềm

1. Sự tiến triển của các phương pháp thiết kế phần mềm
2. Định nghĩa Công nghệ học phần mềm
3. **Vòng đời của 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 đầu 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

55

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

56

3.4 Quy trình phát triển phần mềm

Khung quy trình chung (Common process framework)

Hoạt động khung (Framework activities)

Tập tác vụ (Task sets)

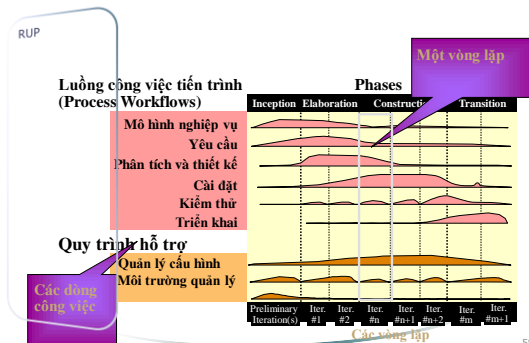
Tác vụ (Tasks)

Điểm quan trọng (milestones), sản phẩm chuyển

Điểm Kiểm Tra Chất Lượng (SQA points)

Các hoạt động giám sát, đánh giá kỹ thuật, đảm bảo chất lượng phần mềm, quản lý cấu hình, quản lý rủi ro, ... (Umbrella activities)

57



58

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

1. Sự tiến triển của các phương pháp thiết kế phần mềm
2. Định nghĩa Công nghệ học phần mềm
3. Vòng đời của phần mềm
4. **Một số quy trình phát triển phần mềm**

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 phân 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, thỏa mãn các yêu cầu khách hàng. Có cơ sở chuẩn xác đánh giá chất lượng, thời gian, chi phí và phân tích dự án và các tiến trình.

59

Các khái niệm

4. Một số quy trình phát triển phần mềm
- 4.1. CMM
- a. Tại sao phải sử dụng mô hình CMM trong công nghệ làm phần mềm
- b. **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.

60

Thực thi tiến trình phần mềm (Software Process Performance)

4. Một số quy trình phát triển phần mềm
4.1. CMM
a. Tại sao phải sử dụng mô hình CMM trong công nghệ làm phần mềm
b. Các khái niệm cơ bản trong CMM

- 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.

61

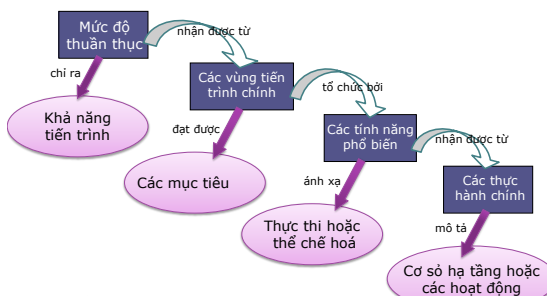
Độ thuần thục của tiến trình phần mềm (Software process maturity)

4. Một số quy trình phát triển phần mềm
4.1. CMM
a. Tại sao phải sử dụng mô hình CMM trong công nghệ làm phần mềm
b. Các khái niệm cơ bản trong CMM

- 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.

62

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



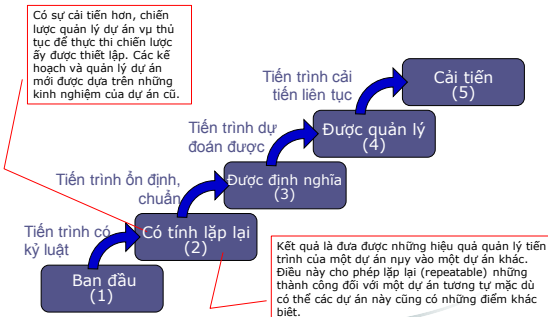
63

d. Mô hình 5 mức của CMM



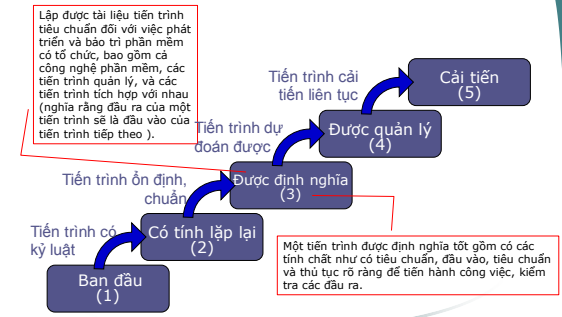
64

d. Mô hình 5 mức của CMM



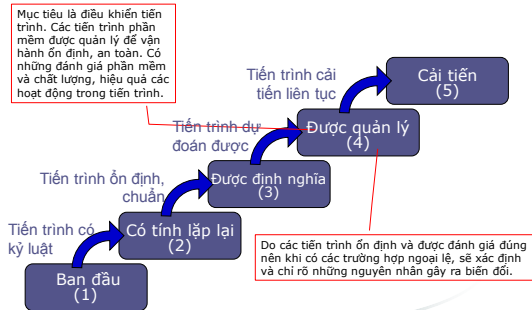
65

d. Mô hình 5 mức của CMM



66

d. Mô hình 5 mức của CMM



67

d. Mô hình 5 mức của CMM



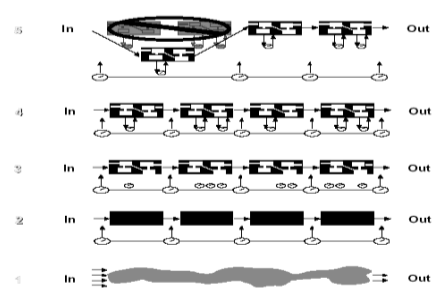
68

18 Vùng tiến trình chính KPA (Key Process Area)



69

Khả năng nhìn nhận tại mỗi mức thuần thực



70

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

- Khi mức độ thuần thực tăng, sự sai khác giữa kết quả đạt được và kết quả dự tính giảm xuống.
- Khi mức độ thuần thực tăng, độ biến động của kết quả thực tế so với kết quả đề ra giảm xuống.
- Khi mức độ thuần thực 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.

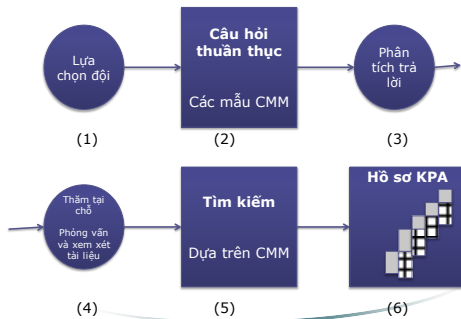
71

e. 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.

72

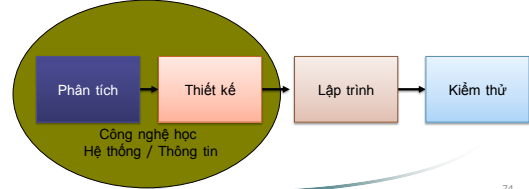
Những điểm chung của 2 phương thức sử dụng CMM



73

4.2. 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

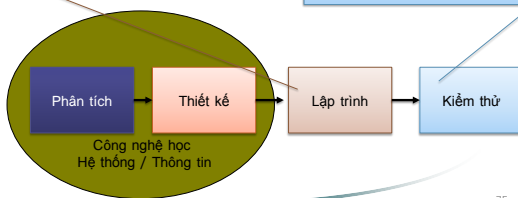


74

4.2. 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

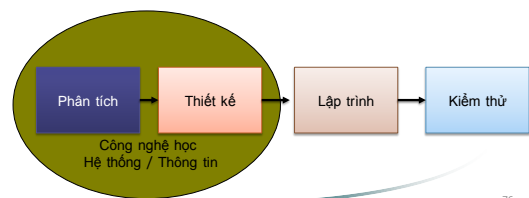
Kiểm thử (Testing): Kiểm tra các chương trình và mô đun cả về logic bên trong và chức năng bên ngoài, nhằm phát hiện ra lỗi và đảm bảo với đầu vào xác định thì cho kết quả mong muốn



75

4.2. 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



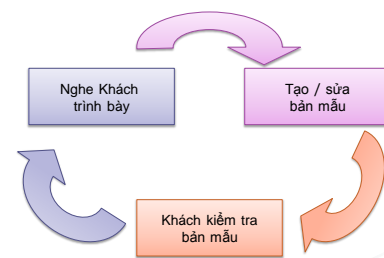
76

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!

77

4.3. Mô hình chế thử (Prototyping model)



78

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 ý yêu cầu của người dùng

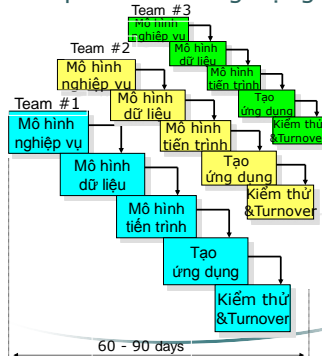
79

4.4. 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)

80

Mô hình phát triển ứng dụng nhanh



82

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

83

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)

84

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 để làm dự án đổ vỡ
- RAD không phải tốt cho mọi ứng dụng, nhất là với ứng dụng không thể mô đun hóa hoặc đòi hỏi tính năng cao
- Mạo hiểm kỹ thuật cao thì không nên dùng RAD

85

Mở đầu

4. Một số quy trình phát triển phần mềm
4.1. CMM
4.2. Mô hình tuyến tính
4.3. Mô hình chế thử
4.4. Mô hình RAD
4.5. 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)

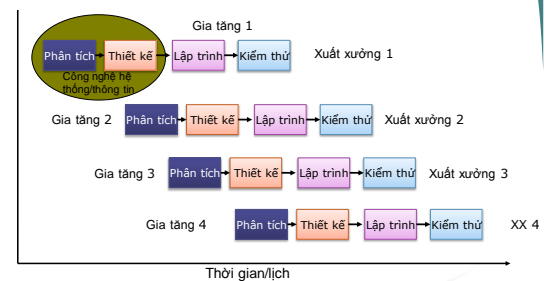
86

Mô hình gia tăng (The incremental model)

- Kết hợp mô hình tuần tự và ý tưởng lặp lại của chế bản mẫu
- Sản phẩm lỗi 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

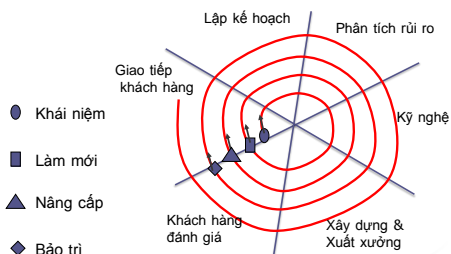
87

Mô hình gia tăng



88

Mô hình xoắn ốc (spiral)



89

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

90

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

91

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

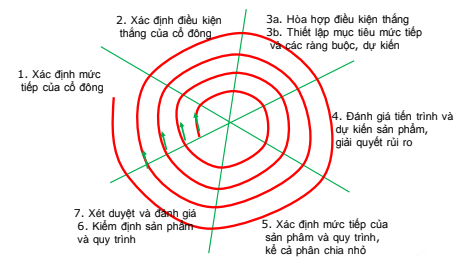
92

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

93

Mô hình xoắn ốc WINWIN



94

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

95

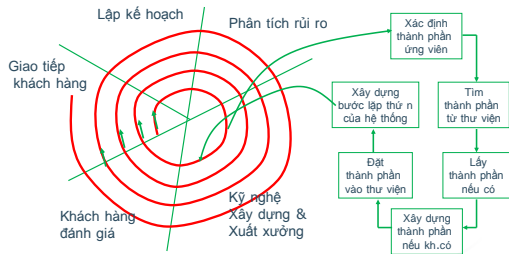
Mô hình hướng thành phần Component-based model

- 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

96

4. Một số quy trình phát triển phần mềm
 4.1. CMM
 4.2. Mô hình tuyến tính
 4.3. Mô hình chế thử
 4.4. Mô hình RAD
 4.5. Các mô hình tiến hóa
4.6. Mô hình dựa thành phần

Mô hình dựa thành phần



97

4.7. Mô hình RUP (Rational Unified Process)

- SV tự nghiên cứu

4. Một số quy trình phát triển phần mềm
 4.1. CMM
 4.2. Mô hình tuyến tính
 4.3. Mô hình chế thử
 4.4. Mô hình RAD
 4.5. Các mô hình tiến hóa
 4.6. Mô hình dựa thành phần

98

Tổng kết các mô hình

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

4. Một số quy trình phát triển phần mềm
 4.1. CMM
 4.2. Mô hình tuyến tính
 4.3. Mô hình chế thử
 4.4. Mô hình RAD
 4.5. Các mô hình tiến hóa
 4.6. Mô hình dựa thành phần

99

4.8. 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

4. Một số quy trình phát triển phần mềm
 4.1. CMM
 4.2. Mô hình tuyến tính
 4.3. Mô hình chế thử
 4.4. Mô hình RAD
 4.5. Các mô hình tiến hóa
 4.6. Mô hình dựa thành phần
 4.7. Mô hình RUP

100

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

101

5. 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

102

Những việc cần làm

Công việc	Người tham gia	Người phụ trách	Kết quả cần đạt	Hạn chót
Bầu lớp trưởng	Tất cả SV	Tất cả SV	Có lớp trưởng	28/8/2012
Tham gia vào hộp thư nhóm ythoigiang-it4080-k55@googlegroups.com	Tất cả SV	Lớp trưởng	Tất cả các SV đăng ký học môn này đều phải gia nhập hộp thư nhóm Lưu ý: khi gia nhập, đổi tên hiện thị theo dạng: Họ và tên MSSV lớp khoá	4/9/2012
Chốt danh sách sinh viên và phân nhóm BTL (3-4 người)	Tất cả SV	Lớp trưởng	Danh sách phân nhóm gồm: mã nhóm (FSE01 – FSExx), họ tên SV, số hiệu SV, lớp, email, nhóm trưởng (t/f).	4/9/2012
Đăng ký bài tập lớn	Nhóm trưởng	Lớp trưởng	Danh sách đăng ký bài tập lớn = danh sách phân nhóm + mã BTL (BTL01 – BTLyy) + mô tả ngắn gọn BTL Mã nhóm Họ và tên MSSV lớp khoá	25/9/2012
Làm bài tập / bài tập lớn	Tất cả SV	Tất cả SV / Nhóm trưởng	Báo cáo, mã nguồn / tài liệu CT	Theo yêu cầu của giáo viên
Chữa bài tập / bài tập lớn	Tất cả SV	Tất cả SV / Nhóm trưởng	Báo cáo, mã nguồn / tài liệu CT	Theo yêu cầu của giáo viên
Báo cáo bài tập lớn	Tất cả SV	Tất cả SV	Báo cáo, mã nguồn / tài liệu CT	Theo yêu cầu của giáo viên

Bài tập về nhà

- Bài tập về nhà: Một số ví dụ về các phần mềm có thể áp dụng các quy trình nói trên

104