# 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 ?

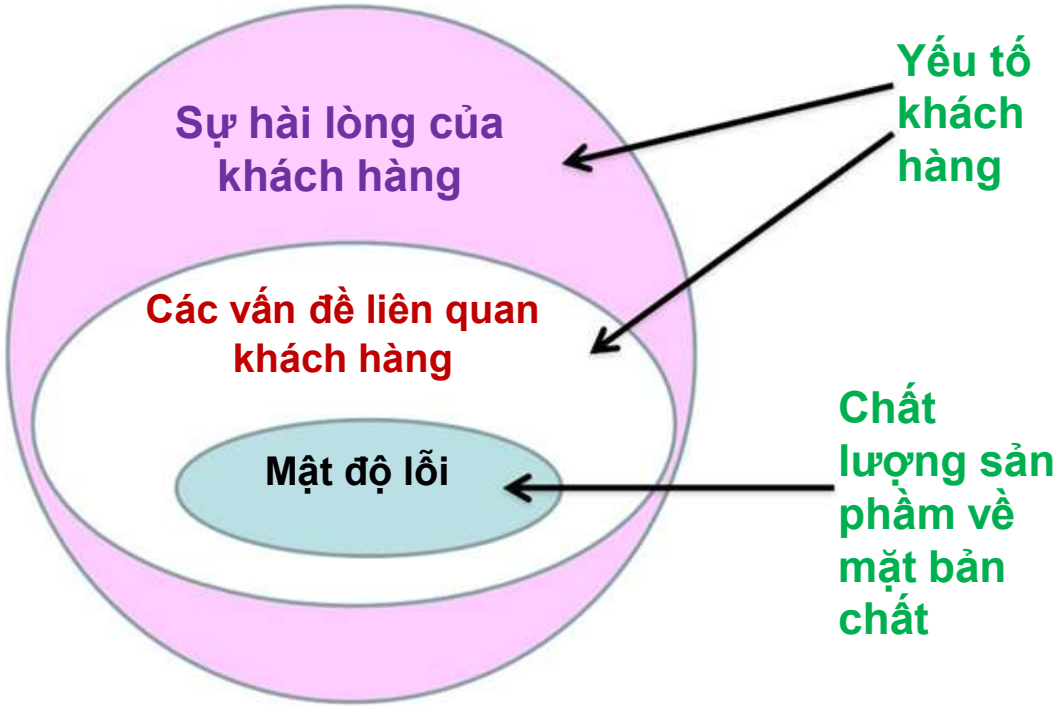
- Chất lượng sản phẩm (rất nhiều góc độ khác nhau):
  - Ít lỗi, dễ bảo trì, tương thích,…
  - Tổ chức đơn thể tốt, có thể tái sử dụng,…
  - Dễ mở rộng, có thể tiến hóa,…
- Độ phức tạp của sản phẩm
- Kích thước (độ lớn) sản phẩm
- ….

## 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

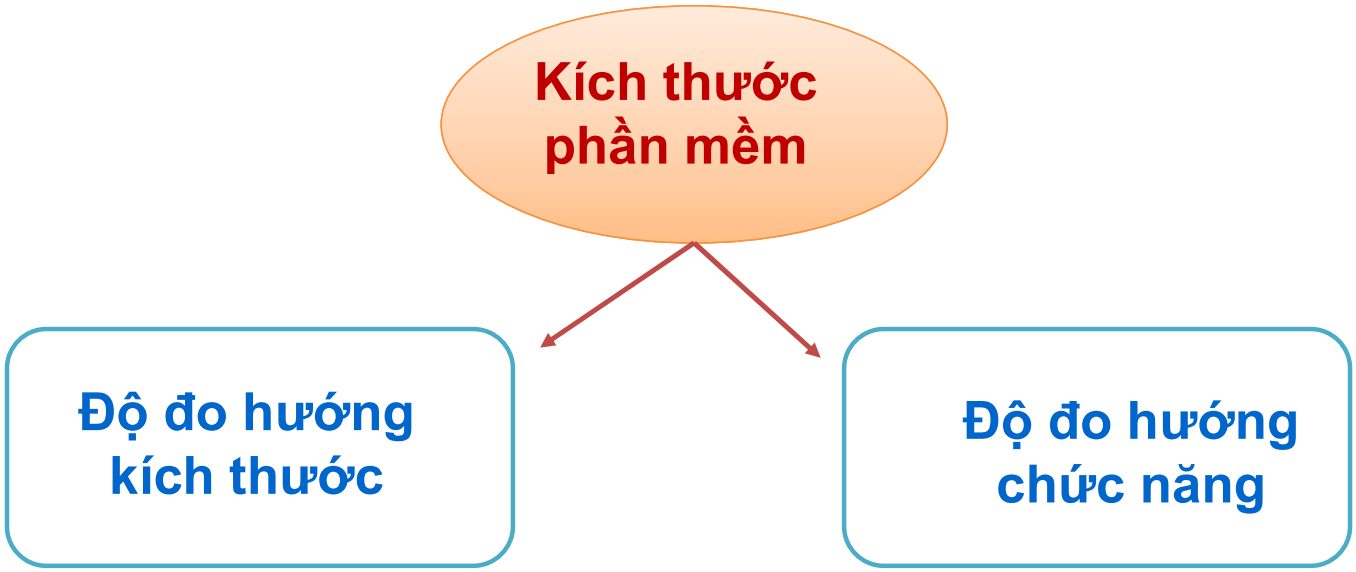
- Cần có đơn vị tính kích thước hay độ lớn của phần mềm để có thể:
  - So sánh giữa các phần mềm với nhau
  - Làm cơ sở để tính năng suất lao động trung bình:

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

  - 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<br>⇔ 5000LOC/7m<br>⇔ 714LOC/m |
| **Ngôn ngữ C** | 3 tuần | 5 tuần | 4 tuần | 6 tuần | 2 tuần | 20 tuần ≈ 5 tháng<br>⇔1500LOC/5m<br>⇔ 300LOC/m |

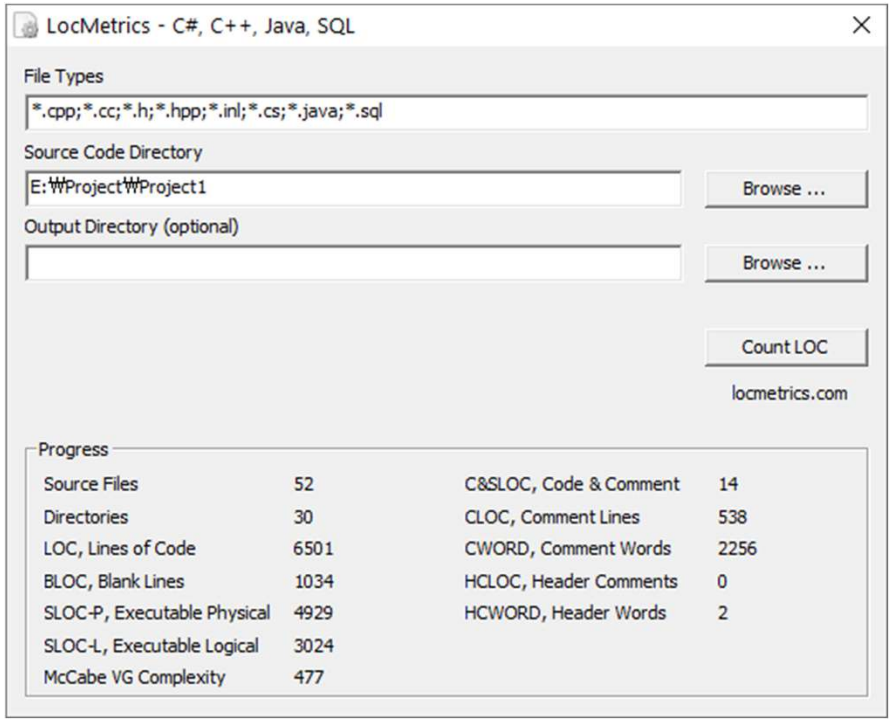
## 3.4.3. Độ đo hướng kích thước (tt)

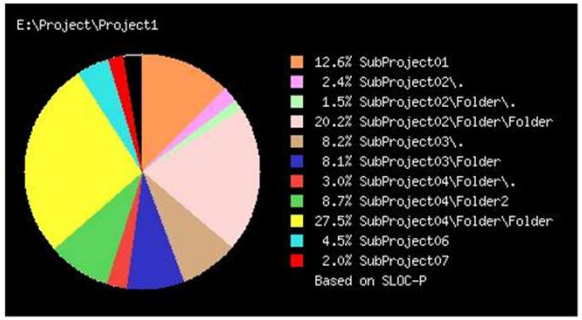
❖ Các vấn đề:

- ✓ 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 $\frac{\Sigma_n^1 t_i}{n}$
  - 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.

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

# HƯỚNG DẪN THỰC HIỆN DỰ ÁN

❖Ví dụ về đặc tả một chức năng

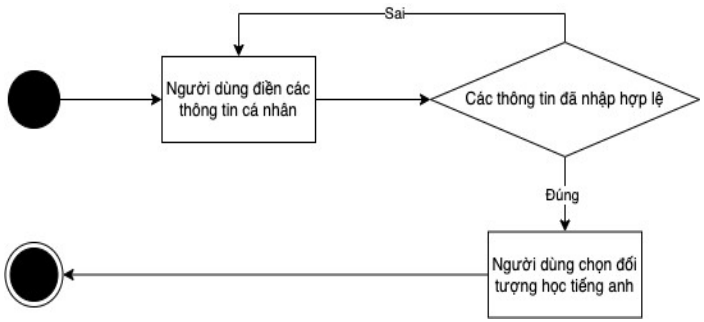

a) Use case

UC1: Đăng ký tài khoản

| | |
|---|---|
| Name | Đăng ký tài khoản |
| Description | Chức năng giúp người dùng đăng ký tài khoản để sử dụng ứng dụng |
| Actor | Người dùng (người lớn và trẻ em) |
| Trigger | Khi người dùng nhấn vào nút "Đăng ký" ở màn hình "Đăng nhập" |
| Pre-condition | Người dùng chưa có tài khoản |
| Post-condition | Tài khoản được tạo<br>Gửi email xác nhận tới người dùng |

b) Mô tả use case

**Activities Flow**




c) "Đăng ký tài khoản" Activities Flow