CHƯƠNG 3. ĐỘ ĐO PHẦN MỀM (SOFTWARE METRICS)

3.1.1. Tại sao phải đo ???

- Để có cơ sở **phân tích, đánh giá** một cách khách quan về một vấn đề hay một đối tượng nào đó
- Nghi ngờ → đặt giả thuyết → muốn tìm hiểu → đo → kết quả → phân tích → kết luận → dự đoán,…
- Mỗi số đo: **không** phản ánh hết mọi khía cạnh của đối tượng (độ phức tạp của phần mềm, của thuật toán,…)

- Cần phối hợp nhiều độ đo
- Vận dụng thêm các tiếp cận **định tính**

3.1.2. Khái niệm

- Các độ đo phần mềm: tính toán, ước lượng được các đại lượng liên quan đến các đối tượng, các hoạt động thuộc về tiến trình sản xuất phần mềm
  - Ước lượng: giá gia công, phỏng đoán kích thước,…
  - Đánh giá: chất lượng phần mềm,…
  - Đánh giá: chất lượng qui trình sản xuất

Cải tiến chất lượng phần mềm, chất lượng tiến trình sản xuất phần mềm

3.1.3. Các phép đo cơ bản

- **Đo dựa vào tỉ số**: chia 1 đại lượng cho 1 đại lượng khác, tử số và mẫu số của tỉ số là số phần tử của hai tập hợp rời nhau
- **Đo dựa vào tỉ lệ**: tỉ lệ khác với tỉ số ở chỗ tử số tham gia vào một phần của mẫu số $\frac{a}{a+b}$.

  Ví dụ: Tỉ lệ người dùng PC= $\frac{Số\ người\ dùng\ được}{số\ người\ dùng\ được + Số\ người\ ko\ dùng\ được}$.

  - Tỉ số thường dùng cho 2 nhóm người, trong khi tỉ lệ có thể dùng cho nhiều phạm trù trong một nhóm. Có thể nhiều hơn 2 phạm trù:

$$\frac{a}{a+b+c+d+e}$$

- **Đo dựa vào tỉ lệ phần trăm (%):** Tỉ lệ % có được bằng cách nhân tỉ lệ với 100

3.1.3. Các phép đo cơ bản (tt)

Ví dụ 1

❖ tỉ số xây dựng PM $=\dfrac{\text{Số nhân viên kiểm tra phần mềm}}{\text{Số nhân viên xây dựng phần mềm}}$

- ❖ Thường có phạm vi từ **1:10** đến **1:1** phụ thuộc vào quy mô tổ chức tiến trình phát triển phần mềm
- ❖ ***Với các tỉ số nhỏ***: đội ngũ xây dựng phần mềm làm cả việc kiểm tra các chức năng chi tiết, trong khi đội ngũ kiểm tra phần mềm thực hiện kiểm tra ở mức độ hệ thống
- ❖ ***Với các tỉ số lớn***: đội ngũ kiểm tra phần mềm có trách nhiệm chính trong pha kiểm tra phần mềm và đảm bảo chất lượng

❖ Đề án phi thuyền con thoi: 70 nhân viên kiểm tra, 49 nhắn viên phát triển phần mềm, kết quả đo:

$\dfrac{70}{49} \approx 7:5$

→ lớn hơn nhiều so với các đề án thông thường

3.1.3. Các phép đo cơ bản (tt)

Ví dụ 2: Đo tỉ lệ thành công của PM

❖ Tỉ lệ $=$ $\dfrac{\text{Số khách hàng vừa ý với phần mềm}}{\text{Tổng số khách hàng sử dụng phần mềm}}$

❖ Dùng để khảo sát cho 1 phần mềm bất kỳ

❖ Độ đo này phụ thuộc vào quan niệm khách hàng

- ❖ ***Giá trị đo lớn:*** Phần mềm có "chất lượng" tốt
- ❖ ***Giá trị đo nhỏ:*** Phần mềm có "chất lượng" không tốt

→ có thể không phản ảnh được "chất lượng bản chất" của phần mềm đang khảo sát

3.1.3. Các phép đo cơ bản (tt)

Ví dụ 3: tỉ lệ %

Sự phân bố các loại lỗi trong mỗi đề án

Dạng lỗi	Đề án A	Đề án B	Đề án C
Khảo sát yêu cầu khách hàng	15,0%	41,0%	20,3%
Thiết kế	25,0%	21,8%	22,7%
Mã hóa chương trình	50,0%	28,6%	36,7%
Các lỗi khác	10,0%	8,6%	20,3%
Tổng %	100%	100%	100%
Tổng số lỗi	200 lỗi	105 lỗi	128 lỗi

So sánh mỗi loại lỗi giữa 3 đề án với nhau

Dạng lỗi	Đề án A	Đề án B	Đề án C	Tổng %	Tổng lỗi
Khảo sát yêu cầu	30,3%	43,4%	26,3%	100%	99
Thiết kế	49%	22,5%	28,5%	100%	102
Mã hóa CT	56,5%	16,9%	26,6%	100%	177
Các lỗi khác	36,4%	16,4%	47,2%	100%	55

3.1.4. Vài lưu ý về tỉ lệ %

- Nên **ghi nhận tổng số** các trường hợp đang xét (giá trị gốc của mẫu số trước khi qui về tỉ lệ %)
- Số trường hợp **không nên quá nhỏ**, nếu ngược lại thì chỉ nên dùng các số liệu gốc.
- Ví dụ: Trong 1000 dòng lệnh thì có 1 lỗi xuất hiện , tỉ lệ = 1/1000 = 0.001 (quá nhỏ).

3.2.1. Phân loại

- Đo các đặc trưng sản phẩm
  - Kích thước
  - Độ phức tạp
  - Số chức năng
  - Hiệu suất hoạt động
  - Xếp loại chất lượng
  - …

3.2.1. Phân loại (tt)

- Đo qui trình phát triển phần mềm
  - Tính hiệu quả của việc khử lỗi trong các pha
  - Khả năng kiểm tra phát hiện lỗi
  - Thời gian hiệu chỉnh lỗi trung bình
  - …
- Đo các đặt trưng đề án phần mềm
  - Số lượng người tham gia đề án
  - Việc bố trí người trong các hoạt động khác nhau
  - Thời gian biểu
  - Năng suất lao động trong đề án
  - …

3.3.1. Các thuộc tính của sản phẩm phần mềm

Làm sao có thể đo được ?

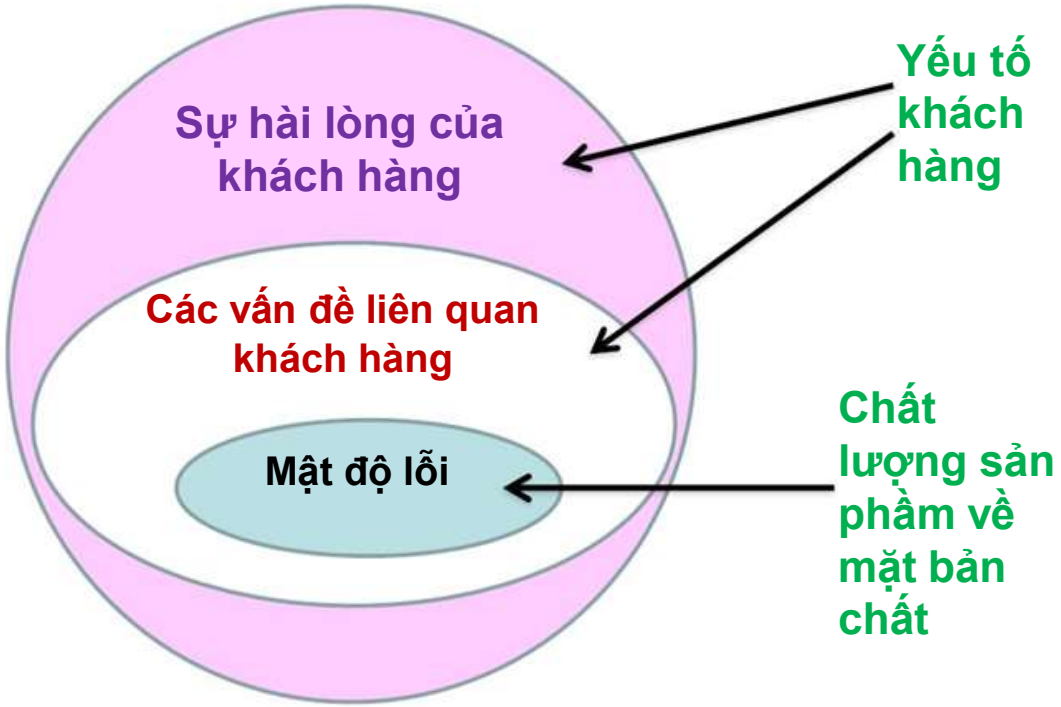
- Chất lượng sản phẩm (rất nhiều góc độ khác nhau):
  - Ít lỗi, dễ bảo trì, tương thích,…
  - Tổ chức đơn thể tốt, có thể tái sử dụng,…
  - Dễ mở rộng, có thể tiến hóa,…
- Độ phức tạp của sản phẩm
- Kích thước (độ lớn) sản phẩm
- ….

3.3.2. Các chỉ số dùng trong đánh giá chất lượng phần mềm

- Thời gian trung bình xảy ra sự cố (MTTF - Mean Time To Failure): đòi hỏi chính xác cao, thường được dùng cho các hệ thống tuyệt đối an toàn
  - Hệ thống điều khiển máy bay lên xuống
  - Hệ thống sử dụng trong điều khiển chế tạo vũ khí
  - …
- Độ đo chất lượng sản phẩm cuối bao gồm:
  - Mật độ lỗi (defect density)
  - Các vấn đề liên quan đến khách hàng
    - Hiểu sai, lỗi trùng, sử dụng không đúng, tự gây ra
  - Sự hài lòng của khách hàng

Thường dùng trong sản xuất phần mềm thông dụng

3.3.3. Sự liên hệ giữa 3 loại chỉ số chất lượng



3.3.4. Mật độ lỗi

$$\text{Mật độ lỗi} = \frac{Tổng\ số\ lỗi}{\text{Kích thước sản phẩm}}$$

- Kích thước?
  - ***Là đại lượng*** *đặc trưng đo độ lớn sản phẩm. Ví dụ: số dòng lệnh (LOC/SLOC), ngàn dòng lệnh (KLOC,KSLOC), số điểm chức năng (FP),…*
- *tổng số lỗi bao gồm:*
  - ***D1****: lỗi đã biết (nhờ thanh tra, kiểm thử, tình cờ,…)*
  - ***D2****: lỗi còn "tiềm ẩn" trong sản phẩm*
- Ví dụ:
  - Kích thước : 50 KLOC
  - Tổng số lỗi : 100 lỗi
  - Mật độ lỗi = 100/50 = 2 lỗi / KLOC

3.3.5. Các vấn đề khách hàng

- Xét đến tất cả các vấn đề xảy ra trong quá trình sử dụng sản phẩm → **Lỗi giả**…
- Các dạng lỗi được xét đến:
  - Các lỗi thực sự (do khuyết điểm của phần mềm)
  - **Các lỗi giả**:
    - Các vấn đề sử dụng phần mềm không đúng
    - Thông tin hay tài liệu không rõ ràng, bị hiểu sai
    - Các lỗi thực sự nhưng bị trùng lập (được phản ảnh nhiều lần)
    - Các lỗi do chính khách hàng gây ra

Độ đo PUM

3.3.6. Độ đo PUM

- PUM - Problems per User Month
- Công thức: $\text{PUM} = \dfrac{\text{Tổng số sự cố do khách hàng báo cáo}}{\text{Số bản cài đặt x số tháng khai thác}}$
- Ví dụ: triển khai 1 phần mềm cho 15 khách hàng trong 6 tháng, có 65 sự cố báo lại: $\dfrac{\mathbf{65}}{(\mathbf{15}\ \boldsymbol{x}\ \mathbf{6})} = 0.72$

- *Liên hệ giữa PUM và chất lượng*

PUM giảm ⟺ ***Chất lượng tăng***

3.3.6. Độ đo PUM (tt)

- Để PUM thấp, có thể chọn những phương án sau:
  - Cải tiến qui trình phát triển phần mềm để giảm các lỗi thực sự
  - Giảm các lỗi giả (giải quyết tốt các vấn đề sử dụng, cải tiến tài liệu, huấn luyện khách hàng, tăng cường các dịch vụ khách hàng, …)

  Giảm tử số PUM (tích cực)

  - Gia tăng số bản bán được (đẩy mạnh việc bán hàng bằng các biện pháp khuyến mãi, quảng cáo,…)

  Giảm mẫu số PUM (tiêu cực)

3.3.7. Khuyết điểm của PUM

Xem nhẹ việc giảm thiểu thực sự các vấn đề (Giảm tử số PUM)

3.3.8. Mối liên hệ giữa mật độ lỗi và PUM

	Mật độ lỗi	PUM
Tử số	Các lỗi thực sự (không trùng lặp)	Tất cả mọi vấn đề liên quan khách hàng
Mẫu số	Kích thước (độ lớn) sản phẩm (KLOC)	Mức độ dùng của khách hàng (user/month)
Góc độ đo	Nhà sản xuất phần mềm	Khách hàng
Phạm vi	Chất lượng bản chất của sản phẩm	Chất lượng bản chất của sản phẩm kết hợp với các yếu tố khác

3.3.9. Đo sự thỏa mãn của khách hàng

- Sự thỏa mãn của khách hàng được đo dựa trên một số phạm trù liên quan đến sản phẩm
  - Chức năng, tốc độ, tài liệu hướng dẫn,…
  - Việc bảo trì, dịch vụ khách hàng,…
- Số liệu đo được lấy dựa trên 5 ngưỡng:
  - Rất thỏa mãn (A)
  - Thỏa mãn (B)
  - Vừa phải (C)
  - Không thỏa mãn (D)
  - Rất không thỏa mãn (E)
- Số đo khác cho các mục đích khác nhau
  - tỷ lệ % các khách hàng hoàn toàn hài lòng (A)
  - tỷ lệ % các khách hàng hài lòng (A)+(B)
  - tỷ lệ % các khách hàng không hài lòng (D)+(E)

3.4.1. Sự cần thiết

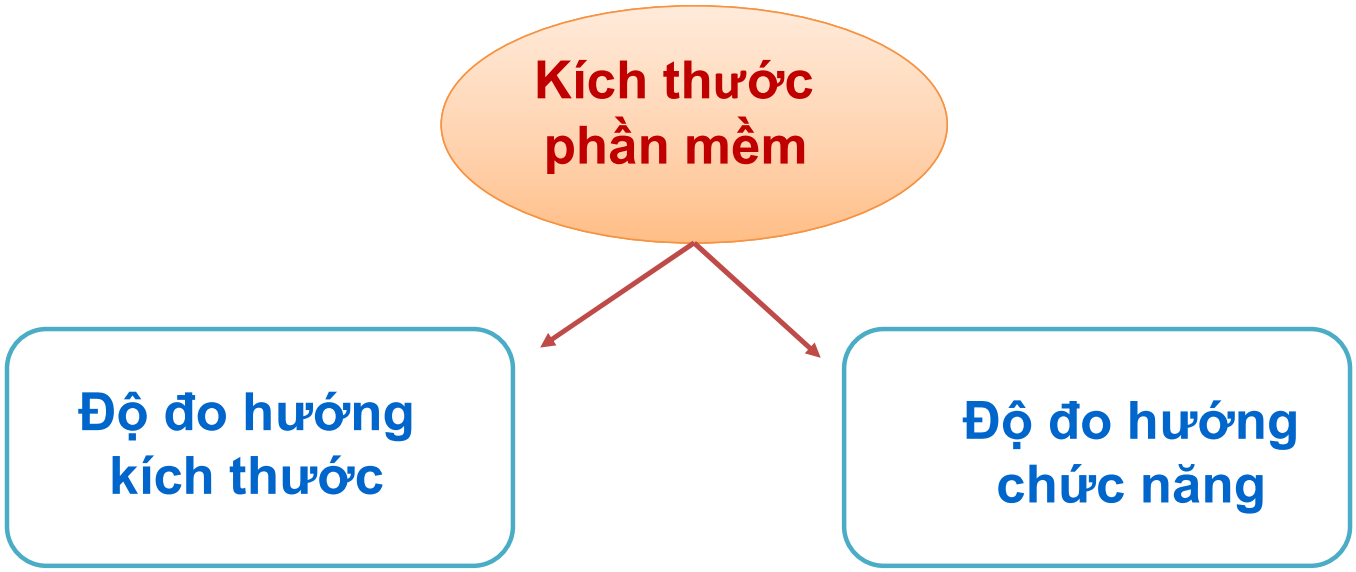
- Cần có đơn vị tính kích thước hay độ lớn của phần mềm để có thể:
  - So sánh giữa các phần mềm với nhau
  - Làm cơ sở để tính năng suất lao động trung bình:

$$\frac{\text{Số đơn vị phần mềm}}{\text{Giờ làm việc}}$$

  - Làm cơ sở để ước lượng một phần mềm:

    "bằng cỡ bao nhiêu đơn vị phần mềm"
  - Làm cơ sở để qui ra tiền. Ví dụ: hiện trung bình cần 100 USD để sản xuất ra 1 đơn vị phần mềm
  - Làm cơ sở để báo cáo. Ví dụ: năm 2013, công ty phần mềm PSV đã xuất khẩu tổng cộng 453 đơn vị phần mềm

3.4.2. Có hai hướng tiếp cận chính



3.4.3. Độ đo hướng kích thước

- Dựa trên số dòng mã nguồn.
- Đơn vị tính là: LOC/pm hay KLOC/pm,

$$\text{Kích thước} = \frac{\text{Tổng số dòng lệnh}}{\text{Số người tham gia x Thời gian}}$$

- LOC (Line Of Code)
- KLOC (Thousand Lines Of Code)
- Thời gian thực hiện: quy về tháng (month)
- Số người tham gia: person
- Đơn vị pm: Person x Month

Độ đo hướng kích thước – ví dụ

- Sau khi ước lượng, nghiên cứu, một hệ thống có thể được **một người** cài đặt bằng 5000 dòng Hợp ngữ hoặc bằng 1500 dòng ngôn ngữ lập trình C với khảo sát chi tiết như sau:

	Phân tích	Thiết kế	Cài đặt	Kiểm chứng	Viết tài liệu	năng suất
Hợp ngữ	3 tuần	5 tuần	8 tuần	10 tuần	2 tuần	28 tuần ≈ 7 tháng ⇔ 5000LOC/7m ⇔ 714LOC/m
Ngôn ngữ C	3 tuần	5 tuần	4 tuần	6 tuần	2 tuần	20 tuần ≈ 5 tháng ⇔1500LOC/5m ⇔ 300LOC/m

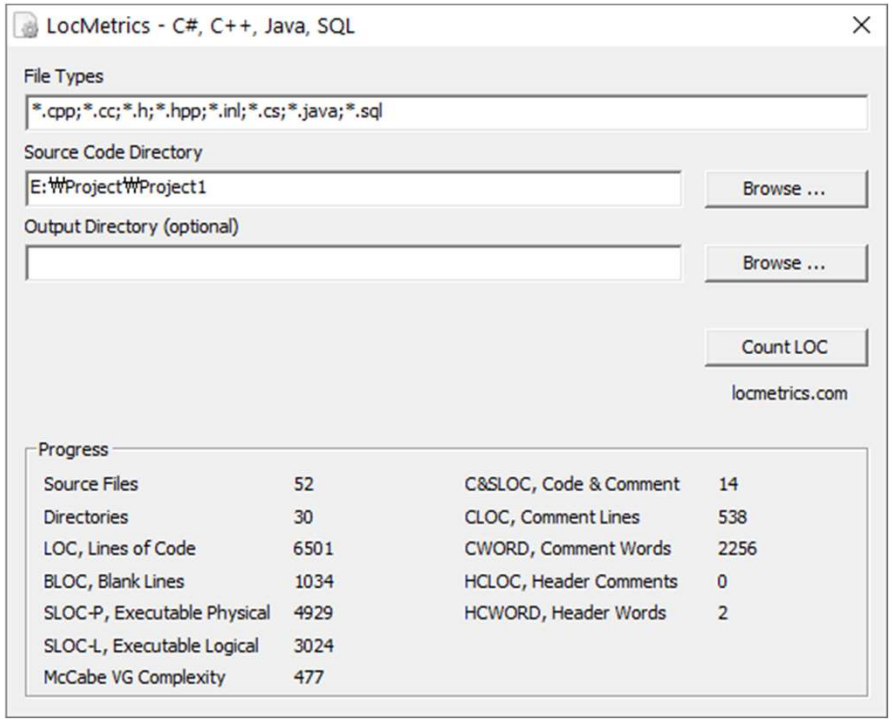
3.4.3. Độ đo hướng kích thước (tt)

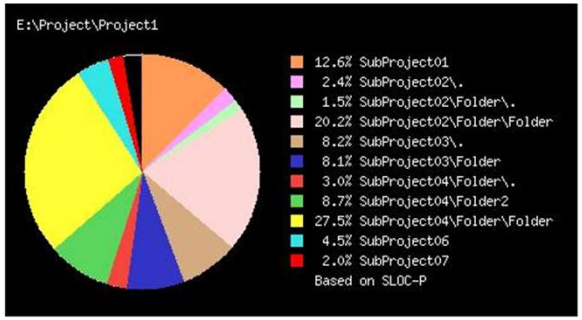
❖ Các vấn đề:

- ✓ Có thể nhiều chỉ thị/lệnh nằm trên cùng một dòng → để có LOC chính xác: chuẩn hóa cách viết, dùng các công cụ hỗ trợ,…
- ✓ Phụ thuộc vào ngôn ngữ lập trình: khó so sánh năng suất giữa các đề án, giữa các công ty có sử dụng ngôn ngữ lập trình khác nhau
- ✓ Số dòng mã nguồn chỉ thực sự chính xác sau khi đã hoàn tất phần mềm → phải có phương pháp để ước tính LOC

3.4.3. Độ đo hướng kích thước (tt)

❖ Ví dụ tính LOC: phần mềm Code Analyzer





Overall

Symbol	Count	Definition
Source Files	52	Source Files
Directories	30	Directories
LOC	6501	Lines of Code
BLOC	1034	Blank Lines of Code
SLOC-P	4929	Physical Executable Lines of Code
SLOC-L	3024	Logical Executable Lines of Code
MVG	477	McCabe VG Complexity
C&SLOC	14	Code and Comment Lines of Code
CLOC	538	Comment Only Lines of Code
CWORD	2256	Commentary Words
HCLOC	0	Header Comment Lines of Code
HCWORD	2	Header Commentary Words

❖ Phần mềm LocMetrics hoặc phần mềm VS2010 trở lên

3.4.4. Độ đo hướng chức năng

- Mục đích: có được con số đặc trưng cho hệ thống chức năng của mỗi phần mềm
- Đơn vị tính là FP (Function Point), không phụ thuộc vào ngôn ngữ lập trình
  - Có thể ước lượng sớm độ lớn phần mềm (nhờ các thông tin từ phân tích yêu cầu, thiết kế tổng thể, … của hệ thống)
  - Làm cơ sở chung để tính năng suất, chi phí
  - Ví dụ: Công ty ABC:
    - Chi phí trung bình là 1150 USD/FP
    - Mỗi nhân viên có năng suất lao động là 65FP/m (trung bình 65 đơn vị FP được làm ra trong 1 tháng)

3.4.4. Độ đo năng suất sản xuất PM

$$\text{Đo năng suất SXPM} = \frac{\text{Số điểm chức năng}}{\text{Số người tham gia x Thời gian}}$$

❖ Công thức tính FP gồm 4 bước:

- Bước 1: Xác định các đại lượng
- Bước 2: Tính tổng
- Bước 3: Tính các giá trị hiệu chỉnh độ phức tạp
- Bước 4: Tính số điểm chức năng

Cách tính FP (tt)

- Bước 1: Tính các đại lượng sau:
  - Số chức năng nhập liệu c1 (chú ý phân biệt với c3)
  - Số chức năng xuất dữ liệu c2 (báo biểu, màn hình xuất, thông báo lỗi).
  - Số chức năng truy vấn dữ liệu c3.
  - Số tập tin dữ liệu c4 (số bảng (CSDL quan hệ), số lớp (CSDL hướng đối tượng)).
  - Số các giao tiếp với hệ thống khác c5

Công thức tính FP (tt)

❖ Bước 2: Tính tổng:

$$\Delta = \Sigma_{i=1}^{5} C_i W_i$$

❖ Với w_i € [3,15] được chọn từ bảng sau:

W_i	Đơn giản	Trung bình	Phức tạp
W_1	3	4	6
W_2	4	5	7
W_3	3	4	6
W_4	7	10	15
W_5	5	7	10

Công thức tính FP (tt)

❖ Bước 3: Tính giá trị hiệu chỉnh

Tính các giá trị hiệu chỉnh độ phức tạp Fi (i=1, 2,… , 14) nhờ vào trả lời 14 câu hỏi và cho điểm từ 0 đến 5 tương ứng với các mức độ: ***không có, ít, vừa phải, trung bình, đáng chú ý, thật sự cần thiết***.

14 CÂU HỎI

1. Hệ thống đòi hỏi phải bảo đảm an toàn về việc cập nhật và cứu dữ liệu hay không?
2. Đòi hỏi việc truyền thông hay không?
3. Có các chức năng xử lý phân bố hay không?
4. Vấn đề tốc độ có quan trọng hay không?
5. Hệ thống có đòi hỏi cấu hình mạnh ?
6. Có đòi hỏi nhập dữ liệu trực tuyến hay không?
7. Dữ liệu nhập trực tuyến (nếu có) có đòi hỏi giao dịch (transaction) hay không (do có nhiều màn hình nhập hay nhiều thao tác đồng thời) ?

Công thức tính FP (tt)

8. Dữ liệu lưu trữ được cập nhật trực tuyến?
9. Có yêu cầu các thao tác nhập xuất hay các câu truy vấn phức tạp không?
10. Xử lý bên trong có phức tạp không?
11. Mã nguồn có cần thiết kế để có thể dùng lại không?
12. Sự chuyển đổi dữ liệu và cài đặt hệ thống có được bao gồm trong giai đoạn thiết kế không?
13. Hệ thống có được thiết kế để cài đặt cho nhiều tổ chức khác nhau không?
14. Hệ thống có được thiết kế để dễ dàng thay đổi và dễ dàng sử dụng bởi người dùng không?

Công thức tính FP (tt)

- Bước 4: Tính điểm chức năng:

$$FP = \Delta \times (0.65 + 0.01 \times \Sigma Fi)$$

Ví dụ

- Một công ty sản xuất phần mềm có hiệu suất sản xuất là **14FP/pm**. Chi phí trả mỗi người/tháng là **500USD**. Hãy ước lượng chi phí của phần mềm quy ra **USD** và đơn vị **pm.**
- Công ty ước lượng các thông số như sau :

Điểm chức năng (Ci)		Trọng lượng (Wi)
• Số chức năng nhập liệu	9	4
• Số chức năng xuất dữ liệu	11	5
• Số chức năng truy vấn	8	6
• Số bảng quan hệ (trong CSDL)	12	10
• Số giao tiếp ngoài	4	7

Ví dụ

❖ Các thông tin liên quan đến hệ số hiệu chỉnh fi:

Fi	Điểm
1. Cập nhật và cứu dữ liệu	4
2. Truyền thông	2
3. Xử lý phân bố	1
4. Vấn đề tốc độ	3
5. Vấn đề môi trường	1
6. Nhập dữ liệu trực tuyến	5
7. Transaction nhập dữ liệu	4
8. Cập nhật trực tuyến	4
9. Nhập, xuất, truy vấn phức tạp	3
10.Xử lý phức tạp	3
11.Mã nguồn dùng lại	5
12.Cài đặt	1
13.Nhiều nơi dùng	4
14.Dễ thay đổi	2

Ví dụ - Giải:

- Tính

  $\Delta = \sum_1^5 ciwi = 9 \times 4 + 11 \times 5 + 8 \times 6 + 12 \times 10 + 4 \times 7 = 287$

  $\sum_1^{14} f_i = 4 + 2 + 1 + 3 + 1 + 5 + 4 + 4 + 3 + 3 + 5 + 1 + 4 + 2$

  = 42

- Số lượng FP = Δ x (0.65 + 0.01x $\sum f_i$) = **307.09**

- Chi phí mỗi FP = $\frac{\text{Chi phí trả cho 1 người}}{\text{Hiệu suất theo tháng}} = \frac{500}{14} = \mathbf{35.7}usd$

- Chi phí quy ra tiền USD = Số lượng FP x Chi phí mỗi FP

  = 307.09 x 35.7 = **10,963.113** USD

- Quy ra đơn vị pm= $\frac{số\ lượng\ FP}{năng\ suất\ sản\ xuất} = \frac{307.09}{14} = \mathbf{21.94}\ pm$

3.4.5. Quan hệ giữa LOC và FP

Ngôn ngữ lập trình	Số LOC/1 FP
Hợp ngữ	320
C	128
Cobol	105
Fortran	105
Pascal	90
Ada	70
Các NNLT Hướng đối tượng	30
Ngôn ngữ thế hệ 4 (4GLs)	20
Các bộ phát sinh mã	15
Bảng tính	6
Ngôn ngữ ICON	4

3.4.6. Độ đo chất lượng theo qui trình

- Cần xác định: mật độ lỗi trong giai đoạn kiểm tra phần mềm
- Việc khử lỗi trong mỗi pha của chu kỳ sống
- Đánh giá sự khử lỗi (sự hiệu quả của việc khử lỗi)

3.4.7. Khử lỗi trong mỗi pha

- Việc phát hiện lỗi được dựa vào:
  - Xem xét lại bảng thiết kế
  - Xem xét lại mã nguồn
  - Kiểm tra hình thức trước khi kiểm tra phần mềm
- Xem xét các dự án nhằm:
  - Theo dõi việc khử lỗi trong từng đề án một
  - So sánh nhiều đề án với nhau
  - Tính giá trị trung bình các số liệu cần thiết dựa trên nhiều đề án đã triển khai

3.4.8. Đánh giá sự khử lỗi DRE

$$DRE = \frac{\text{Số lỗi khử được trong một pha phát triển phần mềm}}{\text{Số lỗi tiềm tang của sản phẩm}} \times 100\%$$

- DRE: Defect Removal Effectiveness
- Số lỗi tiềm tàng (toàn bộ lỗi phát hiện) = Số lỗi đã khử được + Số lỗi tìm được sau này (do khách hàng báo lại, do tình cờ…).

3.4.8. Đánh giá sự khử lỗi

❖ DRE có thể được tính cho từng pha một của chu kỳ sống hay tính cho toàn bộ qui trình phát triển phần mềm.

❖ Một tổ chức sản xuất có thể theo dõi chỉ số này theo các phiên bản liên tiếp nhau của một phần mềm để khai thác được nhiều thông tin hơn.

❖ Để đánh giá chất lượng qui trình sản xuất của một tổ chức sản xuất phần mềm, người ta còn dựa vào các chuẩn về phần mềm được đưa ra để đánh giá mức độ trưởng thành của một phần mềm. Trong mỗi chuẩn, người ta có thể đề nghị nhiều độ đo khác nhau và đề nghị qui trình để đánh giá chất lượng

3.4.9. Độ đo bảo trì phần mềm

$$\text{Mức độ bảo trì} = \frac{\text{Số vấn đề giải quyết trong tháng}}{\text{Tổng số vấn đề nảy sinh trong tháng}}$$

- Thời gian trung bình đáp ứng cho khách hàng $\frac{\sum_{n}^{1} t_i}{n}$
  - n: tổng số vấn đề nảy sinh trong tháng
  - ti: thời gian giải quyết hoàn tất 1 vấn đề thứ i
  - i ∈ [1-n]

3.4.9. Độ đo bảo trì phần mềm (tt)

- Sự chểnh mảng trong công tác bảo trì. RTC (Reponse Time Criteria).

$$\text{Độ đo} = \frac{\text{Số bảo trì có thời gian vượt quá RTC}}{\text{Tổng số lần bảo trì}}$$

Chất lượng của việc sửa lỗi: Sự sai lầm trong việc sửa đổi, không sửa chữa được lỗi do khách hàng báo cáo lại hay sửa chữa được nhưng lại làm nảy sinh ra một hay nhiều lỗi khác.

$$\text{Độ đo} = \frac{\text{Số lần sai lầm trong sửa đổi}}{\text{Tổng số lần sửa đổi}}$$

CHƯƠNG 4. QUẢN LÝ CHẤT LƯỢNG PHẦN MỀM
(SOFTWARE QUALITY MANAGEMENT)

Khái niệm

- Quản lý chất lượng (software quality management): tiến trình liên quan đến việc bảo đảm các yêu cầu về cấp độ chất lượng (level fo quality) của sản phẩm phần mềm
- Quản lý chất lượng bao gồm định nghĩa những tiêu chuẩn chất lượng và các thủ tục thích hợp đồng thời, bảo đảm việc phát triển phần mềm theo đúng tiêu chuẩn đó
- Đối với bất kỳ 1 tổ chức phần mềm nào thì hoạt động xây dựng mục tiêu chất lượng sản phẩm là trách nhiệm của mọi thành viên (quality culture)

Thế nào là chất lượng ?

- Chất lượng sản phẩm nghĩa là sản phẩm làm ra phải **đúng với chi tiết kỹ thuật** của nó
- Một số khó khăn trong việc quản lý chất lượng phần mềm:
  - Áp lực đòi hỏi chất lượng của khách hàng (hiệu quả, độ tin cậy,…) và các yêu cầu về chất lượng của nhà thiết kế (dễ bảo trì, tái sử dụng,…)
  - Một vài yêu cầu về chất lượng thì khó mô tả một cách rõ ràng
  - Đặc tả chi tiết phần mềm thường không đầy đủ và thiếu nhất quán

Thế nào là sản phẩm PM chất lượng ?

- Đáp ứng được các yêu cầu kỹ thuật đề ra
- Kiểm soát dữ liệu nhập và dự toán được các dữ liệu nhập không hợp lệ
- Đã được kiểm tra và thử nghiệm hoàn toàn bởi những người độc lập
- Có tài liệu hướng dẫn cụ thể và rõ ràng
- Nhận biết được tỉ lệ lỗi

Các tính chất cơ bản của chất lượng phần mềm

Mục tiêu của Quản lý chất lượng PM

Một hệ thống quản lý chất lượng phần mềm thường có 2 mục tiêu:

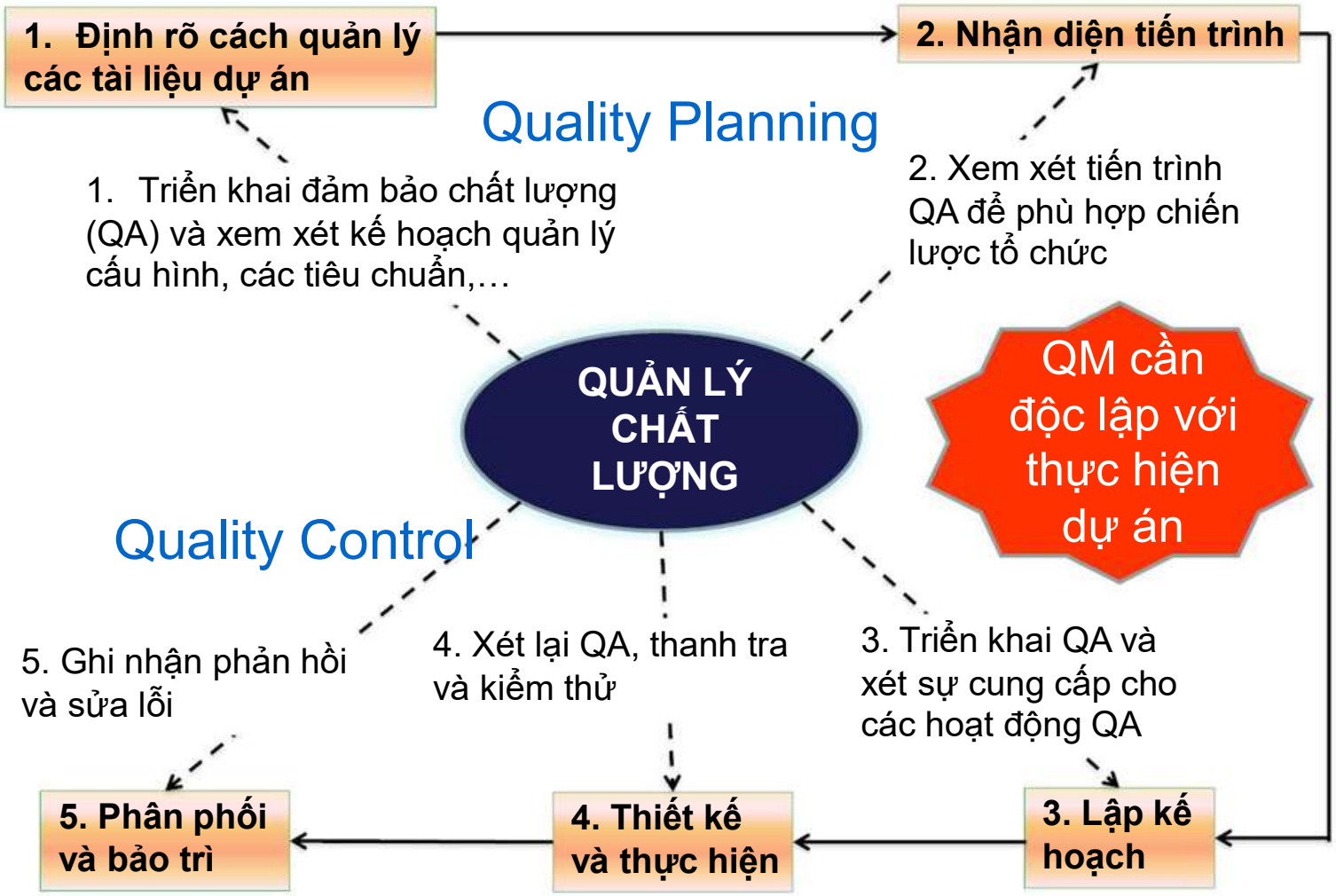
- **Mục tiêu thứ nhất:** xây dựng chất lượng cho PM ngay từ giai đoạn bắt đầu. Điều này đồng nghĩa với việc bảo đảm các yêu cầu cho PM từ mọi nguồn khác nhau phải được định nghĩa, diễn đạt và hiểu một cách đúng đắn, giữa người đưa ra yêu cầu và người thực hiện yêu cầu
- **Mục tiêu thứ hai:** bảo đảm chất lượng của PM xuyên suốt quá trình phát triển

Các thỏa thuận về chất lượng

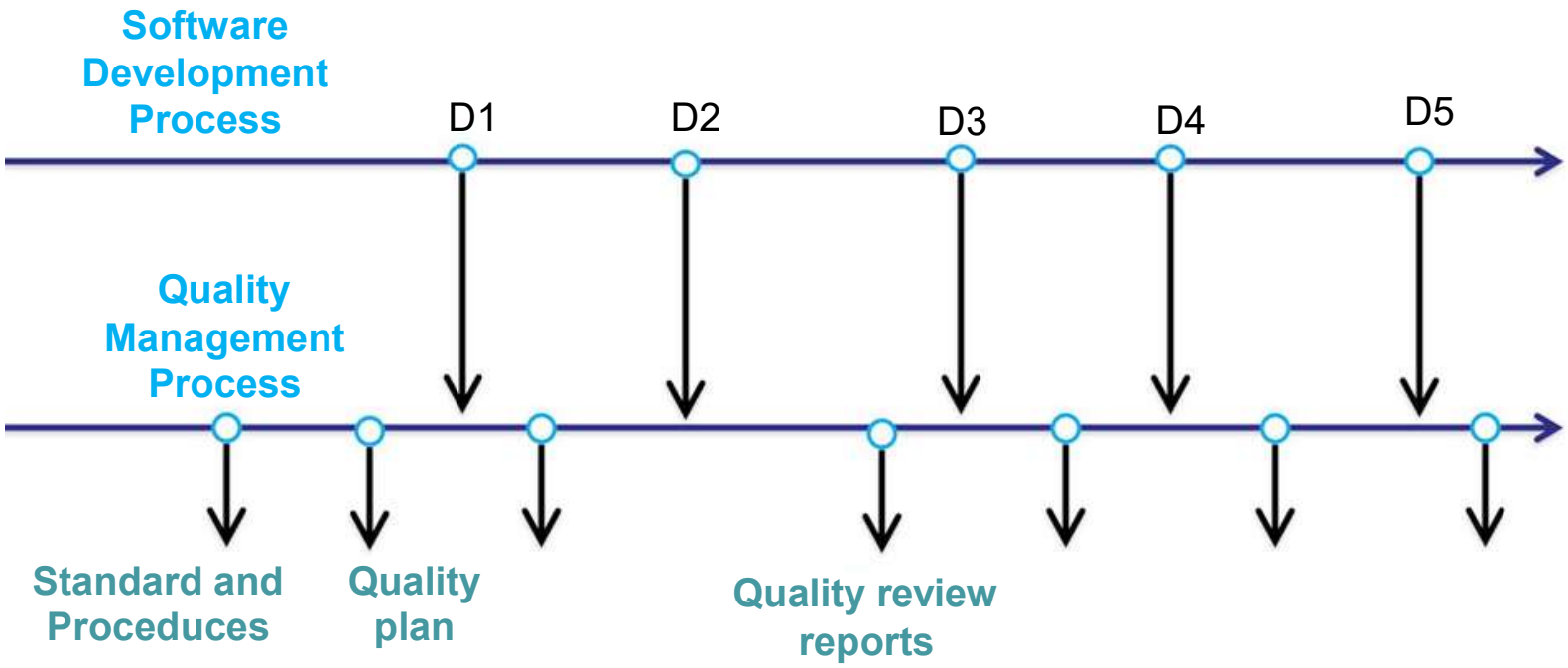
- Không cần đợi biết rõ về chi tiết sản phẩm mới cải tiến mà cần chú ý quản lý chất lượng trước
- Cần đặt các thủ tục cải tiến chất lượng bất chấp các chi tiết vẫn còn đang chưa hoàn chỉnh

→ Quản lý chất lượng không chỉ liên quan đến việc giảm tỉ lệ lỗi mà còn liên quan đến các tiêu chuẩn chất lượng sản phẩm khác

Các thỏa thuận về chất lượng



Quản lý chất lượng và phát triển phần mềm
(Quality Management and Software Development)



10 hoạt động và yếu tố cơ bản

1. Các tiêu chuẩn (Standards)

2. Lập kế hoạch (Planning)

3. Xem xét, xem lại (Reviewing)

4. Kiểm tra (Testing)

5. Phân tích lỗi (Defect analysis)

6. Quản lý cấu hình (Configuration Management)

7. Bảo mật (Security)

8. Đào tạo, huấn luyện (Education/Training)

9. Quản lý người cung cấp, thầu phụ (Vendor Management)

10. Quản lý rủi ro (Risk Management)

1. Các tiêu chuẩn đảm bảo chất lượng
(Quality assurance and standards)

- Sản xuất phần mềm không còn đơn thuần mang tính ngẫu hứng, mà đang trở thành một lĩnh vực được kiểm soát chặt chẽ, theo những tiêu chuẩn nhất định.
- Các tiêu chuẩn có thể là:
  - Các kinh nghiệm hoặc các phương pháp hiệu quả nhất, được đề xuất từ các tổ chức IEEE, ISO, ...
  - Các quy tắc chuẩn hóa để giao tiếp giữa sản phẩm với nhau
  - Có thể do chính tổ chức phát triển PM đề ra
- Các tiêu chuẩn nên được chọn và thể hiện sao cho khi sử dụng, các khía cạnh kỹ thuật cần thiết sẽ được nhấn mạnh, tránh trường hợp hiểu sai

Đặc điểm và vai trò của tiêu chuẩn

- Đặc điểm:
  - Tính cần thiết
  - Tính khả thi
  - Tính đo lường được
- Vai trò:
  - Là chìa khóa tăng hiệu quả quản lý chất lượng
  - Tóm lược được các hoạt động tốt nhất, tránh lặp lại các sai lầm đã qua
  - Đảm bảo chất lượng có đúng theo chuẩn đã đưa ra
  - Cung cấp nội dung về tiêu chuẩn một cách liên tục để nhân viên mới có thể hiểu và áp dụng

2. Lập kế hoạch chất lượng (Quality planning)

- Lập kế hoạch là yêu cầu kinh điển của hầu hết hệ thống quản lý chất lượng phần mềm. Kết quả của hoạt động này thường là một tài liệu gọi là bản kế hoạch
- Ví dụ kế hoạch cho 1 dự án phần mềm bao gồm:
  - Ước lượng phạm vi và kích thước dự án, khối lượng công việc phải làm
  - Xác định nhân lực, vật lực và chi phí
  - Chỉ định phương pháp, cách tiếp cận để thực thi dự án
  - Lập kế hoạch làm việc chi tiết
  - Kế hoạch phối hợp và hỗ trợ hoàn thành dự án
- Một bản kế hoạch về chất lượng phải được tiến hành từ sớm, phải định nghĩa và đánh giá được các thuộc tính chất lượng quan trọng nhất

Cấu trúc bản kế hoạch chất lượng
(Quality plan structure)

- Giới thiệu sản phẩm
- Kế hoạch phát triển sản phẩm
- Mô tả tiến trình
- Mục tiêu chất lượng (Quality Goals)
- Quản lý rủi ro (Risk Management)

→ Kế hoạch quản lý chất lượng nên ngắn gọn và súc tích, nếu quá dài thì sẽ không có ai muốn đọc chúng

IEEE 730-1989 Software Quality Assurance Plans

1. **Purpose**
2. **Referenced documents**
3. **Management**
    1. Organization
    2. Tasks
    3. Responsibilities
4. **Documentation**
    1. Purpose
    2. Minimum documentation requirements
    3. Other
5. **Standards, practices, convention and metrics**
    1. Purpose
    2. Content
6. **Review and audits**
    1. Purpose
    2. Minimum requirements
    3. Other
7. **Testing**
8. **Problem reporting and corrective action**
9. **Tools, techniques and methodologies**
10. **Code control**
11. **Media control**
12. **Supplier control**
13. **Records collection, maintenance and retention**
14. **Tranning**
15. **Risk management**

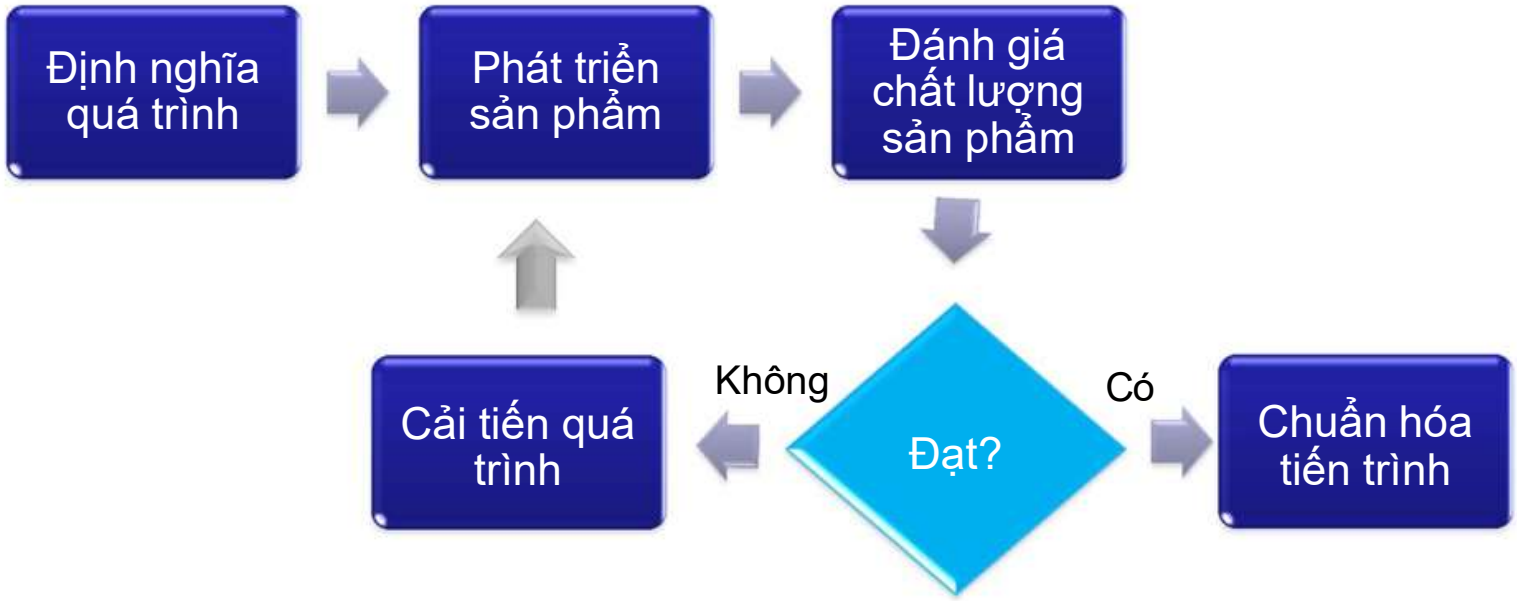
3. Chất lượng phần mềm và tiến trình
(Process and Product Quality)

- Chất lượng của một sản phẩm phần mềm bị ảnh hưởng nhiều bởi chất lượng của tiến trình xây dựng sản phẩm
- Thuộc tính của sản phẩm phụ thuộc vào chức năng của tiến trình
- Việc quan trọng đặc biệt trong phát triển phần mềm là một số thuộc tính chất lượng sản phẩm rất khó có thể đánh giá → Đây là điều rất phức tạp và khó hiểu về mối quan hệ giữa tiến trình phần mềm và chất lượng sản phẩm

Quản lý chất lượng theo tiến trình
(Process-based Quality)

- Có sự liên kết rõ ràng giữa tiến trình và sản phẩm trong việc sản xuất sản phẩm hàng hóa
- Đối với sản phẩm phần mềm thì phức tạp hơn vì:
  - Việc áp dụng các kỹ năng cá nhân và kinh nghiệm là điều quan trọng đặc biệt trong việc phát triển phần mềm
  - Những yếu tố bên ngoài (*như tính mới lạ của một ứng dụng hoặc rút ngắn thời gian triển khai*) → có thể làm hư hại chất lượng sản phẩm

Chất lượng tiến trình (cơ bản)



Chất lượng tiến trình thực tế
(Practical process quality)

- **Định nghĩa các tiêu chuẩn của tiến trình**: gồm quá trình xem xét, quản lý cấu hình,…
- **Quan sát sự phát triển của tiến trình**: để đảm bảo rằng các tiểu chuẩn được thực hiện nghiêm túc
- Báo cáo tiến trình đến Trưởng dự án (Project Manager)

4. Kiểm soát chất lượng
(Quality Control)

- Kiểm tra tiến trình phát triển phần mềm để đảm bảo các thủ tục và các tiêu chuẩn đã được thực hiện đúng
- Hai phương pháp tiếp cận:
  - Xem xét chất lượng (Quality reviews)
  - Đánh giá qua việc đo phần mềm (assessment via software metrics)

5. Đánh giá lại chất lượng
(Quality reviews)

- Đây là biện pháp chủ yếu để xác nhận tính hợp lệ về chất lượng của một tiến trình hay một sản phẩm
- Nhóm kiểm tra một phần hay tất cả quá trình hay hệ thống và tài liệu của nó để phát hiện những vấn đề tiềm ẩn
- Có 3 loại reviews khác nhau với các mục tiêu khác nhau:
  - ✓ Thanh tra để gỡ bỏ các lỗi (product)
  - ✓ Xem xét để đánh giá sự tiến triển (product and process)
  - ✓ Xem xét chất lượng (product and standards)

6. Quản lý rủi ro
(Risk control)

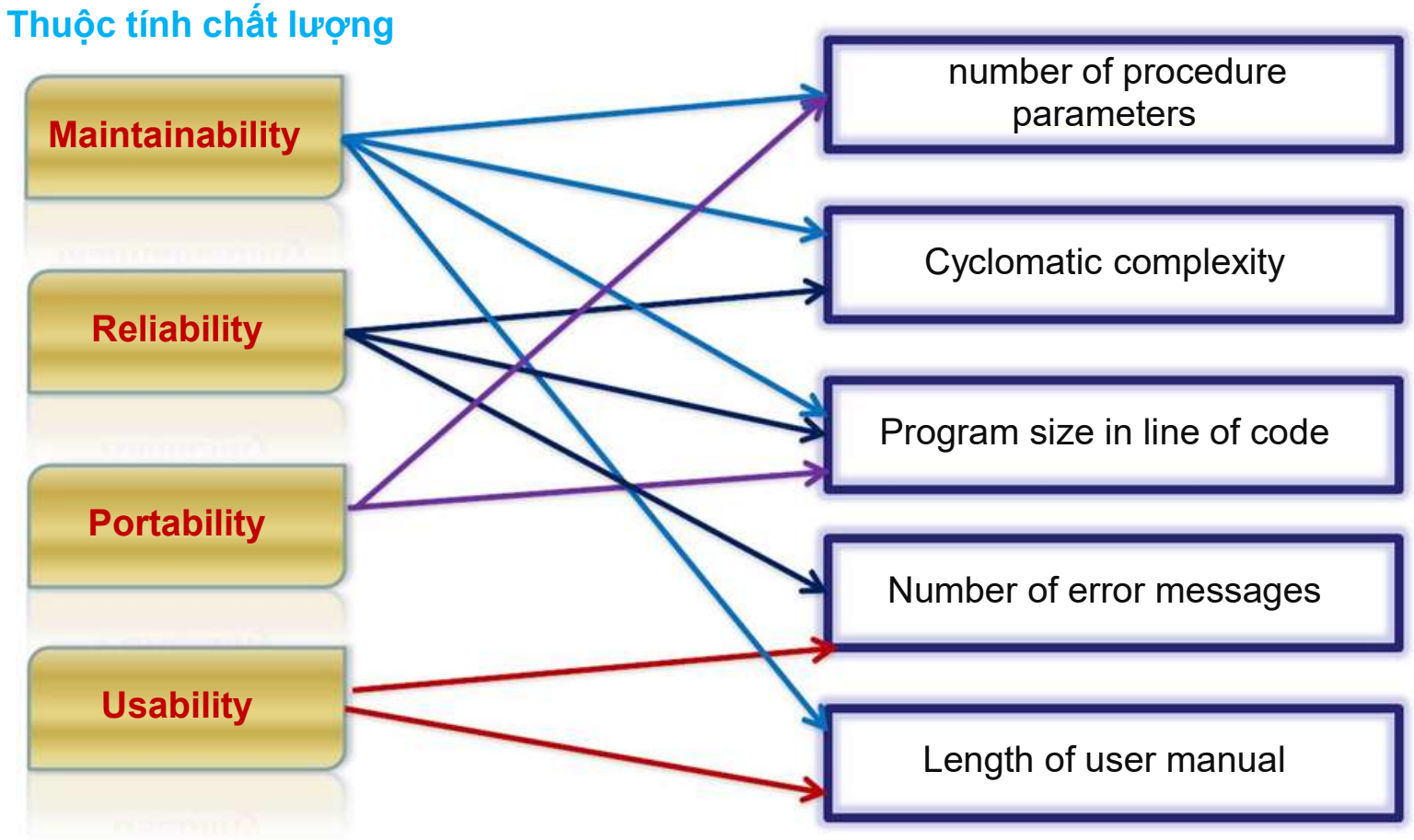
- Rủi ro là một sự kiện xấu chưa nhưng có khả năng xảy ra cản trở khả năng hoàn thành dự án.

- **Phân loại rủi ro:**
  - ✓ Về kỹ thuật: Cần hiểu đúng và đủ về dự án -> đưa ra giải pháp đúng để giải quyết
  - ✓ Về quản lý: gồm rửi ro về tài chính, kế hoạch, nhân lực, thay đổi yêu cầu.
  - ✓ Về vận hành: Huấn luyện không đầy đủ, sử dụng sai chức năng PM,…
  - ✓ Về môi trường: môi trường phát triển, virus,…
  - ✓ Về kiểm tra: không đúng, không đủ yêu cầu.

6. Quản lý rủi ro
(Risk control)

- Quy trình quản lý rủi ro gồm 4 bước
  1. Nhận biết rủi ro
  2. Khảo sát mức tác động (bị ảnh hưởng)
  3. Xác định các giải pháp
  4. Giám sát thực thi giải pháp cho từng rủi ro
- Phân loại giải pháp
  1. Loại bỏ hoàn toàn rủi ro mặc dù kinh phí thấp hay sẽ gây ảnh hưởng nghiêm trọng cho dự án
  2. Phòng tránh
  3. Giảm thiểu thiệt hại:
  4. Chấp nhận: nếu rủi ro không quá lớn, hoặc khả năng xảy ra cực thấp

Các độ đo phần mềm và thuộc tính chất lượng
(Quality attributes and software metrics)



Tiến trình đo phần mềm
(Measurement process)

- Tiến trình đo phần mềm là tiến trình con của tiến trình kiểm soát chất lượng
- Dữ liệu thu thập trong toàn bộ tiến trình con này cần được lưu trữ như tài nguyên
- Cơ sở dữ liệu đo được sẽ làm cơ sở để so sánh với các dự án khác

Độ đo sản phẩm
(Product metrics)

- Tiến trình đo chất lượng là cơ sở để xác định chất lượng sản phẩm phần mềm
- Phân loại độ đo sản phẩm:
  - **Dynamic metrics**: liên quan trực tiếp đến thuộc tính chất lượng, dữ liệu đo được tạo ra trong quá trình thực thi chương trình (thời gian đáp ứng, số lượng lỗi,…) → dùng đánh giá tính hiệu quả và tin cậy
  - **Static metrics**: liên quan gián tiếp đến thuộc tính chất lượng, dữ liệu đo được trong quá trình biểu diễn hệ thống phần mềm (số dòng mã lệnh) → dùng để đánh giá tính phức tạp, tính dễ hiểu và khả năng bảo trì

Một số thuật ngữ đo sản phẩm phần mềm
(Software Product metrics)

Software metric	Description
Fan-in/Fan-out	**Fan-in** is a measure of the **number** of functions or methods that **call** another function or method (say X). **Fan-out** is the number of functions that **are called** by function X. A **high** value for **fan-in** means that X is **tightly coupled** to the rest of the design and **changes** to X will have **extensive** knock-on **effects**. A **high** value for **fan-out** suggests that the **overall complexity** of X may be **high** because of the **complexity** of the control logic needed to **coordinate** the called components.
Length of code	This is a measure of the **size** of a program. Generally, the **larger** the size of the code of a component, the more **complex** and **error-prone** that component is likely to be. Length of code has been shown to be one of the most reliable metrics for **predicting error-proneness** in components.

Một số thuật ngữ đo sản phẩm phần mềm
(Software Product metrics)

Software metric	Description
Cyclomatic complexity	This is a measure of the **control complexity** of a program. This control complexity may be related to program **understandability**.
Length of identifiers	This is a measure of the **average length of identifiers** (names for variables, classes, methods, etc.) in a program. The **longer** the identifiers, the more likely they are to be **meaningful** and hence the more **understandable** the program.
Depth of conditional nesting	This is a measure of the **depth of nesting** of **if-statements** in a program. **Deeply** nested if-statements are hard to **understand** and potentially **error-prone**.
Fog index	This is a measure of the **average length of words** and sentences in **documents**. The **higher** the value of a document's Fog index, the more difficult the document is to understand.

Kết luận

- Một HTQLCLPM không chỉ có quy trình, kiểm tra hoặc ra lệnh, nó là sự kết hợp gắn bó của nhiều yếu tố
- Mục tiêu sau cùng của HTQLCLPM kiểm soát chất lượng, nâng cao lợi ích của công ty và đảm bảo chất lượng PM
- Đây là nghề mới, thu hút giới trẻ với thu nhập khá cao. Yêu cầu:
  - khả năng trình bày, thuyết phục tốt;
  - biết lắng nghe, để thấy cái gì cần điều chỉnh thì điều chỉnh và cũng để thuyết phục tốt hơn;
  - kỹ năng đo lường và phân tích số liệu;
  - kỹ năng làm phần mềm

CHƯƠNG 5.
QUẢN LÝ CẤU HÌNH PHẦN MỀM
(SOFTWARE CONFIGURATION MANAGEMENT)

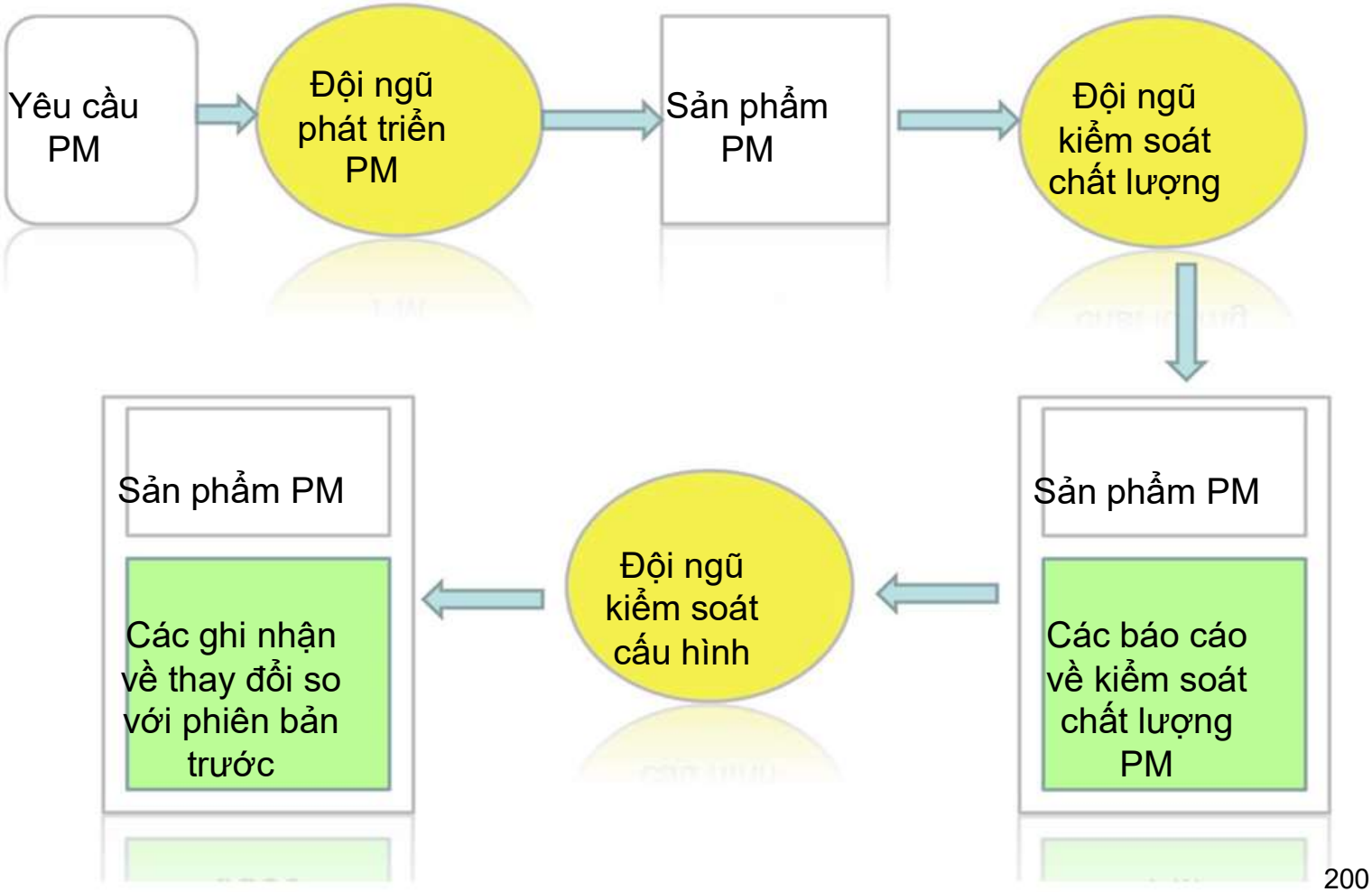
Nội dung

1. Tổng quan về phần mềm
2. Hoạch định quản lý cấu hình
3. Quản lý sự thay đổi phần mềm
4. Quản lý phiên bản
5. Tích hợp hệ thống từ các thành tố

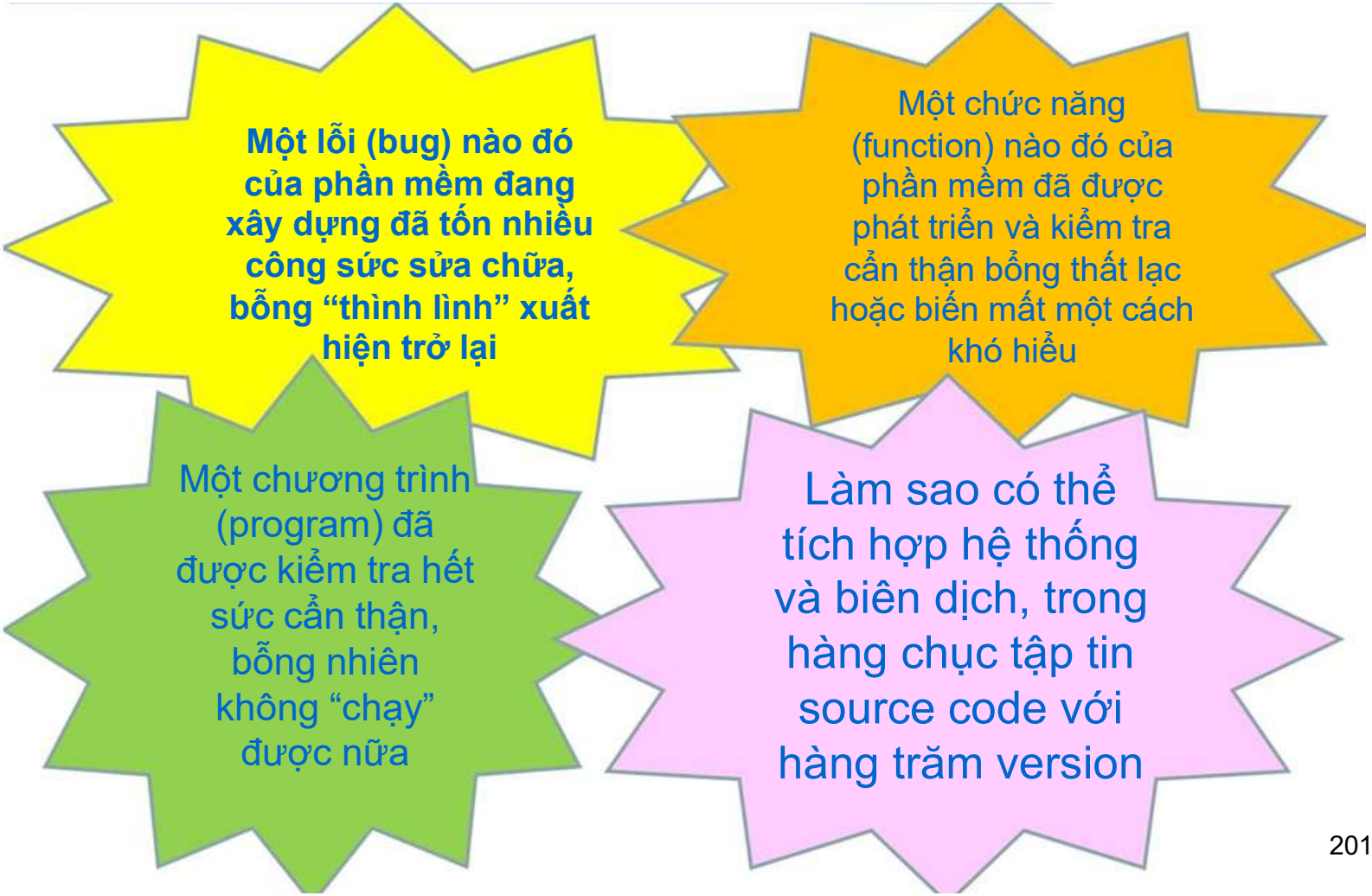
Khái niệm

- Quản lý cấu hình (software configuration management - SCM):
  - Là tiến trình kiểm soát, theo vết các thay đổi của một hệ thống phần mềm
  - Được dùng để quản lý các phiên bản khác nhau của phần mềm đó
- «Mục đích của QLCH là để thiết lập và bảo đảm tính toàn vẹn của các sản phẩm trung gian cũng như các sản phẩm sau cùng của một dự án phần mềm, xuyên suốt chu kỳ sống của dự án đó» – trích từ CMMI và ISO 15504

Mối liên hệ giữa QLCH và QLCL



Tại sao cần quản lý cấu hình ?



Tại sao cần quản lý cấu hình ? (tt)

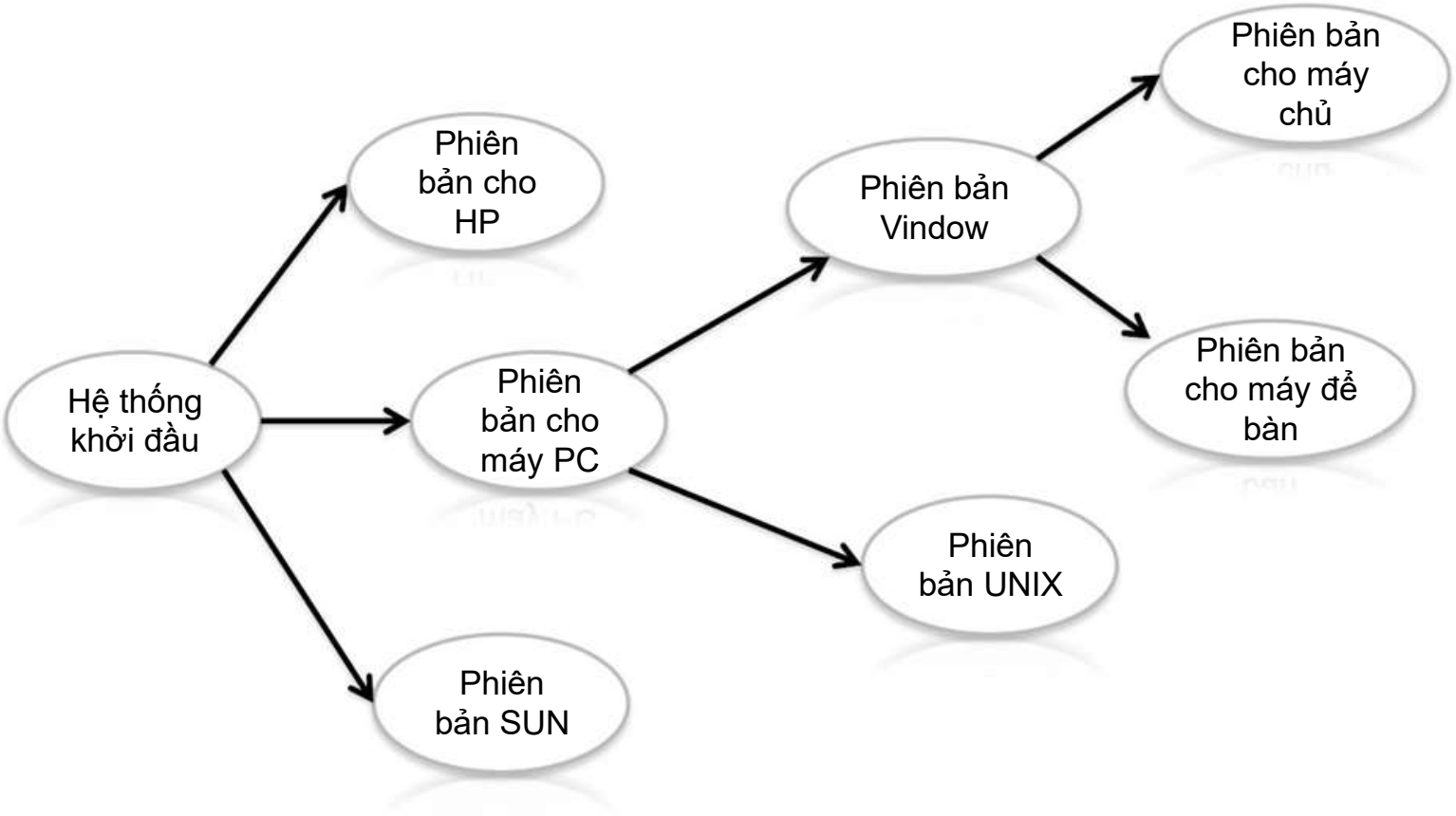
- Phần mềm chạy trên **nhiều họ máy tính** khác nhau
- Phần mềm chạy trên **nhiều hệ điều hành** khác nhau
- Gồm các chức năng được phát triển cho một nhóm **khách hàng cụ thể**
- Sử dụng đa ngôn ngữ

⇒ Cần phải sử dụng **cây quản lý cấu hình** phần mềm

Chức năng cây quản lý cấu hình:

- ❖ Theo dõi và quản lý **sự khác nhau** giữa các phiên bản
- ❖ Đảm bảo các phiên bản mới được bắt nguồn (kế thừa) từ phiên bản gốc trong **một tiến trình được kiểm soát** (không được tùy tiện, tự phát)
- ❖ Đảm bảo các phiên bản mới được **giao đến đúng khách hàng và đúng thời gian quy định**

Sơ đồ minh họa cây QLCH



Chuẩn về quản lý cấu hình ?

- Trong mỗi tổ chức sản xuất phần mềm, những quy định chung về quản lý cấu hình được công bố trong "*sổ tay quản lý cấu hình*" hoặc "*cẩm bang bảo đảm chất lượng phần mềm*". Các quy định này có thể xuất phát từ các chuẩn tổng quát như: IEEE 828-1983, ISO 9000, CMMI,…
- Một số chuẩn về quản lý cấu hình được phát triển theo qui ước mô hình thác nước được dùng để phát triển hệ thống. Do đó, không thể áp dụng hiệu quả cho các tiếp cận phát triển phần mềm theo các mô hình chu kỳ sống tiến hóa

Các hoạt động trong tiến trình QLCH

- Bao gồm 4 hoạt động chính (tiến trình con) sau đây:
  1. Hoạch định quản lý cấu hình
  2. Quản lý sự thay đổi phần mềm
  3. Quản lý phiên bản của phần mềm
  4. Xây dựng phần mềm từ các thành tố

Lưu ý:

- *Tiến trình quản lý cấu hình chỉ thực sự chạy sau khi một phiên bản đầu của hệ thống được phát triển*
- *Tuy nhiên, một số đề xuất hoạch định tiến trình nên bắt đầu ngay khi khởi động đề án và hoạt động suốt thời gian phát triển hệ thống*

Các thông tin trong phương án QLCH

- Định nghĩa các thực thể (dùng cho HĐH nào? Dành cho loại máy nào? Các lỗi đã được fix ? Các cải tiến mới,....) được quản lý và đưa ra cách thức chặt chẽ để nhận ra các thực thể này
- Quy định người chịu trách nhiệm về các thủ tục quản lý cấu hình và trình bày giải thích các thực thể được quản lý trước đội ngũ quản lý cấu hình
- Các phương pháp dùng để quản lý phiên bản và kiểm soát thay đổi
- Mô tả CSDL dùng để ghi thông tin cấu hình
- Các thông tin khác: quản lý các phần mềm được sử dụng cho đề án, quản lý thủ tục kiểm tra tiến trình quản lý cấu hình,…

Các đối tượng cấu hình được quản lý

- Đặc tả hệ thống
- Kế hoạch dự án phần mềm
- Đặc tả yêu cầu
  - Đặc tả yêu cầu phần mềm
  - Nguyên mẫu thi hành được hoặc nguyên mẫu giấy tờ
  - Sổ tay sử dụng sơ cấp
- Các đặc tả thiết kế
  - Dữ liệu
  - Kiến trúc
  - Module
  - Giao diện
  - Đối tượng (hướng đối tượng)

Các đối tượng cấu hình được quản lý (tt)

- Mã nguồn và kiểm thử
  - Kế hoạch và thủ tục kiểm thử
  - Các ca kiểm thử và các kết quả được ghi lại
  - Các sổ tay vận hành và sổ tay lắp đặt
  - Chương trình thi hành được
  - Các module và mã thi hành được
  - Các module đã liên kết
  - Bộ dữ liệu thử nghiệm
- Mô tả cơ sở dữ liệu
  - Lược đồ và cấu trúc các file
  - Nội dung hồ sơ ban đầu

Các đối tượng cấu hình được quản lý (tt)

- Sổ tay người sử dụng
  - Các tài liệu bảo trì
  - Các báo cáo những vấn đề phần mềm
  - Các yêu cầu bảo trì
  - Đặt tả thay đổi kỹ nghệ
  - Các chuẩn và các thủ tục cho kỹ nghệ phần mềm

Các đối tượng cấu hình được quản lý (tt)

- Đặt tên các đối tượng cấu hình: cần đảm bảo các điều kiện sau:
  - Một đối tượng phải có một tên duy nhất trong toàn bộ hệ thống quản lý cấu hình
  - Tên đối tượng nên xúc tích, gợi nhớ và bao hàm ý nghĩa của đối tượng
  - Cách đặt tên hàm ý các mối liên hệ giữa các đối tượng được quản lý (như mối quan hệ giữa văn bản thiết kế và mã nguồn của một phiên bản nào đó cho bản thiết kế đó)
  - Bản thân phương pháp đặt tên đối tượng phải được kiểm soát bởi hệ thống cấu hình (tự động hoặc không), không được tùy ý đặt tên

CSDL dùng để quản lý cấu hình

- Ghi tất cả các thông tin liên quan quản lý cấu hình
- Các chức năng chính:
  - Theo dõi
  - Kiểm soát từ thay đổi
  - Cung cấp các thông tin quản lý
- Các tác vụ của tiến trình hoạch định QLCH liên quan đến khía cạnh CSDL gồm:
  - Định nghĩa lược đồ CSDL
  - Định nghĩa các thủ tục quản lý thông tin
  - Định nghĩa các thủ tục truy xuất thông tin
- Các CSDL có thể được cài đặt tích hợp hoặc riêng lẻ
- Các câu truy vấn thông thường : 6 câu

CSDL dùng để quản lý cấu hình (tt)

Các câu truy vấn thường dùng:

1. Có bao nhiêu khách hàng sử dụng một phiên bản cụ thể của phần mềm ?
2. Phải dùng cấu hình HĐH và phần cứng nào để chạy một phiên bản của một phần mềm đã cho ?
3. Một phần mềm nào đó có bao nhiêu phiên bản và được tạo lập vào những ngày nào ?
4. Những phiên bản nào của một phần mềm bị ảnh hưởng nếu thay đổi một thành tố cụ thể nào đó ?
5. Có bao nhiêu yêu cầu thay đổi phần mềm còn tồn đọng đối với một phiên bản cụ thể ?
6. Có bao nhiêu lỗi đã được ghi nhận cho một phiên bản cụ thể của phần mềm ?

Tiến trình

- Tiến trình quản lý sự thay đổi phần mềm bao gồm:
  - Phân tích sự đánh giá về mặt kỹ thuật
  - Đánh giá chi phí
  - Theo dõi "vết" của thay đổi

Mô tả thuật toán của tiến trình quản lý sự thay đổi phần mềm

*Yêu cầu thay đổi bằng cách điền vào "**mẫu yêu cầu thay đổi**"*
Phân tích các yêu cầu thay đổi
Nếu *<thay đổi hợp lý>* thì {
    Đánh giá sơ bộ cách thức cài đặt các thay đổi
    Ước lượng phí tổn
    Ghi nhận các yêu cầu thay đổi vào CSDL
    Đệ trình yêu cầu thay đổi tới Bộ phận kiểm soát thay đổi
    Nếu *thay đổi được chấp nhận* thì {
        lặp lại {
            tiến hành phần mềm thay đổi theo yêu cầu
            Ghi nhận thay đổi, liên kết với yêu cầu thay đổi
            Đệ trình phần mềm để phê chuẩn chất lượng
        } *đến khi (chất lượng phần mềm có thể chấp nhận)*
        Tạo phiên bản mới }
    Ngược lại
        Bác bỏ yêu cầu thay đổi }
Ngược lại
    Bác bỏ yêu cầu thay đổi

Các quy định chung trong QLCH

- Trừ các thay đổi đơn giản, việc thay đổi phần mềm nên đệ trình lên bộ phận kiểm soát thay đổi để phê duyệt
- Bộ phận kiểm soát thay đổi xem xét tác động của việc thay đổi theo nhiều quan điểm thay vì chỉ dựa theo quan điểm kỹ thuật
- Bộ phận kiểm soát thay đổi bao gồm: một nhóm các thành viên có thể đưa ra các quyết định thay đổi phần mềm
- Quy mô và cấu trúc của bộ phận này có thể thay đổi tùy thuộc vào đề án:
  - ***Với đề án lớn***: có thể bao gồm cả khách hàng lâu năm và hội đồng thầu khoán
  - ***Với đề án nhỏ/vừa***: gồm người quản lý đề án và 1 hay 2 kỹ sư không tham gia trực tiếp vào việc phát triển đề án

Theo dõi sự thay đổi của mỗi thành tố trong một hệ thống phần mềm

- Ghi nhận thông tin lịch sử của mỗi thành tố phần mềm
- Các quy ước về ghi nhận thông tin thay đổi
- Hỗ trợ của công cụ tin học:
  - Các môi trường quản lý cấu hình thường tích hợp hệ thống quản lý phiên bản và hệ thống theo dõi thay đổi phần mềm nhờ vào hệ thống thư điện tử
  - Các môi trường mô phỏng tiến trình hay hệ thống luồng công việc. Tiến trình thay đổi phần mềm được mô phỏng thành các mô hình tiến trình và động cơ thông dịch để trình diễn tiến trình

Khái niệm

- Quản lý phiên bản là tiến trình định nghĩa, theo dõi và kiểm soát các phiên bản khác nhau của một hệ thống phần mềm
- Phiên bản (version): là một thể hiện của phần mềm mà có sự thay đổi so với các thể hiện khác của phần mềm đó
  - Thay đổi có thể bao gồm: chức năng mới, cải tiến hiệu năng, sửa đổi lỗi của phiên bản cũ… Một số phiên bản có thể tương đương hoàn toàn về mặt chức năng nhưng được thiết kế để hoạt động cho các cấu hình phần mềm hay phần cứng khác nhau

Khái niệm (tt)

- Phiên bản phát hành (release): là một phiên bản được phân phối đến khách hàng. Số lượng phiên bản của một hệ thống phần mềm nhiều hơn số lượng phiên bản phát hành hệ thống đó
- Cấu trúc tổng quát của một phiên bản phát hành bao gồm:
  - Các chương trình thực thi hay một tập hợp các chương trình
  - Các tập tin cấu hình để xác định cách thức, nghi thức cài đặt cho từng môi trường cụ thể
  - Các tập tin dữ liệu cần cho sự hoạt động của hệ thống
  - Một chương trình cài đặt dùng để cài đặt hệ thống phần mềm
  - Các tài liệu (*giấy/văn bản điện tử*) liên quan đến hệ thống phần mềm

Khái niệm (tt)

Khi một phiên bản phát hành được sản xuất, tổ chức sản xuất phần mềm cần phải:

- Ghi nhận lại các thông tin về công cụ và môi trường cần thiết để xây dựng phần mềm, chẳng hạn thông tin về hệ điều hành, các trình biên dịch, các công cụ hỗ trợ,…
- Để có thể xây dựng lại phiên bản này, cần tái lập lại chính xác môi trường cấu hình. Trong một số trường hợp, hệ thống quản lý phiên bản có thể chép lưu trữ các phần mềm cơ sở và các công cụ trợ giúp đã dùng để phát triển phiên bản này

Xác định và định danh phiên bản

Có 3 phương pháp chính:

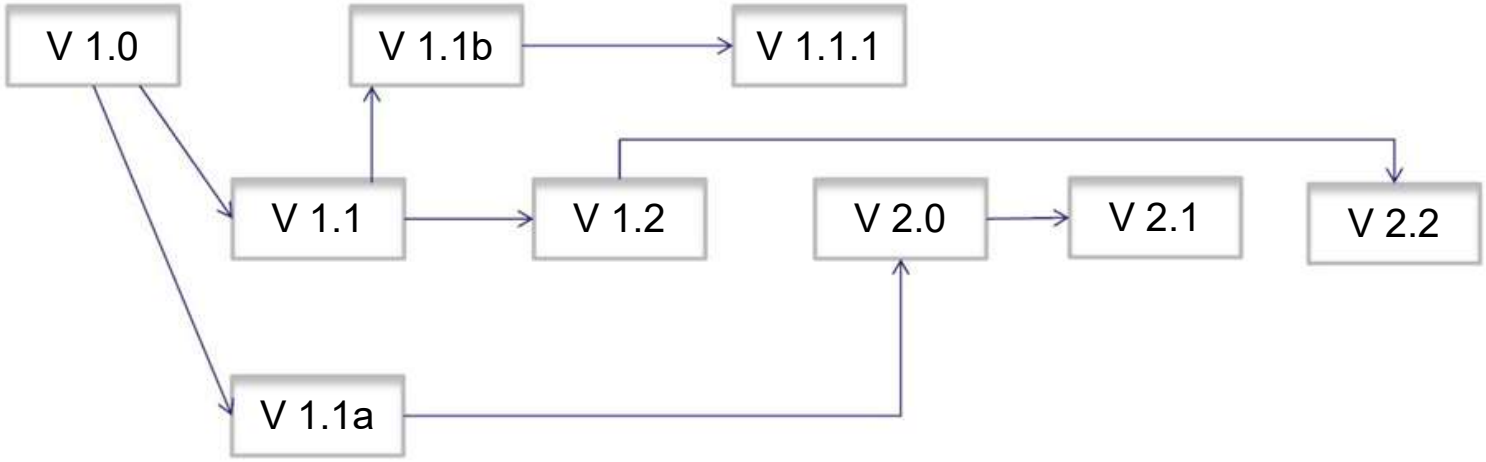
- Sơ đồ tuyến tính
- Sơ đồ định danh theo dạng mạng
- Sơ đồ định danh theo tên

1. Sơ đồ tuyến tính

Sơ đồ này giả sử sự tiến hóa của hệ thống phần mềm diễn ra tuần tự, phiên bản sau tương thích với phiên bản trước:

- Phiên bản đầu tiên là **1.0**
- Các phiên bản cơ sở là **1.0, 2.0, 3.0,…**
- Các phiên bản khác bắt nguồn từ phiên bản cơ sở, như: **2.1, 2.1.1, 3.0.1,…**
- **Khó khăn:**
  - Nếu nhiều phiên bản bắt nguồn từ phiên bản cha thì đánh số phiên bản như thế nào ?
  - Nếu nhiều phiên bản của hệ thống được tạo ra và phân phối đến nhiều khách hàng khác nhau, mỗi khách hàng có một phiên bản duy nhất thì định danh thế nào?

2. Sơ đồ định danh theo dạng mạng



- Phiên bản **1.0** được phát triển theo 2 hướng khác nhau để thành 1.1 và 1.1a. Sau đó, 1.1 được phát triển thành 1.2 và 1.1b
- Phiên bản 2.0 không được phát triển từ 1.2 mà trực tiếp từ 1.1a. Phiên bản 2.2 không được phát triển từ 2.0 mà từ 1.2

3. Sơ đồ định danh theo tên

- Là việc sử dụng tên hoặc ký hiệu để định danh cho phiên bản. Ví dụ: V1/VMS/DBServer → là phiên bản của DB server chạy trên hệ điều hành VMS

Lưu ý chung về việc định danh phiên bản:

Các phương pháp định danh phiên bản đều không phản ánh hết các thuộc tính liên quan đến mỗi phiên bản phần mềm. Các thuộc tính nên được lưu trong CSDL quản lý cấu hình và bao gồm ít nhất là các thông tin sau:

- Khách hàng
- Ngôn ngữ dùng để phát triển
- Tình trạng phát triển
- Cấu hình phần cứng
- Hệ điệu hành
- Ngày tạo phiên bản

Quản lý phiên bản phát hành

Các hoạt động liên quan đến quản lý phiên bản phát hành khi cho ra đời một phiên bản mới:

- Bổ xung thêm mã nguồn
- Xây dựng lại hệ thống
- Chuẩn bị các tập tin cấu hình
- Soạn tài liệu mới
- Đóng gói và phân phối đến khách hàng

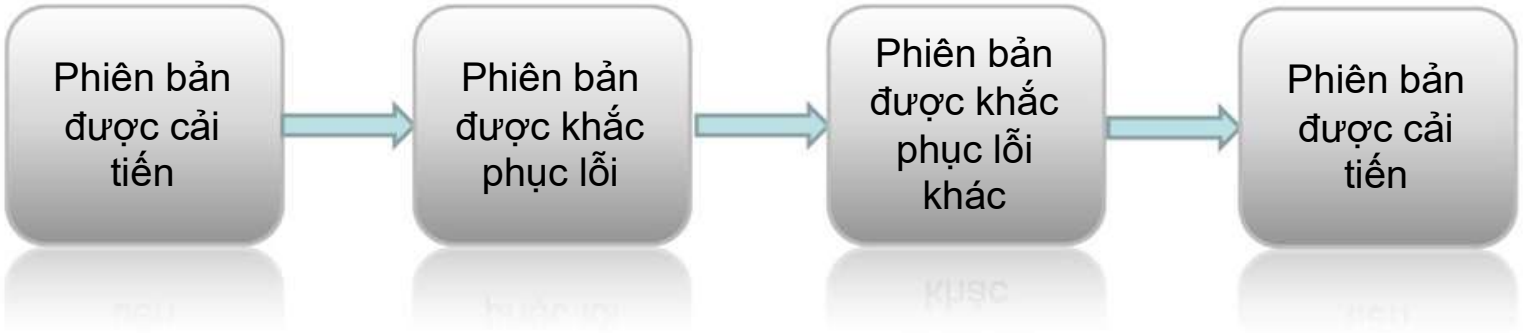
Các thay đổi dẫn đến phát hành phiên bản mới:

- *Để khắc phục lỗi*
- *Để hoàn thiện phần mềm*
- *Để phù hợp với môi trường mới*

Quản lý phiên bản phát hành (tt)

Chú ý:

- *Không nên thực hiện đồng thời nhiều dạng thay đổi cho cùng một phiên bản*
- *Ưu tiên cho việc khắc phục các lỗi trầm trọng*
- *Có thể thực hiện chiến lược luân phiên để cải tiến các phiên bản phát hành*



Công cụ trợ giúp quản lý phiên bản

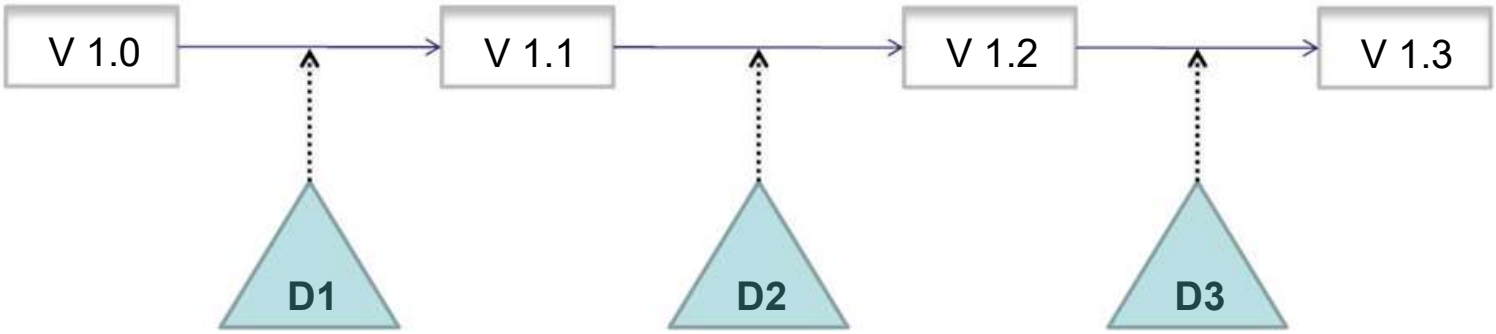
- Một số công cụ thông dụng: *Visual Source Safe* của Microsoft, *ClearCase* của Rational, *CVS* (nguồn mở).
- Các chức năng chính:

1. Định danh phiên bản: có thể định danh tự động và định nghĩa các thuộc tính cho mỗi phiên bản
2. Kiểm soát thay đổi: ghi nhận tường minh các thành tố bị thay đổi, người đã thực hiện sự thay đổi, lưu phiên bản cũ trước đó
3. Quản lý không gian lưu trữ: tiết kiệm kích thước không gian lưu trữ
4. Ghi nhận "vết", lịch sử của việc thay đổi phiên bản

Công cụ trợ giúp quản lý phiên bản (tt)

Để tiết kiệm không gian lưu trữ, một số hệ thống thực hiện như sau:

1. Ghi nhận lại một phiên bản chính
2. Ghi nhận lại sự thay đổi giữa các phiên bản (gọi là Delta, kích thước Delta nên nhỏ hơn nhiều so với kích thước mỗi phiên bản)



Khái niệm

- Là tiến trình tổ hợp các thành tố của hệ thống thành một chương trình có thể thực hiện trên một môi trường có cấu hình cụ thể
- Đối với hệ thống lớn, đây là một tiến trình đóng vai trò quan trọng trong quản lý cấu hình
- Các yếu tố phải xem xét:
  a) Đã xác định đầy đủ các thành tố cần để tích hợp hệ thống hay chưa?
  b) Đã có phiên bản thích hợp của mỗi thành tố hay chưa?
  c) Các tập tin dữ liệu cần thiết đã có sẵn hay chưa?

Khái niệm

d) Vấn đề trùng tên tập tin giữa các tập tin trong hệ thống đang phát triển và các tập tin có sẵn ở môi trường mà hệ thống sẽ hoạt động

e) Đã có sẵn các phiên bản của trình biên dịch và các công cụ cần thiết chưa? Sự tương thích của trình biên dịch và các công cụ với các thành tố của hệ thống đang tích hợp

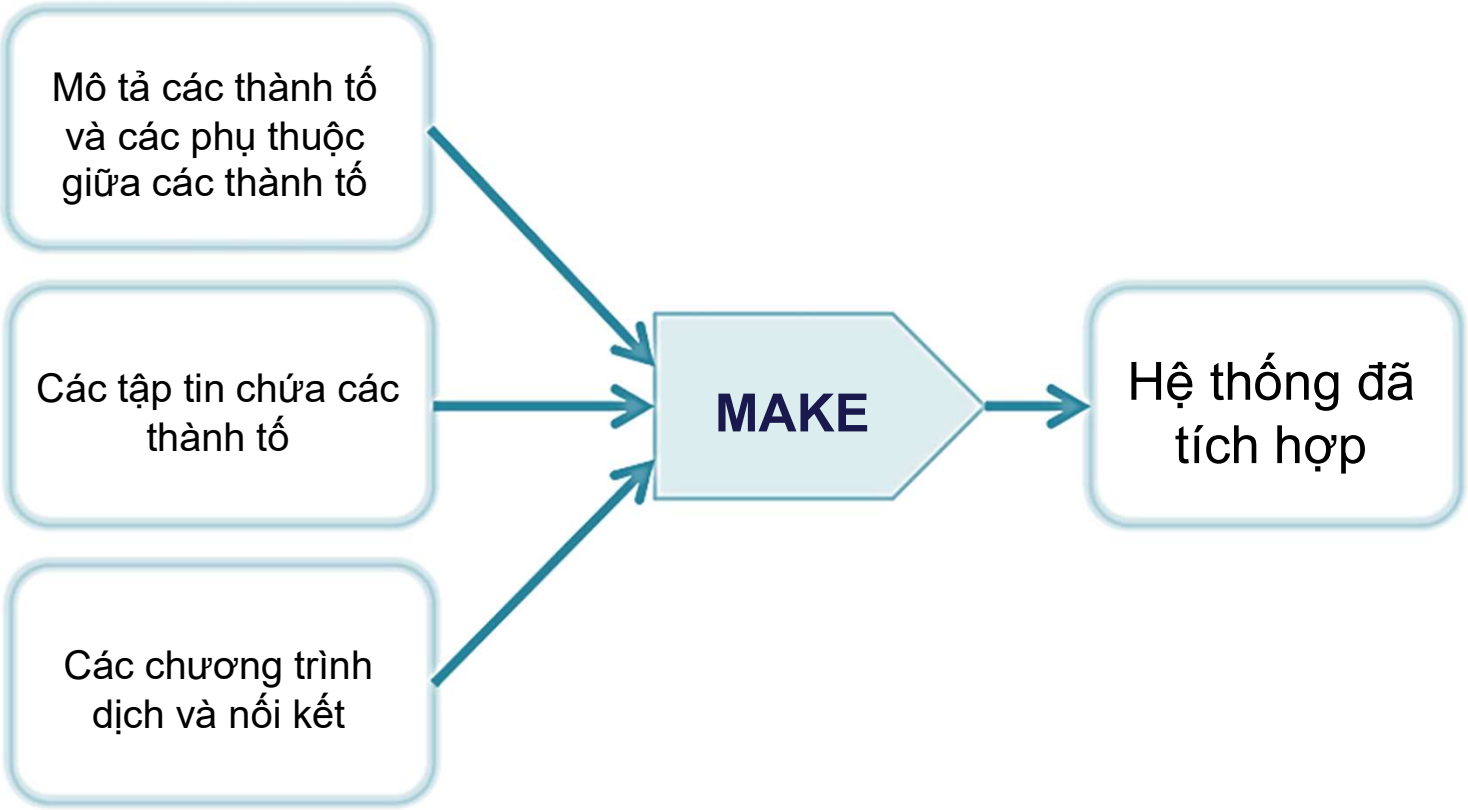
❖ Thời gian dịch và nối kết:

Với các hệ thống lớn, thời gian dịch và nối kết có thể kéo dài hàng giờ, hàng ngày cho mỗi chỉ thị dịch và nối kết. Để cải tiến về thời gian thông thường người ta thực hiện:

d) Ghi nhận các thành tố đã được dịch hoàn tất và biết được các sửa đổi nào không ảnh hưởng đến chúng

e) Cung cấp một ngôn ngữ để mô tả việc tích hợp các thành tố, việc mô tả phải hàm ý được các thành tố không cần xây dựng lại

Ví dụ: lệnh MAKE của UNIX dựa trên mô tả phụ thuộc để không cần dịch lại các thành tố đã biên dịch



Công cụ hỗ trợ

- Các công cụ hỗ trợ việc tích hợp hệ thống:
  - Trên PC có sẵn các môi trường tích hợp hệ thống đang phát triển với các chức năng rất tiện dụng và dễ dàng sử dụng cho người phát triển phần mềm
  - Tuy nhiên chưa giải quyết được việc quản lý phiên bản cho mỗi thành tố và chưa theo dõi được vết của sự thay đổi các thành tố