NHẬP MÔN CÔNG NGHỆ PHẦN MỀM

PHẦN III – ĐÁNH GIÁ, ƯỚC LƯỢNG CHI PHÍ

Bộ môn Công nghệ phần mềm,
Khoa CNTT&TT, Đại học Cần Thơ

Nội dung

- Các loại đánh giá
- Ước lượng chi phí
- Xác định giá trị phần mềm theo công văn 2589/BTTTT

Các loại đánh giá

- Đánh giá phân tích yêu cầu
- Đánh giá thiết kế kiến trúc
- Đánh giá thiết kế chi tiết
- Đánh giá cài đặt
- Đánh giá kiểm thử
- Đánh giá mức độ tin cậy
- Đánh giá mức độ sẵn sàng
- Đánh giá bảo trì

Đánh giá phân tích yêu cầu

- Số lượng yêu cầu n_r trong một đặc tả

  $n_r = n_f + n_{nf}$

  - n_f: số lượng yêu cầu chức năng
  - n_{nf}: số lượng yêu cầu phi chức năng
- Độ cô đọng (specificity) hay súc tích của một đặc tả (ít nhầm lẫn)

$$Q_1 = \frac{n_{ui}}{n_r}$$

  - n_{ui} là số lượng các yêu cầu được thẩm định là tốt (rõ ràng, không mơ hồ)

Đánh giá phân tích yêu cầu

- Độ hoàn chỉnh (completeness) của yêu cầu chức năng $$Q_2 = \frac{n_u}{(n_i \times n_s)}$$
  - n_u : số lượng các yêu cầu chức năng duy nhất
  - n_i : số lượng đầu vào trong đặc tả
  - n_s : số lượng các kịch bản hay trạng thái trong đặc tả
- Mức độ hợp lệ của các yêu cầu $$Q_3 = \frac{n_c}{(n_c + n_{nv})}$$
  - n_c: số lượng yêu cầu được thẩm định là đúng
  - n_{nv} : số lượng yêu cầu chưa được thẩm định

Đánh giá thiết kế kiến trúc

Đối với kiến trúc phân cấp

- Độ phức tạp cấu trúc $$S(i) = f_{out}^2(i)$$
  - $f_{out}(i)$ =fan-out(i) là số lượng module được gọi bởi module i
- Độ phức tạp dữ liệu xác định mức độ phức tạp bên trong của module i $$D(i) = \frac{v(i)}{[f_{out}(i)+1]}$$
  - v(i) : số lượng biến đầu vào và đầu ra.
- Độ phức tạp kiến trúc của hệ thống $$C(i) = S(i) + D(i)$$

Đánh giá thiết kế kiến trúc

- Sự độc lập của module $$D_2 = 1 - \frac{S_2}{S_1}$$
  - S1: tổng số các module định nghĩa trong kiến trúc chương trình
  - S2 : số lượng các module mà sự chính xác trong hoạt động phụ thuộc vào dữ liệu đầu vào hoặc xuất ra các dữ liệu được sử dụng ở chỗ khác (không tính các module điều khiển)

Đánh giá thiết kế chi tiết

- Mức độ nối kết (coupling) của một phân hệ với các phân hệ khác với dữ liệu toàn cục và với môi trường bên ngoài. $$m_c = \frac{k}{M}$$
  - k : hằng số tỷ lệ
  - $M = d_i + (a \times c_i) + d_0 + (b \times c_0) + g_d + (c \times g_c) + w + r$
    - d_i : số lượng tham số dữ liệu đầu vào
    - c_i : số lượng tham số điều khiển đầu vào
    - d_0 : số lượng tham số dữ liệu đầu ra
    - c_0 : số lượng tham số điều khiển đầu ra
    - g_d : số lượng các biến toàn cục sử dụng như dữ liệu
    - g_c : số lượng các biến toàn cục sử dụng như điều khiển
    - w : số lượng các phân hệ đã gọi (fan-out)
    - r : số lượng các phân hệ gọi đến (fan-in)
    - Các giá trị k, a, b và c phải được xác định bằng thực nghiệm

Đánh giá cài đặt

- Độ dài mã nguồn

$$N = n_1 \log_2 n_1 + n_2 \log_2 n_2$$

  - n_1 : số lượng các toán tử khác nhau
  - n_2 : số lượng các toán hạng khác nhau
  - N_1 : số lượng các toán tử
  - N_2 : số lượng các toán hạng
- Số lượng lỗi dự đoán (/KDSI)

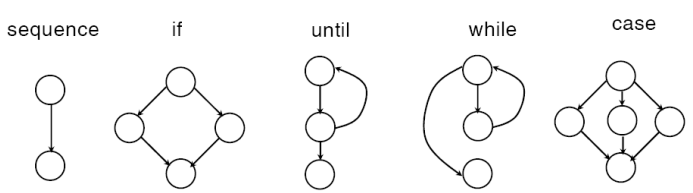
$$B = \frac{(N_1 + N_2)\log_2(n_1 + n_2)}{3000}$$

Đánh giá cài đặt

- Phép đo độ phức tạp của McCabe
  - Một chương trình được xem như một đồ thị có hướng với nút là dòng lệnh và cung là dòng điều khiển giữa các lệnh.
  - *Độ phức tạp của McCabe* được tính theo công thức **v(F) = e – n + 2** với:
    - e: Tổng số cạnh hay cung
    - n: Tổng số nút
  - Giá trị này có thể giúp bảo trì viên:
    - Cân nhắc xem chương trình có cần được hiệu chỉnh để làm giảm độ phức tạp hay không.
    - Ước lượng thời gian cần để hiểu và hiệu chỉnh chương trình.

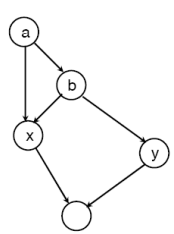
Đo độ phức tạp

- Thiết lập đồ thị có hướng
  - Mỗi nút hình tròn biểu diễn một hoặc một vài tác vụ.
  - Cạnh có hướng miêu tả đường thực thi.
  - Đồ thị dòng chảy được xây dựng từ lưu đồ thuật giải.

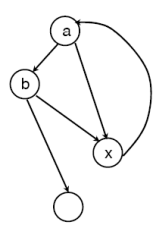


Đo độ phức tạp

- Thiết lập đồ thị
  - Phải phân rã tất cả các điều kiện phức trở thành các điều kiện đơn.



`if a and b then y else x`



`while a or b do x`

Đánh giá cài đặt

- Phép đo độ phức tạp của Halstead

  Phép đo ảnh hưởng đến độ phức tạp:

  - Nỗ lực của chương trình

    $E = (n_1 * N_2 * (N_1+N_2) * \log(n_1+n_2)) / (2*n_2)$

  Với:

  - n_1: Số các toán tử duy nhất được sử dụng
  - n_2: Số các toán hạng duy nhất được sử dụng
  - N_1: Tổng số toán tử được sử dụng
  - N_2: Tổng số toán hạng được sử dụng

Đánh giá kiểm thử

- Công thức xác định thời gian cần có để kiểm thử

$$\frac{\ln\left(\frac{f_{target}}{0.5+f_{target}}\right)}{\ln\left(\frac{0.5+f_{target}}{f_{total}+f_{target}}\right)} \times t_h$$

  - f_{target} là số lượng lỗi dự đoán
  - f_{total} là số lượng lỗi thực sự xảy ra
  - t_h là thời gian kiểm thử xảy ra lỗi

Đánh giá mức độ tin cậy

- Độ tin cậy (realibility) của phần mềm có thể được xác định bằng thời gian trung bình giữa các lỗi (mean-time-between-failure)

  MTBF= MTTF + MTTR

  với

  - MTTF là thời gian trung bình để có lỗi xảy ra (mean-time-to-failure)
  - MTTR là thời gian trung bình để sửa chữa lỗi (mean-time-to-repair).

Đánh giá mức độ sẵn sàng

- Độ sẵn sàng (availability) của phần mềm có thể được xác định theo công thức

$$Availability = \frac{MTTF}{(MTTF + MTTR)} \times 100\%$$

Đánh giá bảo trì

- Sự bền vững của sản phẩm phần mềm

$$SMI = \frac{M_T - (F_a + F_c + F_d)}{M_T}$$

  - M_T : số lượng các module
  - F_c : số lượng các module bị thay đổi
  - F_a : số lượng các module được thêm vào
  - F_d : số lượng các module bị xóa đi

Ước lượng chi phí phần mềm

- Ước lượng chi phí
  - Dựa trên kích thước, độ phức tạp
  - Dựa vào dữ liệu quá khứ
  - Chi phí tỉ lệ với công sức (effort) phát triển phần mềm
  - Chi phí tính dựa theo công sức cho các giai đoạn phát triển phần mềm: khởi đầu, phân tích, thiết kế, cài đặt, kiểm thử nhưng chưa tính đến giai đoạn bảo trì.
  - Đơn vị tính của công sức: người-tháng (hoặc người-ngày)

Xác định kích thước phần mềm

- Đếm số dòng mã lệnh (Lines of Code - LOC)
- Theo số lượng chức năng (Files-Flows-Processes - FFP)
- Tiếp cận hướng đối tượng (Object Points – OP)
- Theo các điểm đặc trưng (Feature Points - FTP)
- Theo số lượng trường hợp sử dụng (Use Case Points – UCP)
- Theo điểm chức năng (Function Points – FP)

Xác định kích thước phần mềm

- Đếm số dòng mã lệnh (Lines of Code - LOC)
  - Có thể sử đếm theo đơn vị ngàn dòng lệnh (thousand lines of code – KLOC, thousand delivered source instructions - KDSI) khi số dòng mã lệnh của một phần mềm là khá lớn.
  - Các vấn đề cần quan tâm khi đếm:
    - Tính toán kích thước cho các giai đoạn khác nhau
    - Cài đặt trên các ngôn ngữ lập trình khác nhau
    - Cách đếm số dòng mã lệnh
    - Mã lệnh tạo ra bằng công cụ dùng để hỗ trợ phát triển

Xác định kích thước phần mềm

- Theo số lượng chức năng (Files-Flows-Processes - FFP)
  - Áp dụng đối với các ứng dụng xử lý dữ liệu có kích thước trung bình.
  - Tập tin (File - Fi): tập hợp các mẩu tin (vật lý hay luận lý) có liên hệ với nhau.
  - Dòng (Flow - Fl): giao diện dữ liệu giữa sản phẩm và môi trường như màn hình, báo cáo,…
  - Xử lý (Process - Pr): (về chức năng mà nói) đó chính là một định nghĩa logic hay toán học dùng để thao tác trên dữ liệu.
  - S = Fi + Fl + Pr

21

Xác định kích thước phần mềm

- Tiếp cận hướng đối tượng (Object Points – OP)
  - Kích thước phần mềm được xác định theo sẽ dựa trên số lượng các kịch bản, các lớp chính, lớp hỗ trợ, tỷ lệ giữa số lượng lớp hỗ trợ và lớp chính và số lượng các hệ thống con.

22

Xác định kích thước phần mềm

- Theo các điểm đặc trưng (Feature Points - FTP)
  - Sử dụng cho các phần mềm chịu ảnh hưởng mạnh về giải thuật:
    - Thời gian thực (real-time software)
    - Nhúng (embedded software)
    - Truyền thông (communication software)
    - FTP = FP – 3 x Maf + 3 x Alg
      - Maf: số lượng tập tin chính
      - Alg: số lượng giải thuật sử dụng
      - FP: số điểm chức năng

23

Xác định kích thước phần mềm

- Theo số lượng trường hợp sử dụng (Use Case Points – UCP)
  Số lượng dòng mã lệnh ước lượng theo các trường hợp sử dụng:

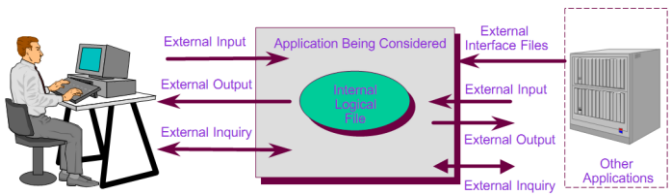
$$LOC_{estimate} = N \times LOC_{avg} + [(\frac{S_a}{S_h} - 1) + (\frac{P_a}{P_h} - 1)] \times LOC_{adjust}$$

  - N: số lượng trường hợp sử dụng hiện hành
  - LOC_{avg} : số lượng LOC trung bình cho hệ thống cùng kiểu
  - LOC_{adjust} : thể hiện sự điều chỉnh dựa trên n% của LOC_{avg} với n được định nghĩa cục bộ và thể hiện sự khác nhau giữa phần mềm này với các phần mềm khác (tính trung bình)
  - S_a : số lượng kịch bản hiện tại cho mỗi trường hợp sử dụng
  - S_h : số lượng kịch bản trung bình cho mỗi trường hợp sử dụng cho hệ thống cùng kiểu
  - P_a : số lượng trang hiện tại cho mỗi trường hợp sử dụng
  - P_h : số lượng trang trung bình cho mỗi trường hợp sử dụng cho hệ thống cùng kiểu

24

Xác định kích thước phần mềm

- Theo điểm chức năng (Function Points – FP): Các loại điểm chức năng
  - EI: External Inputs
  - EO: External Outputs
  - EQ: External Inquiries
  - ILF: Internal Logical File
  - EIF: External Interface File



Xác định kích thước phần mềm

- EI là một tiến trình căn bản trong đó dữ liệu được truyền từ bên ngoài vào bên trong phạm vi của ứng dụng.
  - Dữ liệu có thể từ màn hình nhập liệu hoặc từ một ứng dụng khác.
  - Dữ liệu có thể là thông tin điều khiển hoặc thông tin nghiệp vụ.
  - Dữ liệu (trừ thông tin điều khiển) được sử dụng để cập nhật một hoặc nhiều ILF.

26

Xác định kích thước phần mềm

- EO là một tiến trình căn bản trong đó dữ liệu phát sinh được truyền từ bên trong ra bên ngoài phạm vi ứng dụng.
  - Nó có thể là report hay các tập tin kết xuất được gửi cho ứng dụng khác và được tạo ra từ những thông tin có trong một hay nhiều ILF hoặc EIF.
  - Dữ liệu phát sinh là các dữ liệu được xử lý dựa trên truy tìm trực tiếp và hiệu chỉnh thông tin từ các ILF/IEF. (Dữ liệu phát sinh thường là kết quả của các giải thuật hay sự tính toán.)

27

Xác định kích thước phần mềm

- EQ là một tiến trình căn bản có hai chiều nhập dữ liệu và xuất dữ liệu nhằm truy xuất dữ liệu từ một hay nhiều ILF/EIF.
  - Dữ liệu nhập không cập nhật ILF/EIF và dữ liệu xuất không phải là dữ liệu phát sinh.
  - EQ có thể là các thao tác tìm kiếm, truy vấn dữ liệu từ các ILF/EIF.

28

Xác định kích thước phần mềm

- ILF là một nhóm các dữ liệu có liên quan về mặt logic có thể nhận dạng bởi người sử dụng trong phạm vi ứng dụng và được cập nhật thông qua EI.
- EIF là một nhóm các dữ liệu có liên quan về mặt logic chỉ được sử dụng cho mục đích tham khảo. Dữ liệu nằm ở nên ngoài phạm vi ứng dụng và được cập nhật bởi EI của các ứng dụng khác. Nó được lưu trữ trong tập tin.

Xác định kích thước phần mềm

- Các thành phần cơ bản
  - FTR (File Type Referenced) là một loại tập tin được tham khảo bởi một xử lý. FTR phải là một ILF hoặc EIF.
  - RET (Record Element Type) là các nhóm dữ liệu con liên quan về mặt logic trong ILF/EIF có thể nhận dạng bởi người sử dụng. Hầu hết các RET dựa vào mối quan hệ cha – con.
  - DET (Data Element Type) là một trường không lặp lại, có thể nhận dạng bởi người sử dụng. DET là thông tin động, không phải là thông tin tĩnh. Nó là các trường nhập dữ liệu, các thông báo lỗi, các trường được hiển thị, các giá trị tính toán, các nút.

Mô hình điểm chức năng

Component	RET's	FTR's	DET's
External Inputs (EI)		✓	✓
External Outputs (EO)		✓	✓
External Inquiries (EQ)		✓	✓
External Interface Files (EIF)	✓		✓
Internal Logical Files (ILF)	✓		✓

Tham khảo: Cách tính giá trị các loại điểm chức năng

Số lượng RET	ILF: 1 tới 19	Số lượng DET: 20 – 50	Từ 51 trở lên
1	Thấp (7)	Thấp (7)	Trung bình (10)
2 tới 5	Thấp (7)	Trung bình (10)	Cao (15)
Từ 6 trở lên	Trung bình (10)	Cao (15)	Cao (15)

Số lượng RET	EIF: 1 tới 19	Số lượng DET: 20 – 50	Từ 51 trở lên
1	Thấp (5)	Thấp (5)	Trung bình (7)
2 tới 5	Thấp (5)	Trung bình (7)	Cao (10)
Từ 6 trở lên	Trung bình (7)	Cao (10)	Cao (10)

Số lượng FTR	EI: 1 tới 4	Số lượng DET: 5 – 15	Nhiều hơn 15
Nhỏ hơn 2	Thấp (3)	Thấp (3)	Trung bình (4)
2	Thấp (3)	Trung bình (4)	Cao (6)
Lớn hơn 2	Trung bình (4)	Cao (6)	Cao (6)

EO

Số lượng FTR	Số lượng DET: 1 tới 5	6 – 19	Nhiều hơn 19
Nhỏ hơn 2	Thấp (4)	Thấp (4)	Trung bình (5)
2 hoặc 3	Thấp (4)	Trung bình (5)	Cao (7)
Lớn hơn 3	Trung bình (5)	Cao (7)	Cao (7)

EQ

Số lượng FTR	Số lượng DET: 1 tới 5	6 – 19	Nhiều hơn 19
Nhỏ hơn 2	Thấp (3)	Thấp (3)	Trung bình (4)
2 hoặc 3	Thấp (3)	Trung bình (4)	Cao (6)
Lớn hơn 3	Trung bình (4)	Cao (6)	Cao (6)

Xác định kích thước phần mềm

- Trọng số cho các dạng chức năng

Function Type	Weighting factors		
	Simple	Average	Complex
External Inputs	3	4	6
External Ouputs	4	5	7
External Inquiries	3	4	6
Internal Logical Files	7	10	15
External Interface Files	5	7	10

- Công thức tính số điểm chức năng chưa được hiệu chỉnh

UFP = a1 x EI + a2 x EO + a3 x EQ + a4 x ILF + a5 x EIF

với a1, a2, a3, a4, a5 là giá trị các điểm chức năng theo độ phức tạp cho trong bảng trên.

Xác định kích thước phần mềm

- Công thức tính điểm chức năng FP

FP = Điểm chức năng thô x (0.65 + 0.01 x Tổng các mức độ ảnh hưởng của các nhân tố hiệu chỉnh)

- 14 nhân tố hiệu chỉnh (có mức độ ảnh hưởng nằm trong phạm vi từ 0 (không quan trọng hay không thích hợp hay không ảnh hưởng) tới 5 (cực kỳ quan trọng hay cần thiết tuyệt đối hay ảnh hưởng nhất)

Xác định kích thước phần mềm

- Các nhân tố hiệu chỉnh

1. Data communication
2. Distributed function
3. Performance
4. Heavily used configuration
5. Transaction rate
6. Online data entry
7. End-user efficiency
8. Online update
9. Complex processing
10. Reusability
11. Installation ease
12. Operation ease
13. Multiple site
14. Facilitate change

Xác định kích thước phần mềm

- Điểm chức năng FP có thể được dùng để dự đoán số dòng lệnh LOC

LOC = AVC * số điểm chức năng FP

với AVC : yếu tố phụ thuộc vào ngôn ngữ lập trình được sử dụng

Xác định kích thước phần mềm

Ngôn ngữ	Giá trị nhỏ nhất	Giá trị phổ biến nhất	Giá trị lớn nhất
Ada 83	45	80	125
Ada 95	30	50	70
C	60	128	170
C#	40	55	80
C++	40	55	140
Cobol	65	107	150
Fortran 90	45	80	125
Fortran 95	30	71	100
Java	40	55	80
Macro Assembly	130	213	300
Perl	10	20	30
(Pascal, Fortran 77,…)	65	107	160
Smalltalk	10	20	40
SQL	7	13	15
VB	15	32	41

Tham khảo thêm Bảng chuyển đổi giữa LOC và FP trong Giáo trình Nhập môn Công Nghệ Phần Mềm

37

Ước lượng chi phí phần mềm

- Ước lượng thực nghiệm
  - Mô hình Walston và Felix
  - Mô hình Bailey và Basili
  - Mô hình COCOMO
  - Tham khảo thêm các mô hình khác trong giáo trình

38

Mô hình Walston và Felix

- Một trong các mô hình giải thuật sớm nhất (1977)
- Mô hình được đề nghị sau khi nghiên cứu 60 dự án của IBM
- Có 29 yếu tố ảnh hưởng tới hiệu suất
- Công thức ước lượng *công sức E*

$$E = 5.2 \times S^{0.91}$$ (người - tháng)

Với S là kích thước được ước lượng của hệ thống (*theo KDSI*)

- Công thức ước lượng *thời gian thực hiện T*

$$T = 2.5 \times E^{0.35}$$ (tháng)

39

Mô hình Walston và Felix

- Mỗi yếu tố sẽ nhận 1 trong 3 giá trị tùy thuộc vào sự tác động của nó tới hiệu suất
  - 1 (cao): làm tăng hiệu suất
  - 0 (trung bình): không ảnh hưởng tới hiệu suất
  - -1 (thấp): làm giảm hiệu suất

40

Mô hình Walston và Felix

1. Customer interface complexity	16. Use of design and code inspections
2. User participation in requirements definition	17. Use of top-down development
3. Customer-originated program design changes	18. Use of a chief programmer team
4. Customer experience with the application area	19. Overall complexity of code
5. Overall personnel experience	20. Complexity of application processing
6. Percentage of development programmers who participated in the design of functional specifications	21. Complexity of program flow
7. Previous experience with the operational computer	22. Overall constraints on program's design
8. Previous experience with the programming language	23. Design constraints on the program's main storage
9. Previous experience with applications of similar size and complexity	24. Design constraints on the program's timing
10. Ratio of average staff size to project duration (people per month)	25. Code for real-time or interactive operation or for execution under severe time constraints
11. Hardware under concurrent development	26. Percentage of code for delivery
12. Access to development computer open under special request	27. Code classified as nonmathematical application and input/output formatting programs
13. Access to development computer closed	28. Number of classes of items in the database per 1000 lines of code
14. Classified security environment for computer and at least 25% of programs and data	29. Number of pages of delivered documentation per 1000 lines of code
15. Use of structured programming	

41

Mô hình Bailey và Basili

- Mô hình được đề nghị năm 1981 bởi Bailey và Basili
- Mô hình này sử dụng cơ sở dữ liệu của 18 dự án viết bằng ngôn ngữ Fortran tại trung tâm Goddard Space Flight của NASA
- Các nhóm yếu tố ảnh hưởng tới công sức: phương pháp, độ phức tạp và kinh nghiệm
- Công thức ước lượng công sức ban đầu E

  $E = 5.5 + 0.73 \times S^{1.16}$ (người - tháng)

  với S là kích thước được ước lượng của hệ thống (*theo KDSI*)

42

Mô hình Bailey và Basili

- Mỗi yếu tố ảnh hưởng tới công sức nhận một trong các giá trị từ 0 đến 5

Total methodology (METH)	*Cumulative complexity (CPLX)*	*Cumulative experience (EXP)*
Tree charts	Customer interface complexity	Programmer qualifications
Top-down design	Application complexity	Programmer machine experience
Formal documentation	Program flow complexity	Programmer language experience
Chief programmer teams	Internal communication complexity	Programmer application experience
Formal training	Database complexity	Team experience
Formal test plans	External communication complexity	
Design formalisms	Customer-initiated program design changes	
Code reading		
Unit development folders		

43

Mô hình COCOMO 81

- Mô hình COCOMO 81 được đề nghị bởi Boehm
  - Dạng cơ bản: áp dụng cho nhóm nhỏ, môi trường quen thuộc
  - Dạng trung bình: áp dụng cho dự án khá lớn, có một ít kinh nghiệm
  - Dạng lớn: áp dụng cho dự án lớn, môi trường mới
- Bảng các hệ số khi phát triển sản phẩm (sử dụng mô hình COCOMO 81 trung gian)

Software project	a_b	b_b	c_b	d_b
Dạng cơ bản (organic)	3.2	1.05	2.50	0.38
Dạng trung bình (semi-detached)	3.0	1.12	2.50	0.35
Dạng lớn (embedded)	2.8	1.20	2.50	0.32

Mô hình COCOMO 81 trung gian

- Công sức $E = a_b \times S^{b_b} \times EAF$
  - a_b và b_b: được xác định dựa vào bảng mức độ khó khi phát triển phần mềm
  - EAF (effort adjustment factor): hệ số hiệu chỉnh công sức. Nó được tính bằng tích của các hệ số phát triển
  - S là kích thước được ước lượng của hệ thống (*theo KDSI*)
- Thời gian $T = c_b \times E^{d_b}$
- Số lượng người $P = E/T$

Mô hình COCOMO 81 trung gian

- Các hệ số phát triển

Cost Drivers	Rating					
	Very Low	Low	Nominal	High	Very High	Extra High
Product Attributes						
Required software reliability	0.75	0.88	1.00	1.15	1.40	
Database size		0.94	1.00	1.08	1.16	
Product complexity	0.70	0.85	1.00	1.15	1.30	1.65
Computer Attributes						
Execution time constraint			1.00	1.11	1.30	1.66
Main storage constraint			1.00	1.06	1.21	1.56
Virtual machine volatility*		0.87	1.00	1.15	1.30	
Computer turnaround time		0.87	1.00	1.07	1.15	
Personnel Attributes						
Analyst capabilities	1.46	1.19	1.00	0.86	0.71	
Applications experiences	1.29	1.13	1.00	0.91	0.82	
Programmer capability	1.42	1.17	1.00	0.86	0.70	
Virtual machine experience*	1.21	1.10	1.00	0.90		
Programming language experiences	1.14	1.07	1.00	0.95		
Project Attributes						
Use of modern programming pratices	1.24	1.10	1.00	0.91	0.82	
Use of software tools	1.24	1.10	1.00	0.91	0.83	
Required development schedule	1.23	1.08	1.00	1.04	1.10	

Mô hình COCOMO 2

- Mô hình COCOMO 81 được phát triển trên giả thiết rằng tiến trình phát triển phần mềm thác nước được sử dụng và tất cả các phần mềm được phát triển từ đầu.
- Mô hình COCOMO 2 được thiết kế để thích ứng với những cách tiếp cận khác nhau đối với sự phát triển phần mềm

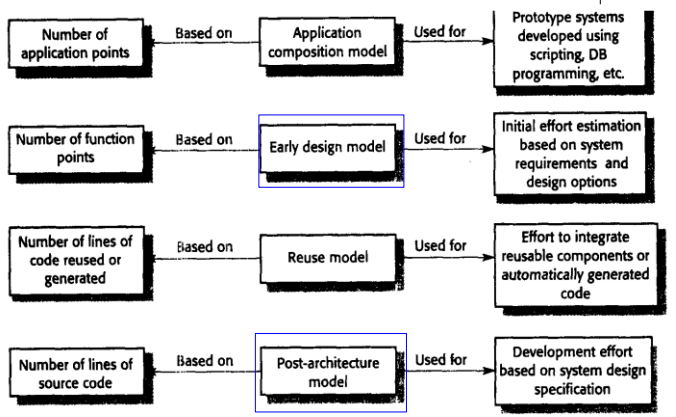
Mô hình COCOMO 2

- COCOMO 2 kết hợp chặt chẽ các mô hình con nhằm đưa ra các dự đoán phần mềm chi tiết
- Các mô hình con trong COCOMO 2:
  - Mô hình Application composition. Mô hình này giả sử rằng hệ thống được tạo thành từ các thành phần có thể tái sử dụng. Nó được thiết kế để ước lượng công sức phát triển bản mẫu.
  - Mô hình Early design: được sử dụng tại giai đoạn thiết kế kiến trúc khi các yêu cầu là có sẵn (và thiết kế chi tiết vẫn chưa được bắt đầu)

Mô hình COCOMO 2

- Các mô hình con trong COCOMO 2:
  - Mô hình Reuse: được sử dụng để tính công sức tích hợp các thành phần có thể dùng lại và / hoặc mã chương trình được tự động sinh ra bởi các công cụ dịch chương trình hay thiết kế. Nó thường được sử dụng kết hợp với mô hình Post - architecture.
  - Mô hình Post-architecture: được sử dụng ngay khi kiến trúc hệ thống đã được thiết kế và các thông tin chi tiết hơn về hệ thống là sẵn có.

Mô hình COCOMO 2



Mô hình Application composition

- Để ước lượng công sức cần để lập bản mẫu các dự án và cho các dự án được phát triển bằng cách kết hợp các thành phần sẵn có.
- Được dựa trên số lượng điểm ứng dụng (đối tượng).
- Công thức ước lượng công sức

  E = (NAP x (1 - %reuse/100)) / PROD (người – tháng)
  - NAP: số lượng điểm ứng dụng.
  - %reuse: % mã lệnh được tái sử dụng từ các dự án khác.
  - PROD: hiệu suất, phụ thuộc vào kinh nghiệm và khả năng của nhà phát triển cũng như tính trưởng thành và khả năng của công cụ .

Mô hình Application composition

- Bảng xác định hiệu suất PROD

Developers' experience and capability	Very low	Low	Nominal	High	Very high
CASE maturity and capability	Very low	Low	Nominal	High	Very high
Productivity factor	4	7	13	25	50

Mô hình Application composition

- Số lượng điểm ứng dụng NAP phụ thuộc vào:
  - Các màn hình riêng biệt được hiển thị (giao diện người dùng)
  - Các báo cáo được sinh ra bởi hệ thống
  - Các thành phần thư viện

53

Mô hình Application composition

- Tính số lượng điểm ứng dụng
  - Đếm số lượng màn hình, báo cáo và module
  - Xác định độ phức tạp cho từng thành phần (1 màn hình hay 1 báo cáo hay 1 module) theo bảng sau

For Screens				*For Reports*			
	Number and source of data tables				*Number and source of data tables*		
Number of views contained	Total < 4 (<2 server, <3 client)	Total < 8 (2-3 server, 3-5 client)	Total 8+ (>3 server, >5 client)	*Number of sections contained*	Total < 4 (<2 server, <3 client)	Total < 8 (2-3 server, 3-5 client)	Total 8+ (>3 server, >5 client)
<3	simple	simple	medium	0 or 1	simple	simple	medium
3 - 7	simple	medium	difficult	2 or 3	simple	medium	difficult
8 +	medium	difficult	difficult	4 +	medium	difficult	difficult

54

Mô hình Application composition

- Tính số điểm ứng dụng cho từng thành phần khi đã biết độ phức tạp theo bảng dưới đây

Object type	*Simple*	*Medium*	*Difficult*
Screen	1	2	3
Report	2	5	8
3GL component	-	-	10

- Tính tổng số điểm ứng dụng cho tất cả các thành phần

55

Mô hình Early design

- Ước lượng công sức khi các yêu cầu đã được chấp nhận và thiết kế hệ thống đang được thực hiện
- Công thức ước lượng công sức

  $E = a \times S^b \times M$ với
  - M: tích của 7 hệ số nhân
  - a : hằng số thực nghiệm (2.5)
  - S kích thước được ước lượng của hệ thống (*theo KDSI*)
  - $b = 1.01 + 0.01 \times \sum W_i$ (i:1 - 5)

56

Mô hình Early design

- Các hệ số nhân phản ánh khả năng của nhà phát triển, các yêu cầu phi chức năng, sự hiểu biết rõ về nền tảng phát triển, v.v.
  - RCPX - product reliability and complexity
  - RUSE - the reuse required
  - PDIF - platform difficulty
  - PREX - personnel experience
  - PERS - personnel capability
  - SCED - required schedule
  - FCIL - the team support facilities
- Giá trị của các hệ số này nằm trong khoảng từ 0 (rất thấp) đến 5 (vô cùng cao)

57

Mô hình Early design

- Các hệ số W

W_i	Very Low	Low	Norminal	High	Very High	Extra High
Precedentedness	4.05	3.24	2.43	1.62	0.81	0
Development flexibility	6.07	4.86	3.64	2.43	1.21	0
Architecture/Risk resolution	4.22	3.38	2.53	1.69	0.84	0
Team cohesion	4.94	3.95	2.97	1.98	0.99	0
Process maturity	4.54	3.64	2.73	1.82	0.91	0

58

Mô hình Post-architecture

- Sử dụng cùng một công thức như mô hình early design nhưng có tới 17 hệ số nhân
  - Công sức $E = a \times S^b \times M$ với
    - M: tích của 17 hệ số nhân
    - a : hằng số thực nghiệm (2.5)
    - S kích thước được ước lượng của hệ thống (*theo KDSI*)
    - $b = 1.01 + 0.01 \times \sum W_i$ (i: 1 – 5)

59

Mô hình Post-architecture

- 17 hệ số nhân M

Cost Drivers	Very Low	Low	Norminal	High	Very High	Extra High
Product Factors						
Required software reliability	0.75	0.88	1.00	1.15	1.39	
Database size		0.93	1.00	1.09	1.19	
Product complexity	0.75	0.88	1.00	1.15	1.30	1.66
Required reusability		0.91	1.00	1.14	1.29	1.49
Documentation needs	0.89	0.95	1.00	1.06	1.13	
Computer Factors						
Execution time constraint			1.00	1.11	1.31	1.67
Main storage constraint			1.00	1.06	1.21	1.57
Platform volatility		0.87	1.00	1.15	1.30	
Personal factors						
Analyst capability	1.50	1.22	1.00	0.83	0.67	
Programmer cabability	1.37	1.16	1.00	0.87	0.74	
Applications experience	1.22	1.10	1.00	0.89	0.81	
Platform experience	1.24	1.10	1.00	0.92	0.84	
Language and tool experience	1.25	1.12	1.00	0.88	0.81	
Personnel continuity	1.24	1.10	1.00	0.92	0.84	
Project Factors						
Use of software tools	1.24	1.12	1.00	0.86	0.72	
Multi-site development	1.25	1.10	1.00	0.92	0.84	0.78
Required development schedule	1.29	1.10	1.00	1.00	1.00	

60

Xác định giá trị phần mềm theo công văn 2589/BTTTT

- Sinh viên tự đọc thêm

HẾT PHẦN III