# Chương 4:
# Quản lý ước lượng

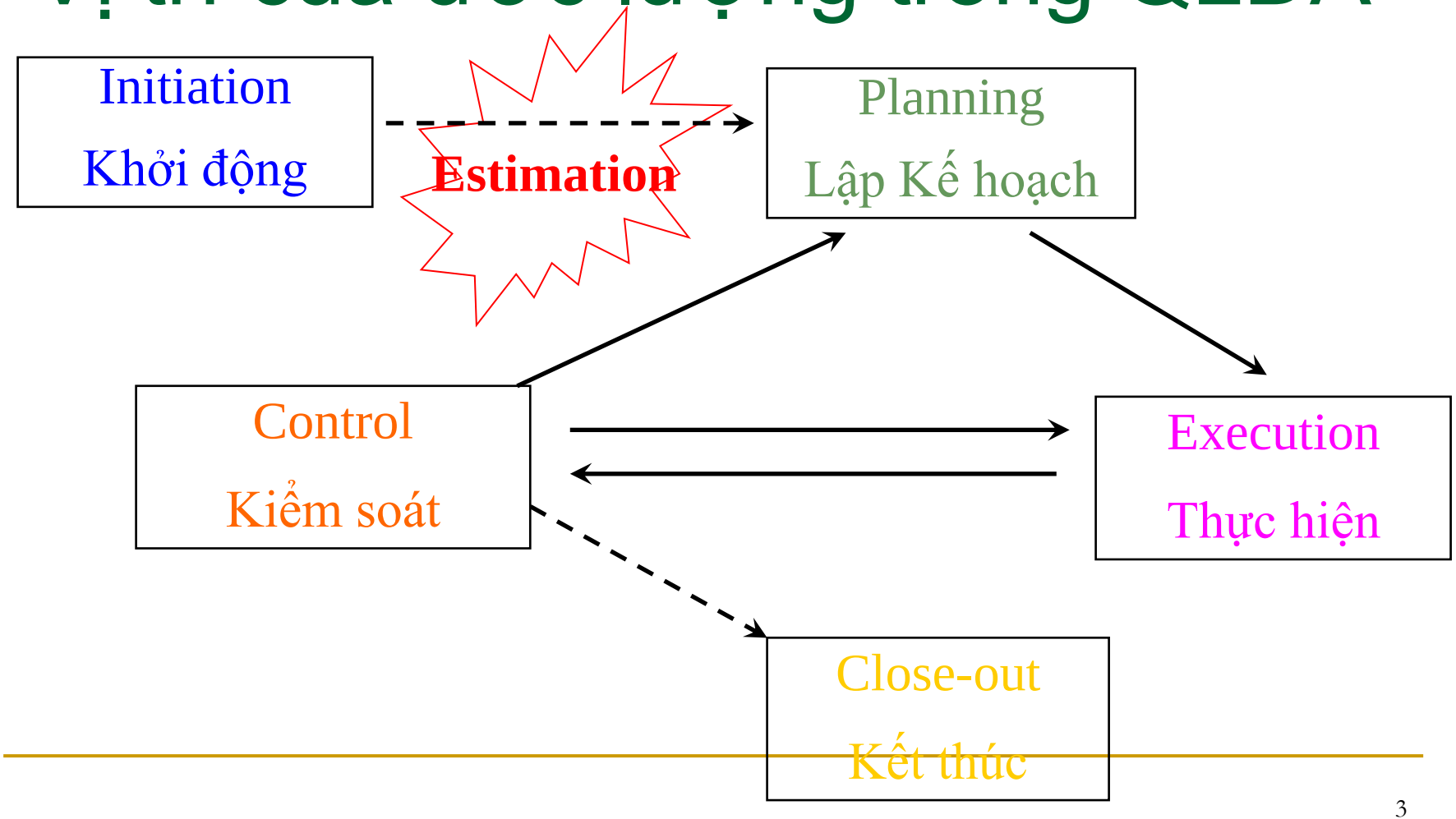
Giảng viên: Nguyễn Văn Hòa

Khoa CNTT - ĐH An Giang

# Nội dung

- Vị trí của ước lượng trong ĐA
- Vai trò của ước lượng
- Cơ sở cho ước lượng phần mềm
- Các phương pháp ước lượng
  - Ước lượng cổ điển
  - Ước lượng công thức
- Ước lượng nổ lực

# Vị trí của ước lượng trong QLDA

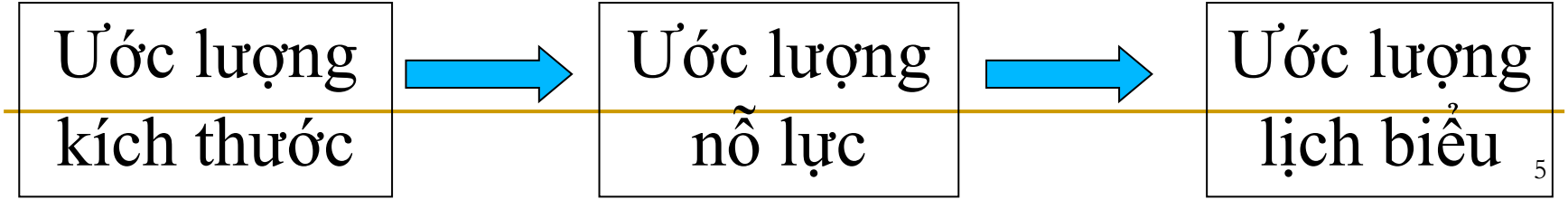


# Tại sao cần ước lượng?

- Xác định thời gian, chi phí, nguồn lực để thực hiện đề án
- Ước lượng là bước bắt buộc trước khi lập kế hoạch dự án
- Cơ sở để ước lượng là WBS

# Các yếu tố cần ước lượng

- Kích thước: Kích cỡ của chương trình, được tính bằng số dòng lệnh, điểm chức năng, điểm Usecase
- Nguồn lực: Nhân lực cần thiết để thực hiện chương trình đó, được tính bằng người/tháng
- Chi phí: Để thực hiện đề án ($/Đ)
- Thời gian: Cần thiết để xây dựng chương trình, được tính bằng giờ, tháng, năm



# Các câu hỏi ước lượng cơ bản

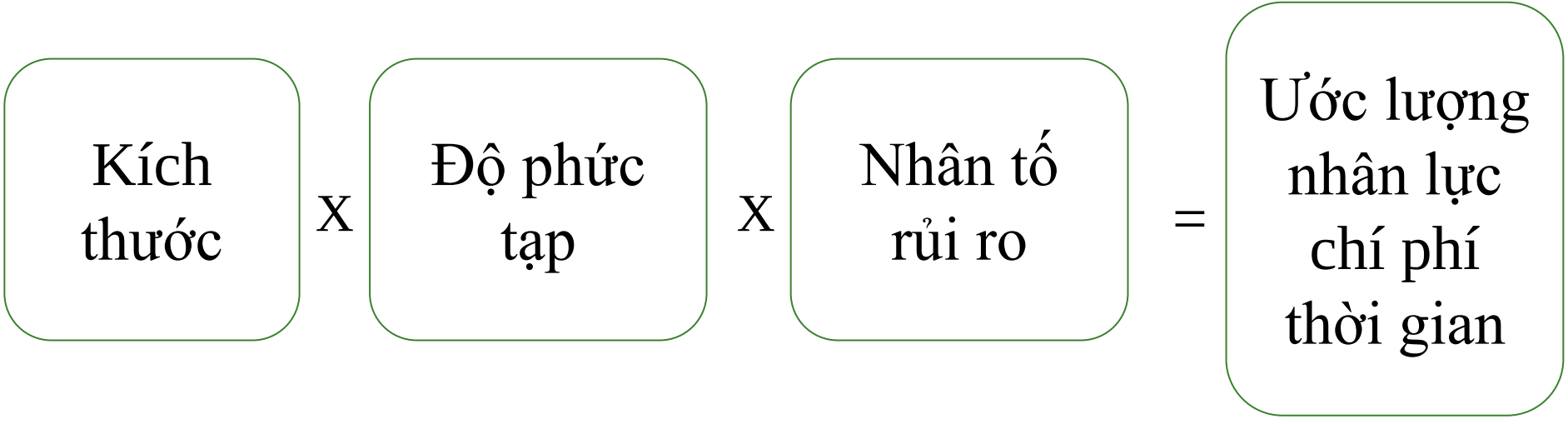
- Cần bao nhiêu nỗ lực để hoàn thành một thao tác/hoạt động?
- Cần bao nhiêu thời gian để hoàn thành một thao tác/hoạt động?
- Tổng chi phí của một thao tác/hoạt động là bao nhiêu?

**Ước lượng và lập kế hoạch dự án là các hoạt động quản trị xen kẻ nhau!**

# Ai là người ước lượng?

- Là người có trách nhiệm thực hiện trong tổ chức
  - Có thể so sánh các dự án trước đây với dự án hiện tại.
  - Thường là người có kinh nghiệm
- Là người bên ngoài tổ chức
  - Có thể cung cấp ước lượng trung thực (unbiased)
  - Có khuynh hướng sử dụng các phương pháp ước lượng theo kinh nghiệm
  - Khó khăn:
    - Thiếu tự tin mô hình sẽ thực hiện tốt hơn một chuyên gia
    - Thiếu các dữ liệu cũ để định kích cỡ cho mô hình mới

# Nguyên tắc ước lượng



# Qui trình ước lượng

# Độ chính xác khi ước lượng

- Ước lượng "chính xác" là điều không thể! (Oxymoron)

**Ước lượng khoảng thời gian từ trường về nhà?**

- Hầu hết các ước lượng phần mềm có sai số từ 25-100%

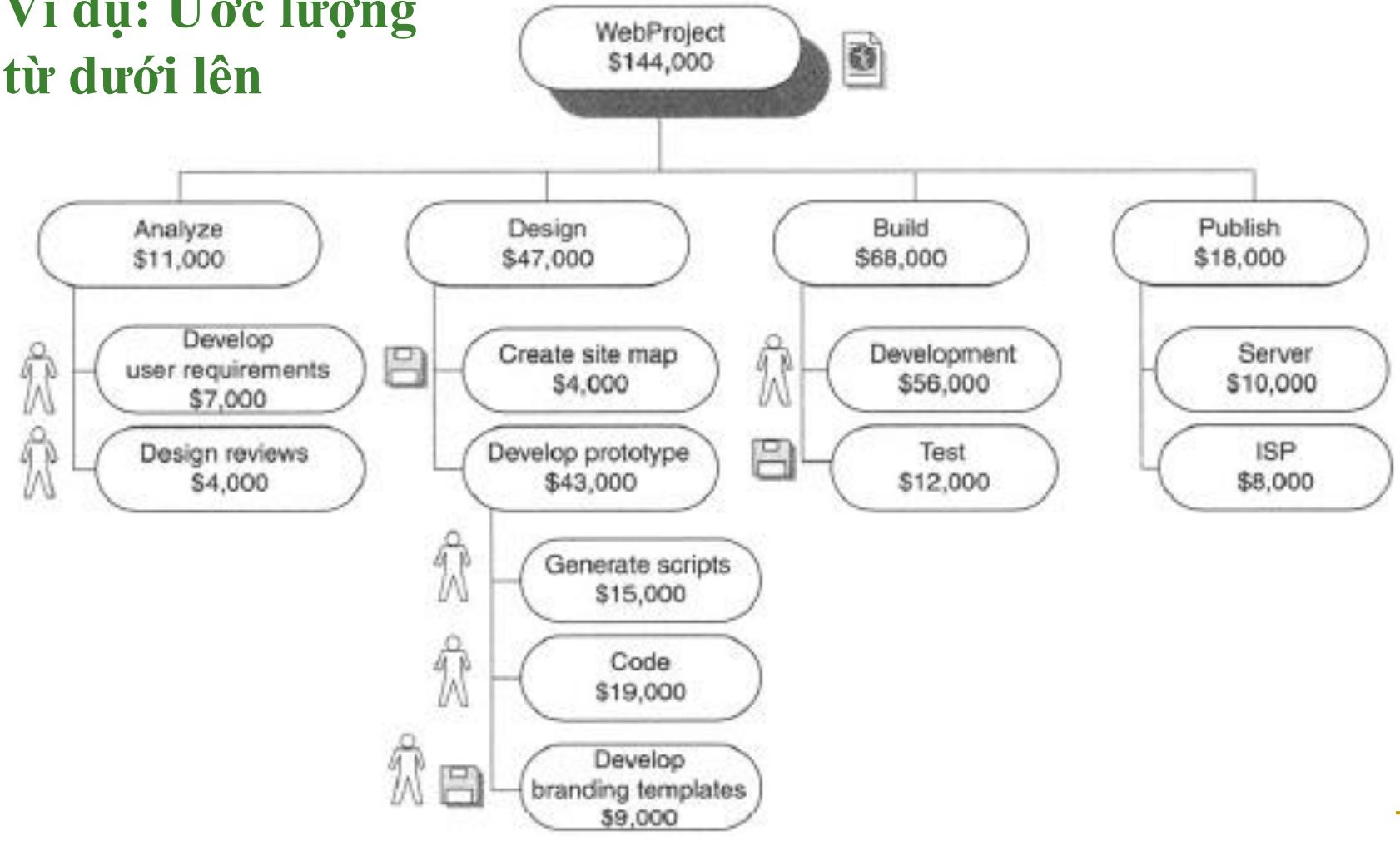
# Các phương pháp ước lượng

- Các phương pháp truyền thống
  - Từ dưới lên (Bottom – up)
  - Từ trên xuống (Top – down)
  - Trả giá để thắng (Price to win)
- Các phương pháp sử dụng công thức
  - Số dòng code
  - Điểm chức năng
  - Điểm người trường hợp người dùng

# PP: Từ dưới lên (Bottom-up)

- Nguyên tắc: Dựa trên kết quả ước lượng từng bộ phận hay tác vụ của đề án, sau đó kết hợp các kết quả để có được kết quả ước lượng cho toàn bộ đề án
- Ưu điểm
  - Dễ dàng ước lượng
  - Độ tin cậy cao
  - Chi tiết và ổn định
- Nhược điểm
  - Rất dễ bỏ sót các yếu tố quan trọng ở cấp độ chi tiết

# Ví dụ: Ước lượng từ dưới lên

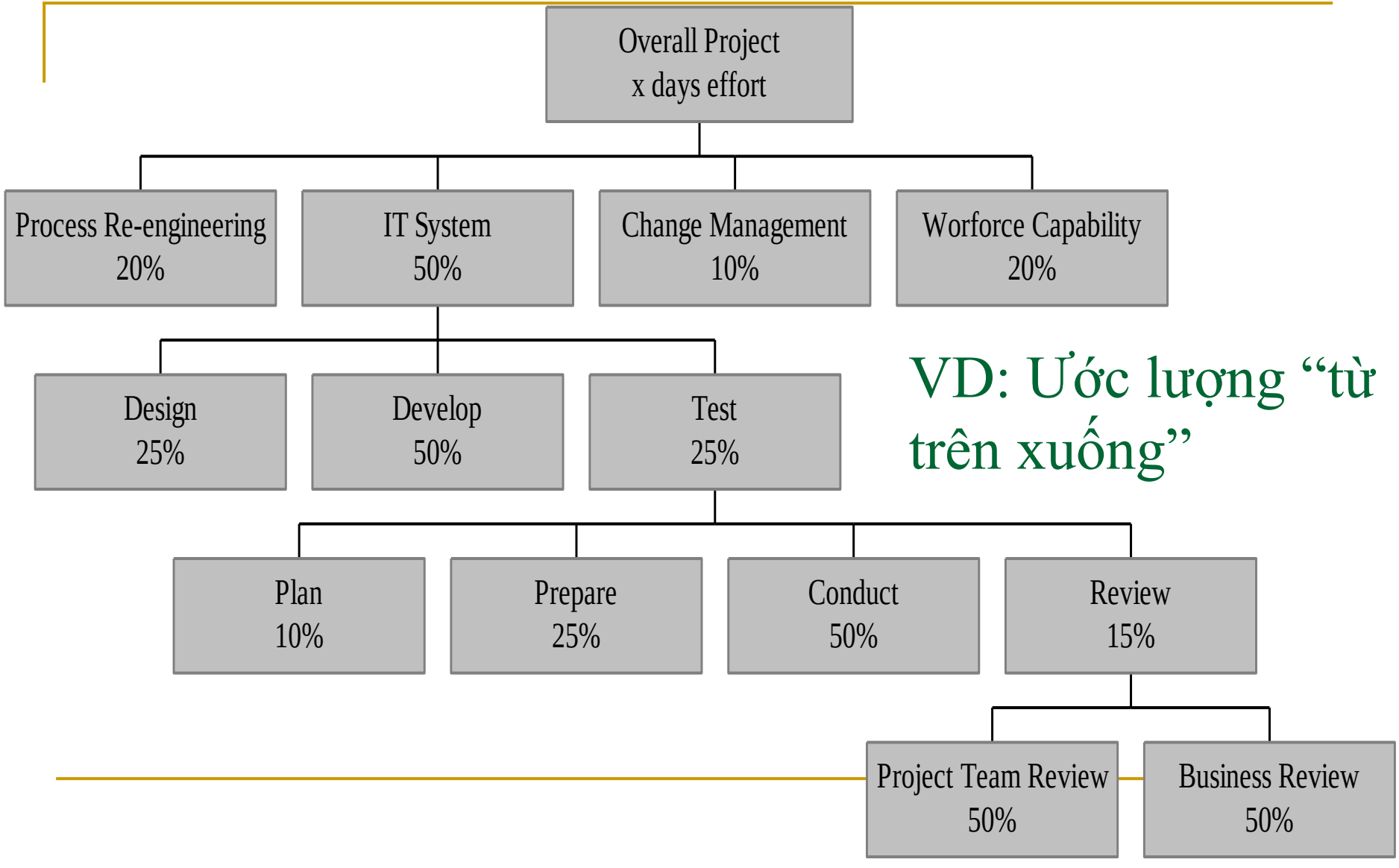


# PP: Từ trên xuống (Top–down)

- Phương pháp ước lượng này bắt đầu từ các đặc trưng tổng quát của đề án phần mềm
- Từ đó đề án được tách ra thành các thành phần khác ở mức thấp hơn
- Phương pháp này thích hợp với các ước lượng ban đầu khi chỉ mới biết được các thuộc tính tổng quát

# PP: Từ trên xuống (Top–down)

- Ưu điểm
  - Xem xét các hoạt động ở mức hệ thống (tích hợp, lập tài liệu, giám sát đề án,….) thường bị bỏ qua trong các PP ước lượng khác
  - Thực hiện nhanh và dễ dàng hơn PP từ dưới lên
  - Không đòi hỏi nhiều sự hiểu biết về hệ thống
- Nhược điểm
  - Độ chính xác ít hơn các PP khác
  - Thường không chú ý tới các thành phần ở mức thấp và các vấn đề kỹ thuật
  - Không có nhiều chi tiết để điều chỉnh các ước lượng




VD: Ước lượng "từ trên xuống"

# PP: Trả giá để thắng

- Giá ước lượng đưa ra nhằm giành được hợp đồng
- Ưu điểm
  - Thường giành được hợp đồng
- Nhược điểm
  - Cạn kiệt tài nguyên trước khi đề án được hoàn thành

# PP: Sử dụng công thức

- Nguyên tắc: Dựa vào công thức toán và các đánh giá đã thực hiện trước đó
- Ưu điểm
  - Hoàn thiện về phương diện công thức
  - Cho phép ước lượng trên với độ tin cậy cao
- Nhược điểm
  - Đánh giá chủ quan các tham biến đầu vào
  - Không thể xét đến các điều kiện  ngoại lệ

# Ước lượng sử dụng công thức

- Số dòng lệnh (Lines of Code-LOC)
- Điểm chức năng (Function points-FP)
- Điểm trường hợp sử dụng (Use case points)
- UCP và FP được dùng nhiều nhất

# Ước lượng dựa trên LOC

- Ưu điểm
  - Dễ hiểu
  - Cho phép so sánh chi tiết
  - Dễ ước lượng
- Nhược điểm
  - Khó ước lượng ở giai đoạn đầu
  - Số dòng lệnh sẽ phụ thuộc vào ngôn ngữ lập trình
  - Nhiều chi phí không được xem xét (Phân tích yêu cầu)
  - Bộ sinh mã tạo ra nhiều dòng lệnh không cần thiết

# Ước lượng Function Points

- Function points là một cách đo kích cỡ chương trình dựa vào số dòng mã và sự phức tạp của đầu vào (inputs), đầu ra (outputs), các câu hỏi truy vấn (queries), files và giao diện chương trình (program interface)
- Khoa học hơn dùng LOC. Ví dụ: Trong một căn nhà
  - Diện tích ~ = LOC
  - Số phòng ngủ và nhà tắm ~= Function points
  - LOC chỉ chú ý đến kích thước, FP chú ý cả kích thước và chức năng

# Ước lượng Function Points tt.

- Mỗi input, output hoặc query là các thành phần chủ yếu của hệ thống, chẳng hạn như: screen, report hay database search
- Sự phức tạp của mỗi thành phần này được đánh giá bởi người QLDA
- Sự phức tạp của hệ thống, chẳng hạn như tính thân thiện của đội ngũ thực hiện DA với môi trường kinh doanh và công nghệ sẽ được thực hiện trong dự án
- DA có thể ít phức tạp đối với đội ngũ có nhiều kinh nghiệm và phức tạp hơn với đội ngũ ít kinh nghiệm

# Ước lượng Function Points tt.

- Số dòng lệnh và sự phức tạp của mỗi thành phần được ước tính và ghi nhận trong một bảng tính. Sau đó dùng các giá trị này để tính TUFP (Total Unadjusted Function Points)
- Tất cả 14 yếu tố ảnh hưởng đến độ phức tạp của DA như: hiệu quả người sử dụng, tính tái sử dụng, liên kết dữ liệu... sẽ đánh giá độ phức tạp của DA

# Ước lượng Function Points tt.

- PC (Project complexity) bằng tổng các yếu tố này
- PCA = Y + 0.01 x PC

  (PCA = Adjusted Project complexity)

  Trong đó:

  0.65<=Y<=1.0 cho DA đơn giản

  Y=1.35 cho DA phức tạp cao
- TAFP = PCA x TUFP

  (TAFP = Total adjusted function points)

# Ước lượng Function Points tt.

| | Mức độ phức tạp | | | |
|---|---|---|---|---|
| Mô tả | Thấp | Tr. bình | Cao | T.cộng |
| Input | ----x 3 | ----x 4 | ----x 6 | ------ |
| Output | ----x 4 | ----x 5 | ----x 7 | ------ |
| Queries | ----x 3 | ----x 4 | ----x 6 | ------ |
| Files | ----x 7 | ----x 10 | ----x 15 | ------ |
| Program Interfaces | ----x 3 | ----x 7 | ----x 10 | ------ |

Total Unadjusted Function Points (TUFP): ----------

# Ước lượng Function Points tt.

| | 0-5 |
|---|---|
| Kết nối dữ liệu | |
| Cấu hình khó | |
| Tỷ lệ giao dịch | |
| Hiệu quả người dùng cuối | |
| Xử lý phức tạp | |
| Dễ cài đặt | |
| ..... ..... | |

1. TUFP= ?
2. PC= ?
3. PCA=0.65 +0.01xPC
3. TAFP = PCAx TUFP

**PC** (Project complexity).....

(0: Không có phức tạp; 5: Độ phức tạp cao nhất)

# Ước lượng Function Points tt.

| Ngôn ngữ LT | Số dòng cho mỗi FP |
|---|---|
| C | 130 |
| COBOL | 110 |
| Java | 55 |
| C++ | 50 |
| Turbo Pascal | 50 |
| Visual Basic | 30 |
| PowerBuilder | 15 |
| …….. | …….. |

Khi đã có tổng số Function Points (TAFP), cần phải chuyển nó thành số dòng mã cần thực hiện. Sự qui đổi này dựa vào NNLT được sử dụng.

Ví dụ: TAFP =10

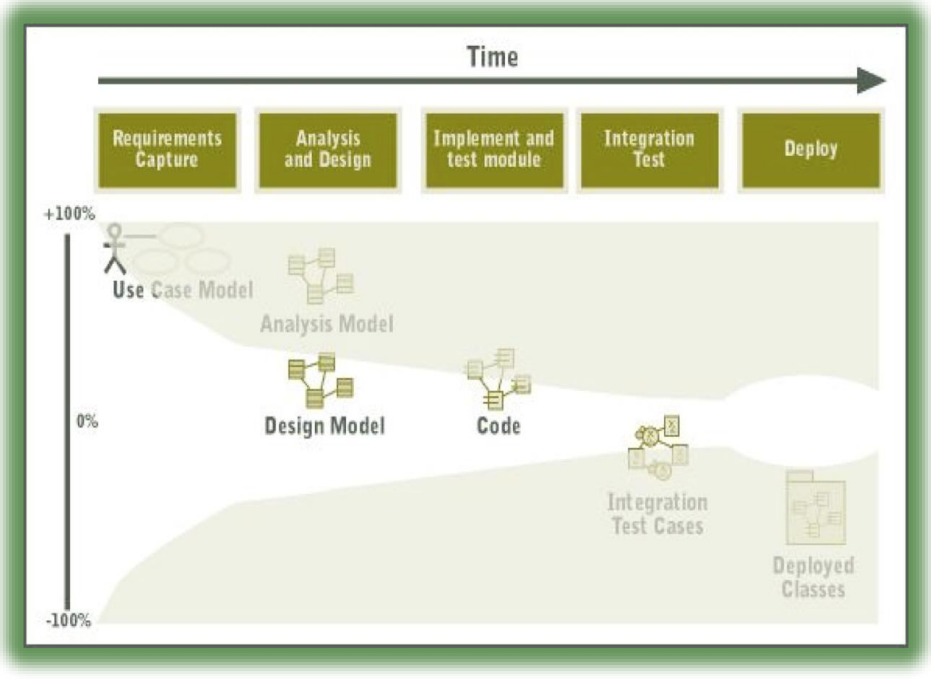
C :1300 dòng

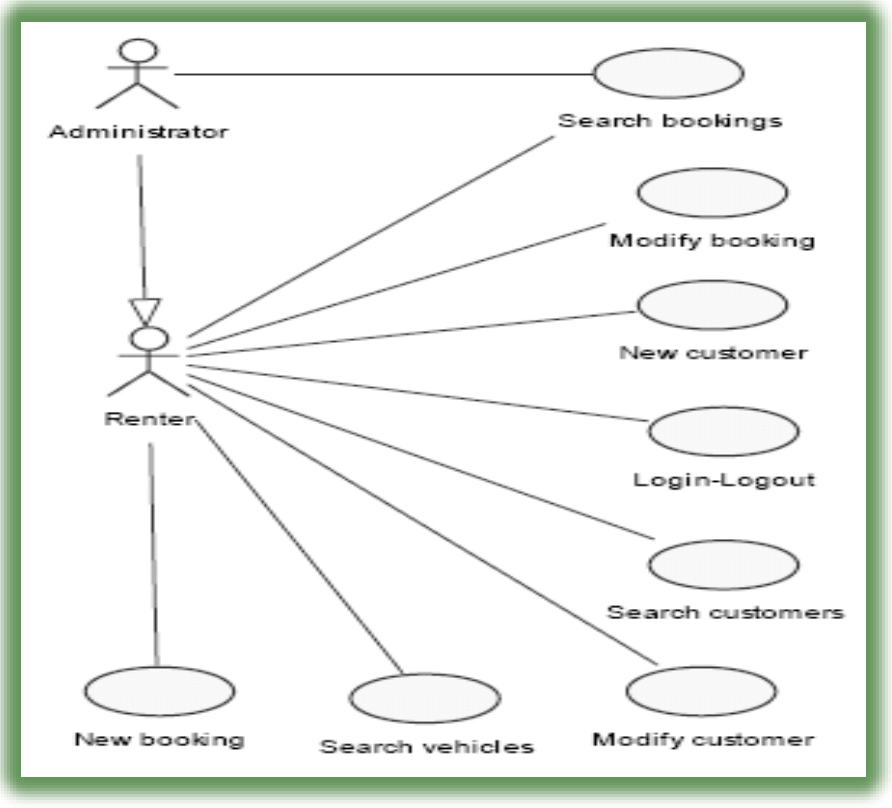
C++ : 500 dòng

Java : 550 dòng

.....

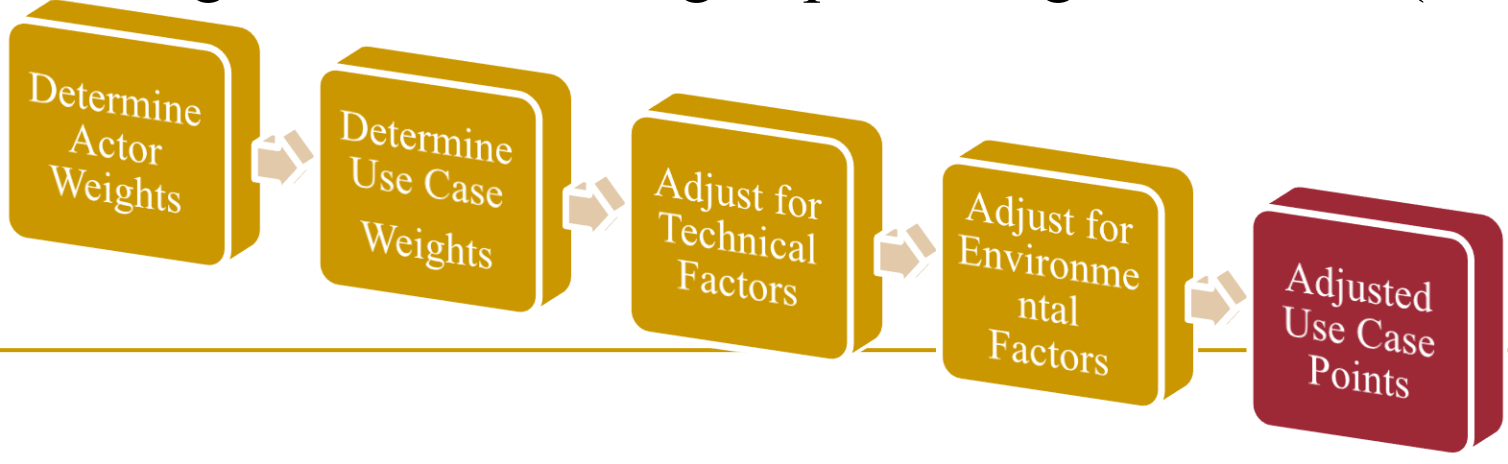
# Ước lượng Use Case Point



# Ước lượng Use Case Point tt.

- Quy trình ước lượng điểm trường hợp sử dụng
  - Tính giá trị điểm tác nhân (UAW)
  - Tính giá trị điểm trường hợp sử dụng (UUCW)
  - Tính yếu tố phức tạp kỹ thuật (TCF)
  - Tính yếu tố phức tạp môi trường (ECF)
  - Tính giá trị điểm trường hợp sử dụng điều chỉnh (UCP)



# Tính giá trị điểm tác nhân

| Step | Rule | Output |
|---|---|---|
| 1 | Classify actors:<br><br>a) Simple, WF (Weight Factor) = 1<br>b) Average, WF = 2<br>c) Complex, WF = 3 | Unadjusted Actor Weights (UAW) = ∑(#Actors * WF) |
| 2 | Classify use cases:<br><br>a) Simple- 3 or fewer transactions, WF = 5<br>b) Average- 4 to 7 transactions, WF = 10<br>c) Complex- more than 7 transactions, WF= 15† | Unadjusted Use Case Weights (UUCW) = ∑ (#Use Cases * WF) |
| 3 | Calculate the Unadjusted Use Case Point (UUCP). | UUCP = UAW + UUCW |

# Tính giá trị điểm tác nhân tt.

- Yếu tố kỹ thuật

| Factor | Factor description | Weight |
|---|---|---|
| T1 | Distributed system | 2 |
| T2 | Response or throughput performance objective | 1 |
| T3 | End-user efficiency | 1 |
| T4 | Complex internal processing | 1 |
| T5 | Code must be reusable | 1 |
| T6 | Easy to install | 0.5 |
| T7 | Easy to use | 0.5 |
| T8 | Portable | 2 |
| T9 | Easy to change | 1 |
| T10 | Concurrent | 1 |
| T11 | Includes special security features | 1 |

# Tính giá trị điểm tác nhân tt.

- Yếu tố môi trường

| Factor | Factor description | Weight |
|---|---|---|
| F1 | Familiar with RUP | 1.5 |
| F2 | Application experience | 0.5 |
| F3 | Object-oriented experience | 1 |
| F4 | Lead analyst capability | 0.5 |
| F5 | Motivation | 1 |
| F6 | Stable requirements | 2 |
| F7 | Part-time workers | -1 |
| F8 | Difficult programming language | -1 |

# Tính giá trị điểm tác nhân tt.

| | | |
|---|---|---|
| 4 | Assign values to the technical and environmental factors [0..5], multiply by their weights [-1..2], and calculate the weighted sums (TFactor and EFactor). Calculate TCF and EF as shown. | Technical Complexity Factor (TCF) = 0.6 + (0.01 * TFactor)<br><br>Environmental Factor (EF) = 1.4 + (-0.03 * EFactor) |
| 5 | Calculate the adjusted Use Case Points (UCP). | UCP = UUCP * TCF * EF |

# Ví dụ tính điểm trường hợp SD

- Xét dự án shop bán hàng có các Usecase sau:
  - UAW= No simple actors * 1 + No average actors * 2
    + No Complex actors * 3

    UAW = 1 * 1 + 0 * 2 + 4 *3 = 13
  - UUCW = No simple UC * 5 + No average UC * 10
    + No Complex actors * 15

    UUCW = 2 * 5 + 3 *10 + 4 * 15 = 100

# Ví dụ tính điểm trường hợp SD tt.

- Tính yếu tố phức tạp kỹ thuật
  - TCF = 0.6 + (TF*0.01) $\Rightarrow$ TCF = 0.6 + (42*0.01) =1.02

| Factor | Description | Weight | Assigned Value | Weight x Assigned Value |
|---|---|---|---|---|
| T1 | Distributed system | 2.0 | 5 | 10 |
| T2 | Response time/performance objectives | 1.0 | 5 | 5 |
| T3 | End-user efficiency | 1.0 | 3 | 3 |
| T4 | Internal processing complexity | 1.0 | 2 | 2 |
| T5 | Code reusability | 1.0 | 3 | 3 |
| T6 | Easy to install | 0.5 | 1 | 0.5 |
| T7 | Easy to use | 0.5 | 5 | 2.5 |
| T8 | Portability to other platforms | 2.0 | 2 | 4 |
| T9 | System maintenance | 1.0 | 2 | 2 |
| T10 | Concurrent/parallel processing | 1.0 | 3 | 3 |
| T11 | Security features | 1.0 | 5 | 5 |
| T12 | Access for third parties | 1.0 | 1 | 1 |
| T13 | End user training | 1.0 | 1 | 1 |
| | | | **Total (TF):** | **42** |

# Ví dụ tính điểm trường hợp SD tt.

- Tính yếu tố phức tạp môi trường
  - ECF =1.4+(-0.03*EF) $\Rightarrow$ 1.4+(-0.03*10.5) = 1.085

| Factor | Description | Weight | Assigned Value | Weight x Assigned Value |
|---|---|---|---|---|
| E1 | Familiarity with development process used | 1.5 | 3 | 4.5 |
| E2 | Application experience | 0.5 | 3 | 1.5 |
| E3 | Object-oriented experience of team | 1.0 | 2 | 2 |
| E4 | Lead analyst capability | 0.5 | 5 | 2.5 |
| E5 | Motivation of the team | 1.0 | 2 | 2 |
| E6 | Stability of requirements | 2.0 | 1 | 2 |
| E7 | Part-time staff | -1.0 | 0 | 0 |
| E8 | Difficult programming language | -1.0 | 4 | -4 |
| | | | **Total (EF):** | **10.5** |

# Ví dụ tính điểm trường hợp SD tt.

- Tính giá trị điểm trường hợp sử dụng
  - UCP = (UUCW + UAW) x TCF x ECF

    UCP = (100 + 13) x 1.02 x 1.085 = 125.06
  - Vậy tổng điểm trường hợp sử dụng của dự án là 125.06

# Ước lượng nỗ lực

- Effort được tính bởi sự kết hợp giữa size và production rates
- Production rates (PR) thể hiện bao nhiêu công việc mà một người có thể hoàn thành trong một thời gian xác định
- Giải thuật COCOMO (Constructive Cost Model), thiết kế bởi Barry W Boehm, đã chuyển việc ước lượng số dòng mã sang việc ước lượng person-months

# Ước lượng nỗ lực (tt)

- Đối với các phần mềm kinh tế mức độ vừa phải (khoảng 100.000 dòng mã và khoảng 10 lập trình viên) thì effort được tính như sau

  effort (in person-months) = 1,4 x ngàn dòng

- Trong trường hợp, số dòng của DA khoảng 10.000 dòng thì cần 14 person-months để hoàn thành

  effort = 1,4 x 10 = 14

- Nếu DA có độ phức tạp cao hơn, kích cỡ lớn hơn... thì effort này sẽ thay đổi

# Ước lượng lịch biểu thời gian

- Sau khi đã tính được effort, sự tối ưu lịch biểu (optimal schedule) cho DA có thể được ước lượng theo công thức:

  $$\text{Schedule time(months)} = 3{,}0 \times \text{person-months}^{1/3}$$

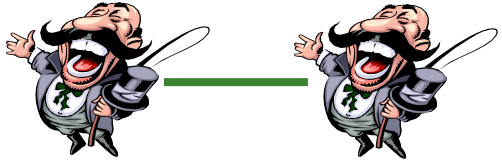
- Ví dụ: DA có 40 person-months thì thời gian hoàn thành DA ít nhất là:

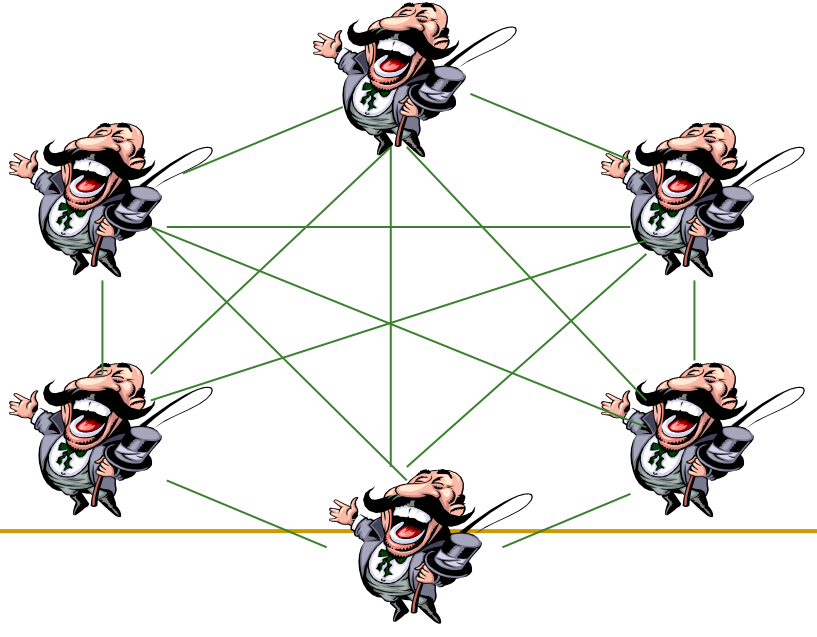
  $$\text{Schedule time} = 3{,}0 \times 40^{1/3} = 3{,}0 \times 3{,}4 \approx 10 \text{ tháng.}$$

- Do đó, để hoàn thành một DA 40 person-moths trong 10 tháng thì một nhóm 4 người làm việc full-time

# Ước lượng lịch biểu thời gian (tt)

- Để giảm thời gian thực hiện DA có thể tăng số người thực hiện nhưng việc làm này là không hiệu quả. Bởi vì, số người tham gia đông sẽ khó quản lý và khó kết hợp.





# Chọn và sử dụng phương pháp UL

- Mỗi phương pháp đều có ưu điểm và nhược điểm riêng
- Việc ước lượng nên dựa trên nhiều phương pháp khác nhau. Nếu các phương pháp này không mang lại kết quả gần giống nhau thì có nghĩa là thông tin cung cấp đã không đầy đủ
- Giám sát dự án để phát hiện khi nào các ước lượng bị sai và có ảnh hưởng nhiều đến độ chính xác của việc ước lượng
- Nên lưu trữ CSDL của các dự án trước đây

# Các lưu ý khi ước lượng

- Ước lượng phải được thực hiện bởi người quen thuộc nhất với công việc
- Nếu có thể thì lấy kết quả ước lượng từ nhiều người rồi sử dụng sự khác nhau cho việc đánh giá rủi ro
- Các ước lượng nên được thực hiện độc lập để tránh "GroupThink"
- Các ước lượng phải dựa trên điều kiện thông thường

# Các lưu ý khi ước lượng

- Dùng các đơn vị nhất quán khi ước lượng thời gian
- Ước lượng cho các gói công việc không nên bao gồm yếu tố ngẫu nhiên
- Sử dụng đánh giá rủi ro cho việc ước lượng tác động của các điều kiện bất thường và các yếu tố ngẫu nhiên.