chương 2

* Xây dựng hệ thống
* Mùa thu 2011

Phát triển và hỗ trợ một hệ thống

* Một hệ thống là gì?
  + Là một chương trình duy nhất một hệ thống?
  + Có bao nhiêu chương trình phải có?
  + Nó có liên quan đến phần cứng, chương trình, quy trình kinh doanh, và những thứ khác không?
* Có sự khác nhau giữa phát triển và hỗ trợ a) một chương trình đơn lẻ so với b) một hệ thống không?

Kích thước và độ phức tạp

* Vấn đề rộng (nhiều hơn về số):
  + Các chức năng chính
  + Các tính năng trong từng chức năng chính
  + Giao diện giữa các tính năng và chức năng
  + Giao diện cho các hệ thống "bên ngoài"
  + Số lượng người dùng đồng thời và tổng số người dùng
* Độ sâu vấn đề (liên kết)
  + Số lượng Nesting
  + Số lượng mối quan hệ "chuỗi"
* Độ sâu và chiều rộng ảnh hưởng đến độ phức tạp như thế nào? (xem trang trình bày tiếp theo)

Kích thước và độ phức tạp tăng

* Thực hiện tác vụ A
* Thực hiện tác vụ B
* Thực hiện tác vụ C
* Khởi đầu
* Dừng lại
* Khởi đầu
* Chờ tín hiệu
* Tín hiệu là gì?
* một
* b
* khác
* Thực hiện tác vụ A
* Thực hiện tác vụ B
* Thực hiện tác vụ C
* Dừng lại
* Thực hiện tác vụ A2
* *(a) Đơn giản*
* *(b) Tăng kích thước và độ phức tạp*

Độ sâu và chiều rộng tăng

* Trong trang trình bày trước, chỉ một vài mục được thêm vào.
  + Chờ tín hiệu
  + Quyết định xử lý dựa trên tín hiệu
* Hãy tưởng tượng điều gì xảy ra khi chúng ta đang phát triển một hệ thống nơi độ sâu và chiều rộng (hoặc độ phức tạp) tăng thêm 100 hoặc 1000 mục và quyết định khác.

Các vấn đề "kỹ thuật": Phát triển hệ thống

* Vấn đề và giải pháp
  + Phân hủy
  + Modularization
  + Tách biệt
  + Lặp lại gia tăng
* Lựa chọn công nghệ và công cụ
  + Nền tảng phát triển
  + Ngôn ngữ phát triển
  + Cơ sở dữ liệu
  + Mạng lưới
  + Quản lý cấu hình
* Quy trình và phương pháp luận
  + Lựa chọn quy trình
  + Lựa chọn phương pháp luận
  + Các công cụ lựa chọn để hỗ trợ quá trình

Các vấn đề “phi kỹ thuật”: Phát triển hệ thống

* Dự đoán và lập kế hoạch nỗ lực dự án
  + Cần xem xét và ước tính nhiều mục hơn
  + Cần phối hợp nhiều mặt hàng hơn về điều kiện tiên quyết và điều kiện tiên quyết
  + Cần xem xét nhiều tiềm năng rủi ro và biến thể hơn
* Nhiệm vụ và Truyền thông
  + Nhiều người có nhiều kỹ năng hơn
  + Thêm thông tin liên lạc giữa những người
  + Các lỗi và sửa đổi khác

Tăng số lượng thông tin liên lạc khi số lượng người tăng lên. Ngoài ra, sự gia tăng xác suất lỗi.

* 2 người:
* 1 đường dẫn
* 4 người:
* có thể 6 con đường
* 6 nguoi:
* tăng lên 15 đường dẫn tiềm năng
* Với sự gia tăng phức tạp của hệ thống, có một tương ứng
* tăng "nhân lực" hoặc nguồn nhân lực.

Một hệ thống phức tạp lớn

* Xây dựng hệ thống “Nhiệm vụ quan trọng” hoặc “Kinh doanh quan trọng” (như bảng lương) yêu cầu một số hoạt động riêng biệt được thực hiện bởi hơn 1 người:
  + Yêu cầu: thu thập, phân tích, đặc điểm kỹ thuật và thỏa thuận
  + Thiết kế: trừu tượng, phân tích, gắn kết, tương tác và phân tích khớp nối
  + Thực hiện: mã hóa và kiểm tra đơn vị
  + Tích hợp và theo dõi các phần và các bộ phận
  + Thử nghiệm riêng biệt: kiểm tra chức năng, thử nghiệm thành phần, kiểm tra hệ thống và thử nghiệm hiệu suất
  + Đóng gói và giải phóng hệ thống

Các thực thể dọc và ngang cần được thiết kế, triển khai, tích hợp, thử nghiệm và phát hành

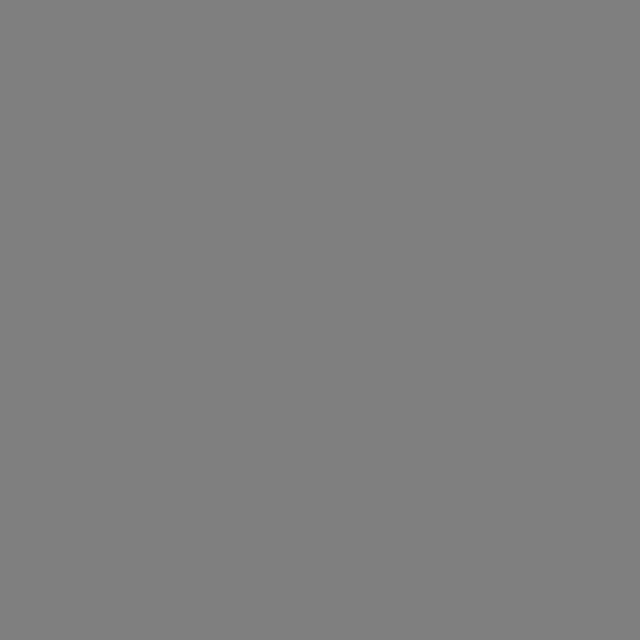
* Màn hình chung
* .
* .
* .
* Tin nhắn chung
* Các hàm dọc khác nhau
* Ngang
* chức năng

Hỗ trợ một hệ thống

* Giáo dục và chuẩn bị trước khi phát hành
  + Số lượng người dùng dự kiến
  + Số lượng sự cố đã biết và chất lượng mong muốn
  + Số lượng người dùng và nhân viên hỗ trợ đào tạo
  + số chu kỳ sửa chữa và bảo trì
* Hỗ trợ khách hàng sau khi phát hành và hỗ trợ khách hàng
  + Gọi cho trung tâm và giải quyết vấn đề
  + Sửa lỗi chính và thay đổi mã
  + Sửa đổi và cải tiến chức năng

Cần nỗ lực phối hợp trong phát triển và hỗ trợ hệ thống

* Bởi vì có nhiều phần và phần để xem xét trong Hệ thống hơn là một chương trình được phát triển bởi một người duy nhất cho một số lượng người dùng hạn chế, cần có sự điều phối của:
  + *Quy trình và phương pháp luận*
  + *Sản phẩm cuối cùng và các đồ tạo tác trung gian*
  + *Con người (nhà phát triển, nhân viên hỗ trợ và người dùng)*
* *Biểu đồ trước đây về con người tăng và đường dẫn truyền thông tiềm năng*
* *tăng cung cấp một đầu mối cho tầm quan trọng của các nỗ lực phối hợp.*

**https://www.gstatic.com/translate/infowindow/iws_n.pnghttps://www.gstatic.com/translate/infowindow/iws_n.pnghttps://www.gstatic.com/translate/infowindow/iws_w.pnghttps://www.gstatic.com/translate/infowindow/iws_e.pnghttps://www.gstatic.com/translate/infowindow/iws_s.pnghttps://www.gstatic.com/translate/infowindow/iws_s.png**

**https://www.gstatic.com/translate/infowindow/iw_n.pnghttps://www.gstatic.com/translate/infowindow/iw_n.pnghttps://www.gstatic.com/translate/infowindow/iw_w.pnghttps://www.gstatic.com/translate/infowindow/iw_e.pnghttps://www.gstatic.com/translate/infowindow/iw_s0.pnghttps://www.gstatic.com/translate/infowindow/iw_s0.png**

**https://www.gstatic.com/images/branding/googlelogo/1x/googlelogo_color_48x16dp.png**

Văn bản Tiếng Anh gốc:

Does it have to involve hardware, programs, business process, and others?

**http://www.google.com/images/zippy_plus_sm.gif** Đóng góp bản dịch hay hơn

Chương 3

* Kỹ thuật phần mềm
* Mùa thu 2011

# Phần mềm

* KỸ THUẬT PHẦN MỀM
* <số>
* Q: Nếu bạn phải viết một chương trình 10.000 dòng trong C để giải quyết một vấn đề, nó sẽ mất bao lâu?
* Trả lời: thường từ 2-4 tháng
* Hãy để chúng tôi phân tích năng suất
  + Năng suất = (tài nguyên đầu ra) / (tài nguyên đầu vào)
  + Trong SW dòng mã (LOC) được coi là đầu ra
  + Nguồn lực đầu vào là nỗ lực - Số tháng người (PM)
    - chi phí trên không được lập mô hình theo tỷ lệ cho tháng người
  + Mặc dù không hoàn hảo, một số biện pháp năng suất là cần thiết, vì dự án phải giữ nó cao

# Phần mềm …

* KỸ THUẬT PHẦN MỀM
* <số>
* Năng suất là 2,5-5 KLOC / PM
  + 10.000 / 2 = 5 KLOC / CH
* Hỏi: Năng suất trong một tổ chức SW thương mại điển hình là gì?
  + A: Từ 100 đến 1000 LOC / PM
* Q: Tại sao nó thấp, khi năng suất của bạn quá cao? (những người như bạn làm việc trong ngành)
  + A: Những gì học sinh đang xây dựng và những gì ngành công nghiệp xây dựng là hai điều khác nhau
* Các từ viết tắt
  + PM - Người Tháng
  + KLOC - 1000 dòng mã

# Phần mềm…

* KỸ THUẬT PHẦN MỀM
* <số>
* Trong một trường đại học, một hệ thống sinh viên được xây dựng trong khi tổ chức thương mại xây dựng phần mềm sức mạnh công nghiệp
* Phần mềm (IEEE): thu thập các chương trình, quy trình, quy tắc và tài liệu và dữ liệu liên quan
* Sự khác biệt giữa chương trình học sinh và phần mềm sức mạnh công nghiệp cho cùng một vấn đề là gì?

# Phần mềm…

* KỸ THUẬT PHẦN MỀM
* <số>
* Sinh viên
* Nhà phát triển là người dùng
  + Lỗi có thể chấp nhận được
  + Giao diện người dùng không quan trọng
  + Không có tài liệu
* Sức mạnh công nghiệp
* Những người khác là người dùng
  + Lỗi không được dung thứ
  + Vấn đề triển khai giao diện người dùng v.
  + Tài liệu cần thiết cho người dùng cũng như cho tổ chức và dự án

# Phần mềm…

* KỸ THUẬT PHẦN MỀM
* <số>
* Sinh viên
* SW không sử dụng quan trọng
* Độ tin cậy, mạnh mẽ không quan trọng
* Không đầu tư
* Không quan tâm đến tính di động
* Sức mạnh công nghiệp
* Hỗ trợ các chức năng / doanh nghiệp quan trọng
* Độ tin cậy, mạnh mẽ là rất quan trọng
* Đầu tư lớn
* Tính di động là một vấn đề quan trọng ở đây

# Phần mềm cường độ công nghiệp

* KỸ THUẬT PHẦN MỀM
* <số>
* Chương trình sinh viên và phần mềm sức mạnh công nghiệp là hai điều khác nhau
* Sự khác biệt chính là về chất lượng (bao gồm khả năng sử dụng, độ tin cậy, tính di động, v.v.)
  + Chất lượng cao yêu cầu thử nghiệm nặng, chiếm 30-50% tổng nỗ lực phát triển
  + Phát triển được chia thành các giai đoạn sao cho các lỗi có thể được phát hiện trong mỗi
  + Giao diện người dùng tốt, sao lưu, dung sai lỗi, theo tiêu chuẩn vv tăng kích thước cho cùng một chức năng

# Phần mềm cường độ công nghiệp

* KỸ THUẬT PHẦN MỀM
* <số>
* Nếu năng suất công nghiệp là 1/5 của sinh viên, và tăng kích thước của một yếu tố 2, phần mềm sức mạnh công nghiệp sẽ mất 10 lần nỗ lực
* Quy tắc ngón tay cái của Brooks: Sw sức mạnh công nghiệp tốn 10 lần so với sinh viên sw
* Tên miền của SW Engineering: Sức mạnh công nghiệp SW
* Trong SW Engineering và trong khóa học này, phần mềm có nghĩa là phần mềm sức mạnh công nghiệp

# Phần mềm đắt tiền

* KỸ THUẬT PHẦN MỀM
* <số>
* Chúng ta hãy xem xét chi phí liên quan
  + Năng suất = 500 LOC / PM
  + Chi phí cho công ty = $ 10K / PM
  + Chi phí mỗi LOC = 20 đô la (10.000 đô la / 500 đô la)
  + Tức là, mỗi dòng mã được phân phối có giá khoảng 20 đô la.
* Một ứng dụng đơn giản cho một doanh nghiệp có thể có 20KLOC đến 50KLOC
  + Chi phí = $ 100K đến $ 1 triệu ($ 20 \* 50.000 = $ 1 triệu)
  + Có thể dễ dàng chạy trên phần cứng $ 10K- $ 20K
  + Vì vậy, chi phí HW trong một giải pháp CNTT là nhỏ so với chi phí SW.

# Phần mềm đắt tiền…

* KỸ THUẬT PHẦN MỀM
* <số>
* Tỷ lệ HW / SW cho một hệ thống máy tính đã cho thấy một sự đảo ngược từ những năm đầu.
  + Trong những năm 50, HW: SW :: 80:20
  + Trong những năm 80, HW: SW :: 20:80
* Vì vậy, SW là rất tốn kém
  + Tầm quan trọng của việc tối ưu hóa HW không nhiều
  + Quan trọng hơn để tối ưu hóa SW

# Late & Unreliable

* KỸ THUẬT PHẦN MỀM
* <số>
* 20-25% dự án SW không bao giờ hoàn thành
  + Bởi vì sau một thời gian họ nhận ra rằng chi phí cuối cùng sẽ cao hơn nhiều
* Nhiều công ty báo cáo runaways
  + ngân sách & chi phí ngoài tầm kiểm soát
  + thuê các công ty tư vấn để giúp kiểm soát họ
* Một cuộc khảo sát quốc phòng cho thấy 70% các vấn đề về thiết bị là do SW
* Nhiều ví dụ về lỗi phần mềm

# Tại sao các dự án không thành công thường xuyên

* KỸ THUẬT PHẦN MỀM
* <số>
* Mục tiêu của dự án không thực tế hoặc không được đề xuất
* Ước tính không chính xác về các tài nguyên cần thiết
* Yêu cầu hệ thống được xác định không đúng
* Báo cáo kém về trạng thái của dự án
* Rủi ro không được quản lý
* Giao tiếp kém giữa khách hàng, nhà phát triển và người dùng
* Sử dụng công nghệ chưa trưởng thành
* Không có khả năng xử lý sự phức tạp của dự án
* Thực hành phát triển cẩu thả
* Quản lý dự án kém
* Chính trị các bên liên quan
* Áp lực thương mại
* KỸ THUẬT PHẦN MỀM
* <số>

# Tại sao dự án phần mềm thất bại?

* KỸ THUẬT PHẦN MỀM
* <số>
* 400 dự án ở Hoa Kỳ, Úc và Chile [http://www.developerdotstar.com/mag/articles/software\_success\_failure.html](https://translate.google.com/translate?hl=vi&prev=_t&sl=en&tl=vi&u=http://www.developerdotstar.com/mag/articles/software_success_failure.html)
  + 60% các tổ chức không có quy trình để đo lường lợi ích
  + 86% dự án có trường hợp kinh doanh, nhưng 60% đã bỏ qua nó
  + 33% dự án cho biết họ không có rủi ro, nhưng 62% số dự án không thành công
  + 49% các tổ chức đã có (một hoặc nhiều) thất bại của dự án
  + Trong một phần ba các dự án, người quản lý dự án không nói gì trong kế hoạch / mục tiêu ngân sách
  + 75% dự án bị đánh giá thấp, không ai được đánh giá quá cao
  + 5% dự án không có người quản lý dự án; 16% thay đổi người quản lý dự án ít nhất một lần (và điều đó tương quan với thất bại dự án)

# Không đáng tin cậy…

* KỸ THUẬT PHẦN MỀM
* <số>
* Lỗi SW khác với thất bại của các hệ thống cơ khí hoặc điện
* Trong phần mềm, thất bại không phải do các vấn đề liên quan đến lão hóa
* Thất bại xảy ra do lỗi hoặc lỗi được giới thiệu trong quá trình phát triển
* Tức là lỗi gây ra lỗi xảy ra ngay từ đầu, chỉ hiển thị sau

# Bảo trì

* KỸ THUẬT PHẦN MỀM
* <số>
* Khi SW được chuyển, nó sẽ chuyển sang giai đoạn bảo trì
* Tại sao cần bảo trì cho SW khi nó không mặc theo tuổi?
  + Các lỗi còn lại yêu cầu sửa chữa bảo trì
  + Nâng cấp và thay đổi môi trường - bảo trì thích ứng
* Trong cuộc sống SW, bảo trì có thể tốn nhiều hơn chi phí phát triển của SW

# Dự án phần mềm

* Các yếu tố để thành công:
  + *Sự tham gia của người sử dụng*
  + *Hỗ trợ quản lý điều hành*
  + *Xóa câu lệnh yêu cầu*
  + *Kế hoạch phù hợp*
* Lý do thất bại hàng đầu:
  + *Thiếu đầu vào của người dùng*
  + *Yêu cầu không đầy đủ*
  + *Thay đổi yêu cầu*

# Nguồn của các vấn đề về sản phẩm phần mềm

* Lỗi mã: 38,33%
* Lỗi thiết kế: 24,17%
* Lỗi tài liệu: 13,33%
* Yêu cầu lỗi: 12,50%
* Lỗi khắc phục lỗi: 11,67%
* Chúng ta có nên lo lắng về việc mã hóa nhiều hơn hoặc yêu cầu nhiều hơn, tại sao?

# Xóa lỗi sớm ...

* Quy trình phần mềm
* <số>

# Kỹ thuật phần mềm

* Điều gì là cần thiết để phát triển các sản phẩm phần mềm phức tạp và lớn và những gì cần thiết để kiểm soát các dự án như vậy?
* Cần có thêm “kỷ luật” trong lĩnh vực này:
  + "KỸ THUẬT PHẦN MỀM"
* ( *Hội nghị NATO - 1968)*

# Kỹ thuật phần mềm là gì

* Sommerville - *"một kỷ luật kỹ thuật mà tập trung là phát triển hiệu quả chi phí của hệ thống phần mềm chất lượng cao"*
* Pfleeger - *ứng dụng các công cụ tính toán để giải quyết các vấn đề*
* CMU / SEI-90-TR-003 *- “dạng kỹ thuật áp dụng các nguyên tắc của khoa học máy tính và toán học để đạt được các giải pháp tiết kiệm chi phí cho các vấn đề về phần mềm*
* IEEE std 610-1990 - *"áp dụng một cách tiếp cận có hệ thống, có kỷ luật, định lượng để phát triển, vận hành và bảo trì phần mềm"*

# Công nghệ phần mềm (Tsui và Karam)

* Công nghệ phần mềm là một lĩnh vực rộng lớn liên quan đến tất cả các khía cạnh của việc phát triển và hỗ trợ một hệ thống phần mềm, trải rộng trên các lĩnh vực chính sau đây:
  + *1. Quy trình kỹ thuật và kinh doanh*
  + *2. Các phương pháp và kỹ thuật cụ thể*
  + *3. Đặc tính sản phẩm và số liệu cho phép đo*
  + *4. Kỹ năng con người và làm việc nhóm*
  + *5. Phối hợp và quản lý dự án*

# Kỹ sư phần mềm

* Phần mềm là một doanh nghiệp nghiêm túc
  + *Đạt tới 180 tỷ đô la năm 2000*
  + *Nó phổ biến trên nhiều ngành công nghiệp*
* Phần mềm là một mặt hàng tăng "Giá trị"
* Việc kinh doanh phần mềm đã tốt nghiệp từ một hoạt động "nhà để xe" cho một doanh nghiệp "doanh nghiệp"
* Chúng ta cần phải coi kĩ nghệ phần mềm như một nghề kỹ thuật

# Báo cáo phiên bản 5.2 của IEEE-CS / ACM

* 8 nguyên tắc đạo đức và thực hành chuyên môn trong kỹ nghệ phần mềm
  + Các kỹ sư phần mềm phải hành động nhất quán với lợi ích công cộng
  + Các kỹ sư phần mềm sẽ hành động theo cách có lợi nhất cho khách hàng và chủ nhân của họ, phù hợp với lợi ích công cộng
  + Các kỹ sư phần mềm phải đảm bảo rằng các sản phẩm của họ và các sửa đổi có liên quan đáp ứng các tiêu chuẩn chuyên nghiệp cao nhất có thể
  + Kỹ sư phần mềm sẽ duy trì tính toàn vẹn và độc lập trong bản án chuyên môn của họ
  + Các nhà quản lý kỹ thuật phần mềm và các nhà lãnh đạo sẽ đăng ký và thúc đẩy một cách tiếp cận đạo đức để quản lý phát triển và bảo trì phần mềm
  + Các kỹ sư phần mềm sẽ nâng cao tính toàn vẹn và danh tiếng của nghề phù hợp với lợi ích công cộng
  + Kỹ sư phần mềm phải công bằng và hỗ trợ đồng nghiệp của họ
  + Các kĩ sư phần mềm sẽ tham gia vào việc học suốt đời liên quan đến việc thực hành nghề nghiệp của họ và sẽ thúc đẩy một cách tiếp cận đạo đức để thực hành nghề.

# Một bộ quy tắc hành vi đơn giản

* Tôn trọng người khác
* Phấn đấu vì sự công bằng
* Thực hiện khả năng tốt nhất của một người
* Theo pháp luật

# Nguyên tắc

* Khác với các ngành kỹ thuật khác như dân sự hoặc cơ khí, *không có một bộ nguyên tắc phổ quát nào trong kỹ nghệ phần mềm được mọi người đồng ý* .
* Có, tuy nhiên, một số được đón nhận và tôn trọng.
  + Nguyên tắc của Davis
  + Nguyên tắc của Royce
  + Các khái niệm của Wasserman

# Các nguyên tắc ban đầu của Davis 15

* *Làm cho số lượng chất lượng 1*
* *Phần mềm chất lượng cao là có thể*
* *Cung cấp sản phẩm cho khách hàng sớm*
* *Xác định vấn đề trước khi viết yêu cầu*
* *Đánh giá các lựa chọn thay thế thiết kế*
* *Sử dụng một mô hình quy trình thích hợp*
* *Sử dụng các ngôn ngữ khác nhau cho các giai đoạn khác nhau*
* *Giảm thiểu khoảng cách trí tuệ*
* *Đặt kỹ thuật trước công cụ*
* *Làm cho nó ngay trước khi bạn làm cho nó nhanh hơn*
* *Kiểm tra mã*
* *Quản lý tốt là quan trọng hơn công nghệ tốt*
* *Con người là chìa khóa để thành công*
* *Theo dõi cẩn thận*
* *Chịu trách nhiệm*
* Là những phù hợp trong bản thân mình - - -
* Cách “chìa khóa” là “con người là chìa khóa thành công”
* Bạn nghĩ sao?

# Các nguyên tắc hiện đại của Royce

* *Dựa trên quá trình tiếp cận kiến ​​trúc đầu tiên*
* *Thiết lập quy trình lặp lại --- rủi ro địa chỉ sớm*
* *Nhấn mạnh sự phát triển dựa trên thành phần để giảm bớt nỗ lực*
* *Thiết lập quản lý thay đổi*
* *Sử dụng kỹ thuật khứ hồi - một dạng của quá trình lặp lại*
* *Sử dụng ký pháp dựa trên mô hình và máy có thể xử lý cho thiết kế*
* *Thiết lập quy trình kiểm soát chất lượng và đánh giá dự án*
* *Sử dụng cách tiếp cận cho phép các hiện vật được chứng minh sớm*
* *Lên kế hoạch phát hành gia tăng*
* *Thiết lập quy trình có thể định cấu hình để phù hợp với nhu cầu*
* Đồng ý với những điều này? Tại sao ?

# Các khái niệm cơ bản của Wasserman

* *Trừu tượng*
* *Phương pháp và ký hiệu phân tích và thiết kế*
* *Tạo mẫu giao diện người dùng*
* *Modularity và kiến ​​trúc*
* *Tái sử dụng*
* *Vòng đời và quy trình*
* *Số liệu*
* *Công cụ và môi trường tích hợp*
* Các khái niệm quan trọng - - - làm thế nào để chúng liên quan đến các nguyên tắc được liệt kê trước đó từ Davis hoặc Royce?

# Làm thế nào về nguyên tắc hỗ trợ?

* Có cần phải đưa ra một số nguyên tắc hoặc quy tắc để hỗ trợ và duy trì phần mềm được phát hành không? Thế còn:
  + khách hàng luôn đúng
  + vấn đề độ phân giải xoay quanh thời gian là chìa khóa
  + vấn đề sửa chữa hoặc độ phân giải chất lượng là quan trọng
* *Bất kì thứ khác?* *Do tốc độ phân giải và chất lượng xung đột với nhau?*

CHƯƠNG 4

* **Chương 4**
* **Mô hình quy trình phần mềm**
* **Mùa thu 2011**

# Tại sao lại là mô hình Process?

* **Cung cấp hướng dẫn cho một sự phối hợp có hệ thống và kiểm soát các nhiệm vụ và của các nhân viên thực hiện các nhiệm vụ**
* ***Lưu ý các từ khóa: phối hợp, nhiệm vụ, con người***

# Mô hình quy trình

* **Xác định tập hợp các tác vụ cần được thực hiện**
* **Xác định đầu vào và đầu ra từ các tác vụ này**
* **Xác định trước điều kiện và sau điều kiện cho mỗi tác vụ**
* **Xác định chuỗi các nhiệm vụ**
* ***Có thể bao gồm mô tả về người thực hiện nó.***

# Quy trình phần mềm

* Quy trình phần mềm
* <số>
* Quy trình khác biệt với sản phẩm - sản phẩm là kết quả của việc thực hiện một quy trình trên một dự án
* SW Engineering tập trung vào quá trình
* Tiền đề: Các quy trình thích hợp sẽ giúp đạt được các mục tiêu dự án của QP cao

# Quy trình phần mềm…

* Quy trình phần mềm
* <số>
* Quy trình: Một phương pháp cụ thể, thường liên quan đến một số bước
* Quá trình phần mềm: Một bộ các bước, cùng với các ràng buộc đặt hàng trên thực thi, để tạo ra phần mềm với kết quả mong muốn
* Nhiều loại hoạt động được thực hiện bởi những người khác nhau trong một dự án phần mềm
* Tốt hơn để xem quá trình phần mềm bao gồm nhiều quá trình thành phần

# Quy trình phần mềm thành phần

* Quy trình phần mềm
* <số>
* Hai quy trình chính
  + Phát triển - tập trung vào các bước phát triển và chất lượng cần thiết để thiết kế phần mềm
  + Quản lý dự án - tập trung vào lập kế hoạch và kiểm soát quá trình phát triển
* Quá trình phát triển là trung tâm của quá trình phần mềm; các quy trình khác xoay quanh nó
* Chúng được thực hiện bởi những người khác nhau
  + các nhà phát triển thực hiện kỹ thuật. Quá trình
  + quản lý dự án thực hiện các quy trình quản lý

# Quy trình thành phần…

* Quy trình phần mềm
* <số>
* Các quy trình khác
  + Quy trình quản lý cấu hình: quản lý sự tiến hóa của các hiện vật
  + Thay đổi quy trình quản lý: các thay đổi được kết hợp như thế nào
  + Quy trình quản lý quy trình: quản lý các quy trình
  + Quy trình kiểm tra: Cách tiến hành kiểm tra trên hiện vật

# Quy trình kỹ thuật

* Quy trình phần mềm
* <số>
* Quy trình thường là một tập hợp các giai đoạn
* Mỗi giai đoạn thực hiện một nhiệm vụ được xác định rõ và thường tạo ra một đầu ra
* Đầu ra trung gian - *sản phẩm công việc*
* Ở cấp cao nhất, thường là một vài giai đoạn trong quá trình
* Làm thế nào để thực hiện một giai đoạn cụ thể - *phương pháp* đã được đề xuất

# Thuộc tính quy trình mong muốn

* Quy trình phần mềm
* <số>
* Cung cấp Q & P cao
  + Hỗ trợ testability như thử nghiệm là nhiệm vụ đắt nhất; thử nghiệm có thể tiêu thụ từ 30 đến 50% tổng nỗ lực phát triển
  + Hỗ trợ bảo trì như bảo trì có thể đắt hơn phát triển; vượt quá 80% tổng chi phí
  + Loại bỏ các khuyết tật sớm, vì chi phí loại bỏ các khuyết tật tăng lên với độ trễ

# Q & P cao: Loại bỏ lỗi sớm ...

* Quy trình phần mềm
* <số>
* Chi phí của một lỗi tăng lên với độ trễ
* Tức là sửa chữa một lỗi yêu cầu trong hoạt động có thể tốn 100 lần chi phí sửa chữa nó trong các yêu cầu
* Do đó, đối với Q & P cao, quy trình phải hỗ trợ loại bỏ lỗi sớm
* Đó là lý do tại sao có một V trong ETVX (Entry, Task, Verification, Exit), và nhiệm vụ kiểm soát chất lượng trong quá trình sw

# Xóa lỗi sớm ...

* Quy trình phần mềm
* <số>

# Thuộc tính mong muốn…

* Quy trình phần mềm
* <số>
* Dự đoán và lặp lại
  + Quy trình nên lặp lại hiệu suất của nó khi được sử dụng trên các dự án khác nhau
  + Tức là kết quả của việc sử dụng một quá trình có thể dự đoán được
  + Không dự đoán được, không thể ước tính hoặc nói bất cứ điều gì về chất lượng hoặc năng suất
  + Với khả năng dự đoán, hiệu suất trong quá khứ có thể được sử dụng để dự đoán hiệu suất trong tương lai

# Dự đoán…

* Quy trình phần mềm
* <số>
* Quy trình dự đoán được cho là dưới sự kiểm soát thống kê
  + Lặp đi lặp lại bằng cách sử dụng quá trình tạo ra kết quả tương tự
  + Kết quả - tính chất quan tâm như chất lượng, năng suất,…
* Để liên tục phát triển sw với Q & P cao, quá trình phải được kiểm soát

# Dự đoán…

* Quy trình phần mềm
* <số>

# Hỗ trợ thay đổi

* Quy trình phần mềm
* <số>
* Thay đổi phần mềm vì nhiều lý do
* Yêu cầu thay đổi là một lý do chính
* Những thay đổi yêu cầu không thể được mong ước hoặc coi là "xấu"
* Họ phải được lưu trú trong quá trình phát triển sw

# Tóm lược

* Quy trình phần mềm
* <số>
* Quy trình - phương pháp để làm điều gì đó
* Quy trình thường có các giai đoạn, mỗi giai đoạn tập trung vào một nhiệm vụ có thể nhận dạng
* Các giai đoạn có phương pháp luận
* Quá trình phần mềm là phương pháp phát triển phần mềm
* Tốt nhất để xem nó như bao gồm nhiều quy trình

# Tóm lược

* Quy trình phần mềm
* <số>
* Mục tiêu là tạo ra phần mềm với chất lượng và năng suất cao
* Quy trình là phương tiện
* Quá trình phát triển là quá trình trung tâm
* Quá trình Mgmt là để kiểm soát dev
* Các quy trình hỗ trợ khác
* Quá trình Sw phải có Q & P cao, khả năng dự đoán và hỗ trợ thay đổi

# Mô hình quy trình và quy trình phát triển

* Quy trình phần mềm
* <số>

# Dự án phần mềm

* Quy trình phần mềm
* <số>
* Dự án - xây dựng một hệ thống phần mềm trong phạm vi chi phí và tiến độ và với chất lượng cao, thỏa mãn khách hàng
* Mục tiêu của dự án - cao Q và P cao
* Quy trình phù hợp cần thiết để đạt được mục tiêu
* Đối với một dự án, quy trình được theo dõi được quy định cụ thể trong quá trình lập kế hoạch

# Quá trình phát triển

* Quy trình phần mềm
* <số>
* Một tập hợp các giai đoạn và mỗi giai đoạn là một chuỗi các bước
* Trình tự các bước cho một pha - phương pháp luận cho giai đoạn đó.
* Tại sao có các giai đoạn
  + Sử dụng phân chia và chinh phục
  + mỗi giai đoạn xử lý một phần khác nhau của vấn đề
  + giúp xác nhận liên tục

# Quá trình phát triển

* Quy trình phần mềm
* <số>
* Thường có các hoạt động này:
  + Phân tích yêu cầu
  + Kiến trúc
  + Thiết kế
  + Mã hóa
  + Thử nghiệm
  + Chuyển
* Các mô hình khác nhau thực hiện chúng theo cách khác nhau

# Phân tích yêu cầu

* Quy trình phần mềm
* <số>
* Để hiểu và nêu chính xác vấn đề
* Hình thành cơ sở thỏa thuận giữa người dùng và nhà phát triển
* chỉ rõ “ **cái gì** ”, không phải “ **làm thế nào** ”.
* Không phải là một nhiệm vụ dễ dàng, vì nhu cầu thường không được hiểu.
* Yêu cầu thông số kỹ thuật của các hệ thống trung bình thậm chí có thể là hàng trăm trang
* Đầu ra là tài liệu Đặc tả Yêu cầu Phần mềm (SRS)

# Thiết kế

* Quy trình phần mềm
* <số>
* Một bước tiến quan trọng trong việc chuyển từ miền vấn đề sang miền giải pháp; ba nhiệm vụ chính
  + Thiết kế kiến ​​trúc - các thành phần và các đầu nối cần có trong hệ thống
  + Thiết kế cấp cao - mô-đun và cấu trúc dữ liệu cần thiết để triển khai kiến ​​trúc
  + Thiết kế chi tiết - logic của mô-đun
* Hầu hết các phương pháp tập trung vào kiến ​​trúc hoặc thiết kế mức cao
* Các đầu ra là các tài liệu thiết kế kiến ​​trúc / thiết kế / logic

# Mã hóa

* Quy trình phần mềm
* <số>
* Chuyển đổi thiết kế thành mã theo ngôn ngữ cụ thể
* Mục tiêu: Thực hiện thiết kế với mã đơn giản và dễ hiểu.
  + Mã nên đơn giản và dễ đọc.
* Giai đoạn mã hóa ảnh hưởng đến cả thử nghiệm và bảo trì. Mã được viết tốt có thể làm giảm nỗ lực kiểm tra và bảo trì.
* Đầu ra là mã

# Thử nghiệm

* Quy trình phần mềm
* <số>
* Khuyết tật được giới thiệu trong từng giai đoạn
* Họ phải được tìm thấy và loại bỏ để đạt được chất lượng cao
* Thử nghiệm đóng vai trò quan trọng này
* Mục tiêu: Xác định hầu hết các khuyết tật
* Là một nhiệm vụ rất tốn kém; phải được lên kế hoạch và thực hiện đúng cách.
* Các đầu ra là các kế hoạch / kết quả thử nghiệm và mã thử nghiệm cuối cùng (hy vọng đáng tin cậy)

# Phân phối nỗ lực

* Quy trình phần mềm
* <số>
* **Phân phối nỗ lực:**
  + Yêu cầu 10-20%
  + Thiết kế 10-20%
  + Mã hóa 20-30%
  + Thử nghiệm 30-50%
* Mã hóa không phải là đắt nhất.

# Phân phối nỗ lực…

* Quy trình phần mềm
* <số>
* Cách người lập trình dành thời gian của họ
  + Viết chương trình 13%
  + Chương trình đọc và hướng dẫn sử dụng 16%
  + Giao tiếp công việc 32%
  + Khác 39%
* Các lập trình viên dành nhiều thời gian hơn trong việc đọc các chương trình hơn là viết chúng.
* Viết chương trình là một phần nhỏ trong cuộc sống của họ.

# Khiếm khuyết

* Quy trình phần mềm
* <số>
* Phân phối các lần xuất hiện lỗi theo pha
  + Req. - 20%
  + Thiết kế - 30%
  + Mã hóa - 50%
* Khuyết tật có thể được tiêm vào bất kỳ giai đoạn chính nào.
* Chi phí thời gian chờ: Chi phí xóa bỏ lỗi tăng theo cấp số nhân với thời gian chờ.

# Khiếm khuyết…

* Quy trình phần mềm
* <số>
  + - * Chi phí sửa chữa
      * Lỗi (thang đo log)
      * Thời gian
* Cách rẻ nhất là phát hiện và loại bỏ các khuyết tật gần nơi nó được tiêm.
* Do đó phải kiểm tra lỗi sau mỗi pha.

# Mô hình quy trình

* Quy trình phần mềm
* <số>
* Mô hình quy trình chỉ định quy trình chung, thường là một tập hợp các giai đoạn
* Mô hình này sẽ phù hợp với một loại dự án
* Tức là một mô hình cung cấp cấu trúc chung của quá trình có thể được theo sau bởi một số dự án để đạt được mục tiêu của họ

# Quy trình dự án

* Quy trình phần mềm
* <số>
* Nếu một dự án chọn một mô hình, nó sẽ thường thay đổi nó cho phù hợp với dự án
* Điều này tạo ra thông số kỹ thuật cho quy trình dự án
* Quá trình này có thể được theo sau trong dự án
* Tức là quá trình thực sự được thực thi; quy trình spec là kế hoạch về những gì nên được thực thi; mô hình quy trình là một đặc tả quy trình chung
* Nhiều mô hình đã được đề xuất cho quá trình phát triển

# Mô hình quy trình sinh viên điển hình

* Quy trình phần mềm
* <số>
* Nhận được vấn đề stmt - Code - làm một số thử nghiệm - cung cấp / demo
* Tại sao mô hình quy trình này không thể được sử dụng cho các dự án thương mại?
  + Sản xuất phần mềm sinh viên, không phải là những gì chúng tôi đang theo đuổi
  + Không thể đảm bảo chất lượng mong muốn cho phần mềm có độ bền công nghiệp

# Các mô hình quy trình chung

* Quy trình phần mềm
* <số>
* Thác nước - lâu đời nhất và được sử dụng rộng rãi
* Tạo mẫu
* Lặp lại - hiện được sử dụng rộng rãi
* Timeboxing

# Một quy trình đơn giản và quen thuộc

* **Mã số**
* **Biên dịch**
* **Gỡ lỗi**
* **Vấn đề**
* **Tuyên bố**
* **Giải phóng**
* **vấn đề**
* **vấn đề**
* **Kiểm tra đơn vị**
* ***1. Hầu hết mọi người thực hiện và làm theo quy trình này, nhưng***
* ***tiếc là một số*** *thử nghiệm đơn vị* bỏ qua *hoặc gỡ lỗi.*
* ***2. Ngoài ra, một số tiến hành mà không hiểu rõ "tuyên bố vấn đề" ---- đó là yêu cầu***

# Một số quy trình phát triển phần mềm “truyền thống”

* **Quy trình “đơn giản” được nhiều người sử dụng trong nhiều năm mà không chính thức bao gồm các hoạt động phát triển quan trọng khác như phân tích yêu cầu, thiết kế, thử nghiệm chính thức hoặc đóng gói.**
* **Việc công nhận sự cần thiết cho các quy trình chính thức ban đầu được điều khiển bởi những thất bại trong việc phát triển phần mềm phức tạp lớn**
  + **Thác nước: *quá trình sớm nhất và đối phó với không có quá trình***
  + **Gia tăng: *đối phó với việc phân hủy các hệ thống lớn***
  + **Xoắn ốc: *đối phó với quản lý rủi ro***
  + **Rational Unified Process: *đối phó với nhiều vấn đề phát triển và quản lý***

# Mô hình thác nước

* **Yêu cầu**
* **Thiết kế**
* **Mã số**
* **Kiểm tra**
* **Tích hợp và**
* **Gói**
* **1. Yêu cầu phải được chỉ định trong**
* **bước đầu tiên.**
* **2. Bốn nhiệm vụ chính phải được hoàn thành trong**
* **trình tự: *yêu cầu* , *thiết kế* , *mã* ,**
* **và *thử nghiệm,* tiếp theo là đóng gói.**
* **3. Đầu ra của một giai đoạn nguồn cấp dữ liệu vào tiếp theo**
* **theo thứ tự, và do đó dễ dàng**
* **được theo dõi bởi ban quản lý**

# Mô hình gia tăng A - Tích hợp liên tục

* **Req.** **1**
* **Req.2**
* **Req.** **n**
* **Des.**
* **Des.**
* **Des.**
* **mã**
* **Thùng tích hợp**
* **Kiểm tra**
* **mã**
* **mã**
* **Kiểm tra**
* **Kiểm tra**
* **.** **.** **.**
* **.** **.** **.** **.**
* **.** **.** **.** **.** **.** **.**
* **Hệ thống**
* **Kiểm tra**
* **Mỗi yêu cầu / mục chính**
* **được phát triển riêng thông qua**
* **cùng một chuỗi: yêu cầu ,**
* **thiết kế , mã và kiểm tra đơn vị .**
* **Khi các phần phát triển được hoàn thành,**
* **chúng được liên tục hợp nhất và**
* **được tích hợp vào một nhóm chung cho**
* **kiểm tra hệ thống tích hợp**

# Mô hình gia tăng B - Nhiều bản phát hành

* **Yêu cầu**
* **Thiết kế**
* **Mã số**
* **Kiểm tra**
* **Gói**
* ***Rel.*** ***1***
* **Yêu cầu**
* **Thiết kế**
* **Mã số**
* **Kiểm tra**
* **Gói**
* ***Rel.*** ***n***
* **.**
* **.**
* **.**
* **Mỗi bộ yêu cầu nhỏ được phát triển,**
* **đóng gói và phát hành trong một bản phát hành đa**
* **thời trang.**

# Mô hình xoắn ốc

* **Kế hoạch giai đoạn tiếp theo**
* **Phát triển, Xác minh**
* **Sản phẩm cấp độ tiếp theo**
* **Xác định mục tiêu,**
* **Giải pháp thay thế, Ràng buộc**
* **Đánh giá các lựa chọn thay thế,**
* **Xác định, giải quyết rủi ro**
* **req.**
* **Spec.**
* **thiết kế**
* **mô hình**
* **thiết kế**
* **xác thực**
* **kế hoạch kiểm tra**
* **mã**
* **đơn vị**
* **kiểm tra**
* **kế hoạch phát triển**
* **req.** **kế hoạch**
* **rủi ro**
* **phân tích**
* **proto**
* **kiểu**
* **thiết kế**
* **sys.**
* **kiểm tra**
* ***"Ôn tập"***
* **- Phát triển phần mềm**
* **các hoạt động được luân chuyển qua**
* **4 giai đoạn**
* **- Một quá trình chống lại rủi ro**

# Mô hình xoắn ốc

* Xác định các mục tiêu, lựa chọn thay thế hoặc ràng buộc cho mỗi chu kỳ xoắn ốc.
* Đánh giá các lựa chọn thay thế liên quan đến mục tiêu và ràng buộc. Khi thực hiện bước này, nhiều rủi ro được xác định và đánh giá.
* Tùy thuộc vào số lượng và loại rủi ro được xác định, phát triển mẫu thử nghiệm, đánh giá chi tiết hơn, phát triển tiến hóa hoặc một số bước khác để giảm thêm rủi ro đạt được mục tiêu đã xác định. Mặt khác, nếu rủi ro giảm đáng kể, bước tiếp theo có thể chỉ là một nhiệm vụ như yêu cầu, thiết kế hoặc mã.
* Xác nhận thành tích của mục tiêu và kế hoạch cho chu kỳ tiếp theo.

# -Quy trình hợp nhất hợp lý

* **Các giai đoạn**
* ***Khởi đầu***
* ***Xây dựng***
* ***Xây dựng***
* ***Chuyển đổi***
* ***Yêu cầu***
* ***Thiết kế***
* ***Triển khai thực hiện***
* ***Kiểm tra***
* ***Tích hợp***
* **Hoạt động**
* **Mọi phát triển phần mềm**
* **hoạt động đi qua 4**
* **giai đoạn khởi đầu , xây dựng ,**
* **xây dựng** và chuyển đổi

# Tiêu chí xuất nhập cảnh

* **Quá trình**
* **Hoạt động**
* **Entry**
* **Tiêu chí**
* **Gặp?**
* **Lối thoát**
* **Tiêu chí**
* **Gặp?**
* **Vâng**
* **Vâng**
* **Không**
* **Không**
* **Để các mô hình quy trình không chỉ đơn thuần là**
* **"hướng dẫn", nó phải bao gồm một danh sách các điều kiện hoặc**
* **yêu cầu xác định:**
* **- tiêu chí nhập cảnh trước khi biểu diễn**
* **một hoạt động trong một quá trình.**
* **- tiêu chí thoát trước khi hoạt động trong**
* **quá trình được coi là hoàn thành.**

# Đánh giá quy trình

* **Phát triển phần mềm và hỗ trợ phần mềm có thể được thực hiện với quá trình rất ít hoặc với các quy trình rất tinh vi, được xác định rõ, được tổ chức tốt và được thực thi tốt.**
* **Tổ chức kỹ nghệ phần mềm của bạn trưởng thành như thế nào và bạn có cần cải thiện không?**
* **ISO (ISO 9000 series) và SEI (tại Carnegie Mellon) là hai tổ chức hàng đầu giúp đánh giá quá trình**
* ***Không có quy trình***
* ***Quy trình trưởng thành***
* ***Bạn đang ở đâu***
* ***phổ rộng này?***

# SEI CMM

* **Viện Kỹ thuật phần mềm (SEI) đã đề xuất một Mô hình trưởng thành năng lực (CMM) để giúp các tổ chức phần mềm *đánh giá* *sự trưởng thành của họ* và *đưa ra hướng dẫn* trong phát triển phần mềm.**
  + **Ban đầu** : không có quy trình và thành công nào là do may mắn hoặc với một người đặc biệt.
  + **Lặp lại** : đã làm chủ được 6 quy trình và có thể lặp lại thành công với 6 quy trình này: 1) yêu cầu mgmt, 2) theo dõi dự án, 3) đảm bảo chất lượng, 4) lập kế hoạch dự án, 5) quản lý cấu hình phụ và 6) quản lý cấu hình
  + **Xác định** : đã làm chủ được 7 quy trình và có thẩm quyền xây dựng phần mềm: 1) quy trình tổ chức, 2) chương trình đào tạo, 3) kỹ thuật sản phẩm, 4) đánh giá ngang hàng, 5) định nghĩa quy trình tổ chức, 6) tích hợp mềm. mgmt và 7) phối hợp giữa các nhóm
  + **Được quản lý** : đã giới thiệu thêm 2 quy trình xử lý đo lường và chất lượng định lượng: 1) quản lý quy trình định lượng và 2) chất lượng mgmt
  + **Tối ưu hóa** : đạt đến mức cao nhất này đòi hỏi phải làm chủ cải tiến liên tục với 3 quy trình khác: 1) phòng ngừa khiếm khuyết, 2) quản lý thay đổi công nghệ, 3) quản lý thay đổi quy trình

# 5 cấp độ “Mô hình trưởng thành khả năng” ban đầu (CMM)

* **Ban đầu**
* **Lặp lại**
* **Được định nghĩa**
* **Được quản lý**
* **Tối ưu hóa**
* **Cấp độ 1**
* **Cấp 4**
* **Cấp 3**
* **Cấp độ 2**
* **Cấp 5**
* ***Ít trưởng thành nhất***
* ***Hầu hết người trưởng thành***
* **Tổng số 18 quy trình cần phải được làm chủ để đạt được mức "tối ưu"**

# SEI CMMI

* **Năm 2001, CMM đã được nâng cấp lên CMMI (CMM tích hợp).** **Có những khía cạnh chính yếu đối với CMMI:**
  + **Kỹ thuật hệ thống**
  + **Kỹ thuật phần mềm**
  + **Tích hợp sản phẩm và quy trình phát triển**
  + **Tìm nguồn cung ứng**
* **Phần kỹ thuật phần mềm của CMMI có hai biểu diễn:**
  + **Được tổ chức** : tương tự như đánh giá CMM của tổ chức
  + **Liên tục** : tốt hơn để đánh giá và cải thiện sự trưởng thành của từng quy trình

# Quy trình của CMMI

* **Có 25 quy trình bao gồm 4 loại chính:**
  + **Quản lý quy trình (có 5 quy trình):**
    - **Tập trung quá trình tổ chức**
    - **Định nghĩa quy trình tổ chức**
    - **Đào tạo tổ chức**
    - **Hiệu suất quy trình tổ chức**
    - **Tổ chức đổi mới và triển khai**
  + **Quản lý dự án (có 8 quy trình):**
    - **Lập kế hoạch dự án**
    - **Giám sát và kiểm soát dự án**
    - **Quản lý thỏa thuận nhà cung cấp**
    - **Quản lý dự án tích hợp**
    - **Quản lý rủi ro**
    - **Tích hợp teaming**
    - **Quản lý nhà cung cấp tích hợp**
    - **Quản lý dự án định lượng**

# Quy trình của CMMI (cont,)

* **Kỹ thuật (có 6 quy trình)**
  + **Yêu cầu phát triển**
  + **Quản lý yêu cầu**
  + **Giải pháp kỹ thuật**
  + **Tích hợp sản phẩm**
  + **xác minh**
  + **Xác nhận**
* **Hỗ trợ (có 6 quy trình)**
  + **Quản lý cấu hình**
  + **Quy trình và đảm bảo chất lượng sản phẩm**
  + **Đo lường và phân tích**
  + **Môi trường tổ chức để tích hợp**
  + **Phân tích và phân tích quyết định**
  + **Phân tích và phân tích nhân quả**

# Cấp độ cho các mô hình liên tục và được tổ chức trong CMM I

* **Cấp 5**
* **Cấp 4**
* **Cấp 3**
* **Cấp độ 2**
* **Cấp độ 1**
* **Cấp 0**
* **Liên tiếp**
* **( *Mức độ khả năng* )**
* **Đã tổ chức**
* **( *Mức độ trưởng thành* )**
* **chưa hoàn thiện**
* **Đã thực hiện**
* **Được quản lý**
* **Được định nghĩa**
* **Quản lý định lượng**
* **Tối ưu hóa**
* **Quản lý định lượng**
* **Ban đầu**
* **Được quản lý**
* **Được định nghĩa**
* **Tối ưu hóa**
* **- - - - - -**

# Mô hình liên tục và được tổ chức

* **Trong biểu diễn liên tục, mỗi quá trình bắt đầu ở mức năng lực 0 và nâng cao các mức năng lực dựa trên việc đạt được “mục tiêu chung” và “các mục tiêu phụ cụ thể”.**
  + **Cho phép tổ chức chọn và chọn quy trình để tập trung vào dựa trên nhu cầu của tổ chức**
  + **Cho phép so sánh khu vực quy trình theo khu vực quy trình giữa các tổ chức**
  + **Cho phép di chuyển dễ dàng hơn từ các tiêu chuẩn khác**
* **Trong biểu diễn theo giai đoạn, tổ chức bắt đầu ở mức trưởng thành 1 và di chuyển lên các cấp dựa trên việc nắm vững các quy trình.**
  + **Cho phép di chuyển dễ dàng từ mô hình CMM trước đó**
  + **Cung cấp hướng dẫn về trình tự trưởng thành theo khu vực quy trình**
  + **Cho phép so sánh dễ dàng hơn các tổ chức theo mức độ trưởng thành**

# Mối quan hệ của mục tiêu và thực tiễn

* **Quá trình**
* **Khu vực 1**
* **Quá trình**
* **Khu vực 25**
* **- - - - - - - - - - - - - - - - -**
* **Riêng**
* **Mục tiêu 1**
* **Riêng**
* **Mục tiêu x**
* **- - - -**
* **Riêng**
* **Thực hành 1**
* **Riêng**
* **Thực hành w**
* **- - -**
* **Riêng**
* **Mục tiêu 1**
* **- - - -**
* **Riêng**
* **Mục tiêu z**
* **Chung**
* **Mục tiêu 1**
* **Chung**
* **Mục tiêu 5**
* **- - - -**
* **Chung**
* **Thực hành 1**
* **Chung**
* **Thực hành n**
* **Chung**
* **Thực hành 1**
* **- - - -**
* **- - - -**
* **Riêng**
* **Thực hành 1**
* **Riêng**
* **Thực hành k**
* **Chung**
* **Thực hành p**
* **- - -**

# Đạt được “ *Cấp độ khả năng”* của từng Vùng quy trình trong Mô hình đại diện liên tục

* **chưa hoàn thiện**
* **Đã thực hiện**
* **Được quản lý**
* **Được định nghĩa**
* **Quản lý định lượng**
* **Tối ưu hóa**
* **CL0**
* **CL1**
* **CL2**
* **CL3**
* **CL4**
* **CL5**
* ***+ (Mục tiêu cụ thể)***
* ***+ (Mục tiêu chung 1)***
* ***+ (Mục tiêu chung 2)***
* ***+ (Mục tiêu chung 3)***
* ***+ (Mục tiêu chung 4)***
* ***+ (Mục tiêu chung 5)***

# Đạt được *“Độ trưởng thành”* (ML) trong mô hình Đại diện theo giai đoạn

* **ML1 (0 quy trình): không có quy trình**
* **ML2 (7 quy trình): 1) Yêu cầu Mgmt, 2) Lập kế hoạch dự án, 3) Theo dõi dự án, 4) Thỏa thuận nhà cung cấp, 5) Đo lường và phân tích, 6) Đảm bảo chất lượng sản phẩm và quy trình, 7) Cấu hình mgmt**
* **ML3 (14 quy trình): 1) Yêu cầu phát triển, 2) Giải pháp kỹ thuật, 3) Tích hợp sản phẩm, 4) Xác minh, 5) Xác nhận, 6) Tập trung tổ chức, 7) Định nghĩa quy trình tổ chức, 8) Đào tạo tổ chức, 9) quản lý, 10) Quản lý rủi ro, 11) Tích hợp nhóm, 12) Tích hợp nhà cung cấp mgmt, 13) Phân tích và giải quyết quyết định, 14) Môi trường tổ chức để tích hợp**
* **ML4 (2 quy trình): 1) Hiệu suất quy trình tổ chức, 2) Quản lý dự án định lượng**
* **ML5 (2 quy trình): 1) Đổi mới và triển khai tổ chức, 2) Phân tích và giải quyết nhân quả**

# Quy trình và định nghĩa quy trình

* **2 thành phần chính của quá trình định nghĩa:**
  + **Các hoạt động chính**
  + **Sắp xếp các hoạt động**
* **Hầu hết các tổ chức cần phải sửa đổi một quy trình hiện có để phù hợp hơn với nhu cầu của họ ----- do đó họ phải xác định chi tiết hơn và truyền đạt các định nghĩa quy trình đã sửa đổi (một nỗ lực lớn)**
* **Mở rộng định nghĩa quy trình thành cấp độ "tinh tế" hơn:**
  + **Mô tả chi tiết các *hoạt động***
  + ***Kiểm soát*** cần thiết cho *lối vào* và *lối ra* của từng hoạt động và thứ tự của các hoạt động
  + ***Các tạo phẩm*** phát sinh từ các hoạt động
  + ***Nguồn nhân lực*** cần thiết để thực hiện các hoạt động
  + ***Các công cụ*** có thể cần thiết để hỗ trợ hoạt động của các hoạt động

# Đạt được *“Độ trưởng thành”* (ML) trong mô hình Đại diện theo giai đoạn

* **ML1 (0 quy trình): không có quy trình**
* **ML2 (7 quy trình): Khu vực xử lý**
  + **1) Yêu cầu Mgmt, Kỹ thuật**
  + **2) Dự án lập kế hoạch dự án**
  + **3) Dự án giám sát Dự án Mgmt**
  + **4) Thỏa thuận nhà cung cấp Mgmt Project Mgmt**
  + **5) Hỗ trợ đo lường và phân tích**
  + **6) Quy trình và đảm bảo chất lượng sản phẩm**
  + **7) Cấu hình mgmt Hỗ trợ**

# Đạt được *“Độ trưởng thành”* (ML) trong mô hình Đại diện theo giai đoạn

* **ML3 (14 quy trình): Khu vực xử lý**
  + **1) Yêu cầu phát triển Kỹ thuật**
  + **2) Kỹ thuật giải pháp kỹ thuật**
  + **3) Kỹ thuật tích hợp sản phẩm**
  + **4) Kỹ thuật xác minh**
  + **5) Kỹ thuật xác nhận**
  + **6) Quá trình tổ chức tập trung quá trình Mgmt**
  + **7) Quy trình tổ chức định nghĩa Quy trình Mgmt**
  + **8) Quy trình đào tạo tổ chức Mgmt**
  + **9) Quản lý dự án tích hợp Dự án Mgmt**
  + **10) Quản lý rủi ro**
  + **11) Tích hợp Team Mgmt dự án**
  + **12) Nhà cung cấp Mgmt dự án tích hợp**
  + **13) Phân tích và phân tích quyết định**
  + **14) Môi trường tổ chức để hỗ trợ tích hợp**

# Đạt được *“Độ trưởng thành”* (ML) trong mô hình Đại diện theo giai đoạn

* **ML4 (2 quy trình): Khu vực xử lý**
  + **1) Quy trình tổ chức thực hiện Quy trình Mgmt**
  + **2) Dự án quản lý dự án định lượng Mgmt**
* **ML5 (2 quy trình): Khu vực xử lý**
  + **1) Đổi mới tổ chức và triển khai Quy trình Mgmt**
  + **2) Phân tích và phân tích nhân quả**

CHƯƠNG 4

* **Chương 4**
* **Mô hình quy trình phần mềm**
* **Mùa thu 2011**

# Tại sao lại là mô hình Process?

* **Cung cấp hướng dẫn cho một sự phối hợp có hệ thống và kiểm soát các nhiệm vụ và của các nhân viên thực hiện các nhiệm vụ**
* ***Lưu ý các từ khóa: phối hợp, nhiệm vụ, con người***

# Mô hình quy trình

* **Xác định tập hợp các tác vụ cần được thực hiện**
* **Xác định đầu vào và đầu ra từ các tác vụ này**
* **Xác định trước điều kiện và sau điều kiện cho mỗi tác vụ**
* **Xác định chuỗi các nhiệm vụ**
* ***Có thể bao gồm mô tả về người thực hiện nó.***

# Quy trình phần mềm

* Quy trình phần mềm
* <số>
* Quy trình khác biệt với sản phẩm - sản phẩm là kết quả của việc thực hiện một quy trình trên một dự án
* SW Engineering tập trung vào quá trình
* Tiền đề: Các quy trình thích hợp sẽ giúp đạt được các mục tiêu dự án của QP cao

# Quy trình phần mềm…

* Quy trình phần mềm
* <số>
* Quy trình: Một phương pháp cụ thể, thường liên quan đến một số bước
* Quá trình phần mềm: Một bộ các bước, cùng với các ràng buộc đặt hàng trên thực thi, để tạo ra phần mềm với kết quả mong muốn
* Nhiều loại hoạt động được thực hiện bởi những người khác nhau trong một dự án phần mềm
* Tốt hơn để xem quá trình phần mềm bao gồm nhiều quá trình thành phần

# Quy trình phần mềm thành phần

* Quy trình phần mềm
* <số>
* Hai quy trình chính
  + Phát triển - tập trung vào các bước phát triển và chất lượng cần thiết để thiết kế phần mềm
  + Quản lý dự án - tập trung vào lập kế hoạch và kiểm soát quá trình phát triển
* Quá trình phát triển là trung tâm của quá trình phần mềm; các quy trình khác xoay quanh nó
* Chúng được thực hiện bởi những người khác nhau
  + các nhà phát triển thực hiện kỹ thuật. Quá trình
  + quản lý dự án thực hiện các quy trình quản lý

# Quy trình thành phần…

* Quy trình phần mềm
* <số>
* Các quy trình khác
  + Quy trình quản lý cấu hình: quản lý sự tiến hóa của các hiện vật
  + Thay đổi quy trình quản lý: các thay đổi được kết hợp như thế nào
  + Quy trình quản lý quy trình: quản lý các quy trình
  + Quy trình kiểm tra: Cách tiến hành kiểm tra trên hiện vật

# Quy trình kỹ thuật

* Quy trình phần mềm
* <số>
* Quy trình thường là một tập hợp các giai đoạn
* Mỗi giai đoạn thực hiện một nhiệm vụ được xác định rõ và thường tạo ra một đầu ra
* Đầu ra trung gian - *sản phẩm công việc*
* Ở cấp cao nhất, thường là một vài giai đoạn trong quá trình
* Làm thế nào để thực hiện một giai đoạn cụ thể - *phương pháp* đã được đề xuất

# Thuộc tính quy trình mong muốn

* Quy trình phần mềm
* <số>
* Cung cấp Q & P cao
  + Hỗ trợ testability như thử nghiệm là nhiệm vụ đắt nhất; thử nghiệm có thể tiêu thụ từ 30 đến 50% tổng nỗ lực phát triển
  + Hỗ trợ bảo trì như bảo trì có thể đắt hơn phát triển; vượt quá 80% tổng chi phí
  + Loại bỏ các khuyết tật sớm, vì chi phí loại bỏ các khuyết tật tăng lên với độ trễ

# Q & P cao: Loại bỏ lỗi sớm ...

* Quy trình phần mềm
* <số>
* Chi phí của một lỗi tăng lên với độ trễ
* Tức là sửa chữa một lỗi yêu cầu trong hoạt động có thể tốn 100 lần chi phí sửa chữa nó trong các yêu cầu
* Do đó, đối với Q & P cao, quy trình phải hỗ trợ loại bỏ lỗi sớm
* Đó là lý do tại sao có một V trong ETVX (Entry, Task, Verification, Exit), và nhiệm vụ kiểm soát chất lượng trong quá trình sw

# Xóa lỗi sớm ...

* Quy trình phần mềm
* <số>

# Thuộc tính mong muốn…

* Quy trình phần mềm
* <số>
* Dự đoán và lặp lại
  + Quy trình nên lặp lại hiệu suất của nó khi được sử dụng trên các dự án khác nhau
  + Tức là kết quả của việc sử dụng một quá trình có thể dự đoán được
  + Không dự đoán được, không thể ước tính hoặc nói bất cứ điều gì về chất lượng hoặc năng suất
  + Với khả năng dự đoán, hiệu suất trong quá khứ có thể được sử dụng để dự đoán hiệu suất trong tương lai

# Dự đoán…

* Quy trình phần mềm
* <số>
* Quy trình dự đoán được cho là dưới sự kiểm soát thống kê
  + Lặp đi lặp lại bằng cách sử dụng quá trình tạo ra kết quả tương tự
  + Kết quả - tính chất quan tâm như chất lượng, năng suất,…
* Để liên tục phát triển sw với Q & P cao, quá trình phải được kiểm soát

# Dự đoán…

* Quy trình phần mềm
* <số>

# Hỗ trợ thay đổi

* Quy trình phần mềm
* <số>
* Thay đổi phần mềm vì nhiều lý do
* Yêu cầu thay đổi là một lý do chính
* Những thay đổi yêu cầu không thể được mong ước hoặc coi là "xấu"
* Họ phải được lưu trú trong quá trình phát triển sw

# Tóm lược

* Quy trình phần mềm
* <số>
* Quy trình - phương pháp để làm điều gì đó
* Quy trình thường có các giai đoạn, mỗi giai đoạn tập trung vào một nhiệm vụ có thể nhận dạng
* Các giai đoạn có phương pháp luận
* Quá trình phần mềm là phương pháp phát triển phần mềm
* Tốt nhất để xem nó như bao gồm nhiều quy trình

# Tóm lược

* Quy trình phần mềm
* <số>
* Mục tiêu là tạo ra phần mềm với chất lượng và năng suất cao
* Quy trình là phương tiện
* Quá trình phát triển là quá trình trung tâm
* Quá trình Mgmt là để kiểm soát dev
* Các quy trình hỗ trợ khác
* Quá trình Sw phải có Q & P cao, khả năng dự đoán và hỗ trợ thay đổi

# Mô hình quy trình và quy trình phát triển

* Quy trình phần mềm
* <số>

# Dự án phần mềm

* Quy trình phần mềm
* <số>
* Dự án - xây dựng một hệ thống phần mềm trong phạm vi chi phí và tiến độ và với chất lượng cao, thỏa mãn khách hàng
* Mục tiêu của dự án - cao Q và P cao
* Quy trình phù hợp cần thiết để đạt được mục tiêu
* Đối với một dự án, quy trình được theo dõi được quy định cụ thể trong quá trình lập kế hoạch

# Quá trình phát triển

* Quy trình phần mềm
* <số>
* Một tập hợp các giai đoạn và mỗi giai đoạn là một chuỗi các bước
* Trình tự các bước cho một pha - phương pháp luận cho giai đoạn đó.
* Tại sao có các giai đoạn
  + Sử dụng phân chia và chinh phục
  + mỗi giai đoạn xử lý một phần khác nhau của vấn đề
  + giúp xác nhận liên tục

# Quá trình phát triển

* Quy trình phần mềm
* <số>
* Thường có các hoạt động này:
  + Phân tích yêu cầu
  + Kiến trúc
  + Thiết kế
  + Mã hóa
  + Thử nghiệm
  + Chuyển
* Các mô hình khác nhau thực hiện chúng theo cách khác nhau

# Phân tích yêu cầu

* Quy trình phần mềm
* <số>
* Để hiểu và nêu chính xác vấn đề
* Hình thành cơ sở thỏa thuận giữa người dùng và nhà phát triển
* chỉ rõ “ **cái gì** ”, không phải “ **làm thế nào** ”.
* Không phải là một nhiệm vụ dễ dàng, vì nhu cầu thường không được hiểu.
* Yêu cầu thông số kỹ thuật của các hệ thống trung bình thậm chí có thể là hàng trăm trang
* Đầu ra là tài liệu Đặc tả Yêu cầu Phần mềm (SRS)

# Thiết kế

* Quy trình phần mềm
* <số>
* Một bước tiến quan trọng trong việc chuyển từ miền vấn đề sang miền giải pháp; ba nhiệm vụ chính
  + Thiết kế kiến ​​trúc - các thành phần và các đầu nối cần có trong hệ thống
  + Thiết kế cấp cao - mô-đun và cấu trúc dữ liệu cần thiết để triển khai kiến ​​trúc
  + Thiết kế chi tiết - logic của mô-đun
* Hầu hết các phương pháp tập trung vào kiến ​​trúc hoặc thiết kế mức cao
* Các đầu ra là các tài liệu thiết kế kiến ​​trúc / thiết kế / logic

# Mã hóa

* Quy trình phần mềm
* <số>
* Chuyển đổi thiết kế thành mã theo ngôn ngữ cụ thể
* Mục tiêu: Thực hiện thiết kế với mã đơn giản và dễ hiểu.
  + Mã nên đơn giản và dễ đọc.
* Giai đoạn mã hóa ảnh hưởng đến cả thử nghiệm và bảo trì. Mã được viết tốt có thể làm giảm nỗ lực kiểm tra và bảo trì.
* Đầu ra là mã

# Thử nghiệm

* Quy trình phần mềm
* <số>
* Khuyết tật được giới thiệu trong từng giai đoạn
* Họ phải được tìm thấy và loại bỏ để đạt được chất lượng cao
* Thử nghiệm đóng vai trò quan trọng này
* Mục tiêu: Xác định hầu hết các khuyết tật
* Là một nhiệm vụ rất tốn kém; phải được lên kế hoạch và thực hiện đúng cách.
* Các đầu ra là các kế hoạch / kết quả thử nghiệm và mã thử nghiệm cuối cùng (hy vọng đáng tin cậy)

# Phân phối nỗ lực

* Quy trình phần mềm
* <số>
* **Phân phối nỗ lực:**
  + Yêu cầu 10-20%
  + Thiết kế 10-20%
  + Mã hóa 20-30%
  + Thử nghiệm 30-50%
* Mã hóa không phải là đắt nhất.

# Phân phối nỗ lực…

* Quy trình phần mềm
* <số>
* Cách người lập trình dành thời gian của họ
  + Viết chương trình 13%
  + Chương trình đọc và hướng dẫn sử dụng 16%
  + Giao tiếp công việc 32%
  + Khác 39%
* Các lập trình viên dành nhiều thời gian hơn trong việc đọc các chương trình hơn là viết chúng.
* Viết chương trình là một phần nhỏ trong cuộc sống của họ.

# Khiếm khuyết

* Quy trình phần mềm
* <số>
* Phân phối các lần xuất hiện lỗi theo pha
  + Req. - 20%
  + Thiết kế - 30%
  + Mã hóa - 50%
* Khuyết tật có thể được tiêm vào bất kỳ giai đoạn chính nào.
* Chi phí thời gian chờ: Chi phí xóa bỏ lỗi tăng theo cấp số nhân với thời gian chờ.

# Khiếm khuyết…

* Quy trình phần mềm
* <số>
  + - * Chi phí sửa chữa
      * Lỗi (thang đo log)
      * Thời gian
* Cách rẻ nhất là phát hiện và loại bỏ các khuyết tật gần nơi nó được tiêm.
* Do đó phải kiểm tra lỗi sau mỗi pha.

# Mô hình quy trình

* Quy trình phần mềm
* <số>
* Mô hình quy trình chỉ định quy trình chung, thường là một tập hợp các giai đoạn
* Mô hình này sẽ phù hợp với một loại dự án
* Tức là một mô hình cung cấp cấu trúc chung của quá trình có thể được theo sau bởi một số dự án để đạt được mục tiêu của họ

# Quy trình dự án

* Quy trình phần mềm
* <số>
* Nếu một dự án chọn một mô hình, nó sẽ thường thay đổi nó cho phù hợp với dự án
* Điều này tạo ra thông số kỹ thuật cho quy trình dự án
* Quá trình này có thể được theo sau trong dự án
* Tức là quá trình thực sự được thực thi; quy trình spec là kế hoạch về những gì nên được thực thi; mô hình quy trình là một đặc tả quy trình chung
* Nhiều mô hình đã được đề xuất cho quá trình phát triển

# Mô hình quy trình sinh viên điển hình

* Quy trình phần mềm
* <số>
* Nhận được vấn đề stmt - Code - làm một số thử nghiệm - cung cấp / demo
* Tại sao mô hình quy trình này không thể được sử dụng cho các dự án thương mại?
  + Sản xuất phần mềm sinh viên, không phải là những gì chúng tôi đang theo đuổi
  + Không thể đảm bảo chất lượng mong muốn cho phần mềm có độ bền công nghiệp

# Các mô hình quy trình chung

* Quy trình phần mềm
* <số>
* Thác nước - lâu đời nhất và được sử dụng rộng rãi
* Tạo mẫu
* Lặp lại - hiện được sử dụng rộng rãi
* Timeboxing

# Một quy trình đơn giản và quen thuộc

* **Mã số**
* **Biên dịch**
* **Gỡ lỗi**
* **Vấn đề**
* **Tuyên bố**
* **Giải phóng**
* **vấn đề**
* **vấn đề**
* **Kiểm tra đơn vị**
* ***1. Hầu hết mọi người thực hiện và làm theo quy trình này, nhưng***
* ***tiếc là một số*** *thử nghiệm đơn vị* bỏ qua *hoặc gỡ lỗi.*
* ***2. Ngoài ra, một số tiến hành mà không hiểu rõ "tuyên bố vấn đề" ---- đó là yêu cầu***

# Một số quy trình phát triển phần mềm “truyền thống”

* **Quy trình “đơn giản” được nhiều người sử dụng trong nhiều năm mà không chính thức bao gồm các hoạt động phát triển quan trọng khác như phân tích yêu cầu, thiết kế, thử nghiệm chính thức hoặc đóng gói.**
* **Việc công nhận sự cần thiết cho các quy trình chính thức ban đầu được điều khiển bởi những thất bại trong việc phát triển phần mềm phức tạp lớn**
  + **Thác nước: *quá trình sớm nhất và đối phó với không có quá trình***
  + **Gia tăng: *đối phó với việc phân hủy các hệ thống lớn***
  + **Xoắn ốc: *đối phó với quản lý rủi ro***
  + **Rational Unified Process: *đối phó với nhiều vấn đề phát triển và quản lý***

# Mô hình thác nước

* **Yêu cầu**
* **Thiết kế**
* **Mã số**
* **Kiểm tra**
* **Tích hợp và**
* **Gói**
* **1. Yêu cầu phải được chỉ định trong**
* **bước đầu tiên.**
* **2. Bốn nhiệm vụ chính phải được hoàn thành trong**
* **trình tự: *yêu cầu* , *thiết kế* , *mã* ,**
* **và *thử nghiệm,* tiếp theo là đóng gói.**
* **3. Đầu ra của một giai đoạn nguồn cấp dữ liệu vào tiếp theo**
* **theo thứ tự, và do đó dễ dàng**
* **được theo dõi bởi ban quản lý**

# Mô hình gia tăng A - Tích hợp liên tục

* **Req.** **1**
* **Req.2**
* **Req.** **n**
* **Des.**
* **Des.**
* **Des.**
* **mã**
* **Thùng tích hợp**
* **Kiểm tra**
* **mã**
* **mã**
* **Kiểm tra**
* **Kiểm tra**
* **.** **.** **.**
* **.** **.** **.** **.**
* **.** **.** **.** **.** **.** **.**
* **Hệ thống**
* **Kiểm tra**
* **Mỗi yêu cầu / mục chính**
* **được phát triển riêng thông qua**
* **cùng một chuỗi: yêu cầu ,**
* **thiết kế , mã và kiểm tra đơn vị .**
* **Khi các phần phát triển được hoàn thành,**
* **chúng được liên tục hợp nhất và**
* **được tích hợp vào một nhóm chung cho**
* **kiểm tra hệ thống tích hợp**

# Mô hình gia tăng B - Nhiều bản phát hành

* **Yêu cầu**
* **Thiết kế**
* **Mã số**
* **Kiểm tra**
* **Gói**
* ***Rel.*** ***1***
* **Yêu cầu**
* **Thiết kế**
* **Mã số**
* **Kiểm tra**
* **Gói**
* ***Rel.*** ***n***
* **.**
* **.**
* **.**
* **Mỗi bộ yêu cầu nhỏ được phát triển,**
* **đóng gói và phát hành trong một bản phát hành đa**
* **thời trang.**

# Mô hình xoắn ốc

* **Kế hoạch giai đoạn tiếp theo**
* **Phát triển, Xác minh**
* **Sản phẩm cấp độ tiếp theo**
* **Xác định mục tiêu,**
* **Giải pháp thay thế, Ràng buộc**
* **Đánh giá các lựa chọn thay thế,**
* **Xác định, giải quyết rủi ro**
* **req.**
* **Spec.**
* **thiết kế**
* **mô hình**
* **thiết kế**
* **xác thực**
* **kế hoạch kiểm tra**
* **mã**
* **đơn vị**
* **kiểm tra**
* **kế hoạch phát triển**
* **req.** **kế hoạch**
* **rủi ro**
* **phân tích**
* **proto**
* **kiểu**
* **thiết kế**
* **sys.**
* **kiểm tra**
* ***"Ôn tập"***
* **- Phát triển phần mềm**
* **các hoạt động được luân chuyển qua**
* **4 giai đoạn**
* **- Một quá trình chống lại rủi ro**

# Mô hình xoắn ốc

* Xác định các mục tiêu, lựa chọn thay thế hoặc ràng buộc cho mỗi chu kỳ xoắn ốc.
* Đánh giá các lựa chọn thay thế liên quan đến mục tiêu và ràng buộc. Khi thực hiện bước này, nhiều rủi ro được xác định và đánh giá.
* Tùy thuộc vào số lượng và loại rủi ro được xác định, phát triển mẫu thử nghiệm, đánh giá chi tiết hơn, phát triển tiến hóa hoặc một số bước khác để giảm thêm rủi ro đạt được mục tiêu đã xác định. Mặt khác, nếu rủi ro giảm đáng kể, bước tiếp theo có thể chỉ là một nhiệm vụ như yêu cầu, thiết kế hoặc mã.
* Xác nhận thành tích của mục tiêu và kế hoạch cho chu kỳ tiếp theo.

# -Quy trình hợp nhất hợp lý

* **Các giai đoạn**
* ***Khởi đầu***
* ***Xây dựng***
* ***Xây dựng***
* ***Chuyển đổi***
* ***Yêu cầu***
* ***Thiết kế***
* ***Triển khai thực hiện***
* ***Kiểm tra***
* ***Tích hợp***
* **Hoạt động**
* **Mọi phát triển phần mềm**
* **hoạt động đi qua 4**
* **giai đoạn khởi đầu , xây dựng ,**
* **xây dựng** và chuyển đổi

# Tiêu chí xuất nhập cảnh

* **Quá trình**
* **Hoạt động**
* **Entry**
* **Tiêu chí**
* **Gặp?**
* **Lối thoát**
* **Tiêu chí**
* **Gặp?**
* **Vâng**
* **Vâng**
* **Không**
* **Không**
* **Để các mô hình quy trình không chỉ đơn thuần là**
* **"hướng dẫn", nó phải bao gồm một danh sách các điều kiện hoặc**
* **yêu cầu xác định:**
* **- tiêu chí nhập cảnh trước khi biểu diễn**
* **một hoạt động trong một quá trình.**
* **- tiêu chí thoát trước khi hoạt động trong**
* **quá trình được coi là hoàn thành.**

# Đánh giá quy trình

* **Phát triển phần mềm và hỗ trợ phần mềm có thể được thực hiện với quá trình rất ít hoặc với các quy trình rất tinh vi, được xác định rõ, được tổ chức tốt và được thực thi tốt.**
* **Tổ chức kỹ nghệ phần mềm của bạn trưởng thành như thế nào và bạn có cần cải thiện không?**
* **ISO (ISO 9000 series) và SEI (tại Carnegie Mellon) là hai tổ chức hàng đầu giúp đánh giá quá trình**
* ***Không có quy trình***
* ***Quy trình trưởng thành***
* ***Bạn đang ở đâu***
* ***phổ rộng này?***

# SEI CMM

* **Viện Kỹ thuật phần mềm (SEI) đã đề xuất một Mô hình trưởng thành năng lực (CMM) để giúp các tổ chức phần mềm *đánh giá* *sự trưởng thành của họ* và *đưa ra hướng dẫn* trong phát triển phần mềm.**
  + **Ban đầu** : không có quy trình và thành công nào là do may mắn hoặc với một người đặc biệt.
  + **Lặp lại** : đã làm chủ được 6 quy trình và có thể lặp lại thành công với 6 quy trình này: 1) yêu cầu mgmt, 2) theo dõi dự án, 3) đảm bảo chất lượng, 4) lập kế hoạch dự án, 5) quản lý cấu hình phụ và 6) quản lý cấu hình
  + **Xác định** : đã làm chủ được 7 quy trình và có thẩm quyền xây dựng phần mềm: 1) quy trình tổ chức, 2) chương trình đào tạo, 3) kỹ thuật sản phẩm, 4) đánh giá ngang hàng, 5) định nghĩa quy trình tổ chức, 6) tích hợp mềm. mgmt và 7) phối hợp giữa các nhóm
  + **Được quản lý** : đã giới thiệu thêm 2 quy trình xử lý đo lường và chất lượng định lượng: 1) quản lý quy trình định lượng và 2) chất lượng mgmt
  + **Tối ưu hóa** : đạt đến mức cao nhất này đòi hỏi phải làm chủ cải tiến liên tục với 3 quy trình khác: 1) phòng ngừa khiếm khuyết, 2) quản lý thay đổi công nghệ, 3) quản lý thay đổi quy trình

# 5 cấp độ “Mô hình trưởng thành khả năng” ban đầu (CMM)

* **Ban đầu**
* **Lặp lại**
* **Được định nghĩa**
* **Được quản lý**
* **Tối ưu hóa**
* **Cấp độ 1**
* **Cấp 4**
* **Cấp 3**
* **Cấp độ 2**
* **Cấp 5**
* ***Ít trưởng thành nhất***
* ***Hầu hết người trưởng thành***
* **Tổng số 18 quy trình cần phải được làm chủ để đạt được mức "tối ưu"**

# SEI CMMI

* **Năm 2001, CMM đã được nâng cấp lên CMMI (CMM tích hợp).** **Có những khía cạnh chính yếu đối với CMMI:**
  + **Kỹ thuật hệ thống**
  + **Kỹ thuật phần mềm**
  + **Tích hợp sản phẩm và quy trình phát triển**
  + **Tìm nguồn cung ứng**
* **Phần kỹ thuật phần mềm của CMMI có hai biểu diễn:**
  + **Được tổ chức** : tương tự như đánh giá CMM của tổ chức
  + **Liên tục** : tốt hơn để đánh giá và cải thiện sự trưởng thành của từng quy trình

# Quy trình của CMMI

* **Có 25 quy trình bao gồm 4 loại chính:**
  + **Quản lý quy trình (có 5 quy trình):**
    - **Tập trung quá trình tổ chức**
    - **Định nghĩa quy trình tổ chức**
    - **Đào tạo tổ chức**
    - **Hiệu suất quy trình tổ chức**
    - **Tổ chức đổi mới và triển khai**
  + **Quản lý dự án (có 8 quy trình):**
    - **Lập kế hoạch dự án**
    - **Giám sát và kiểm soát dự án**
    - **Quản lý thỏa thuận nhà cung cấp**
    - **Quản lý dự án tích hợp**
    - **Quản lý rủi ro**
    - **Tích hợp teaming**
    - **Quản lý nhà cung cấp tích hợp**
    - **Quản lý dự án định lượng**

# Quy trình của CMMI (cont,)

* **Kỹ thuật (có 6 quy trình)**
  + **Yêu cầu phát triển**
  + **Quản lý yêu cầu**
  + **Giải pháp kỹ thuật**
  + **Tích hợp sản phẩm**
  + **xác minh**
  + **Xác nhận**
* **Hỗ trợ (có 6 quy trình)**
  + **Quản lý cấu hình**
  + **Quy trình và đảm bảo chất lượng sản phẩm**
  + **Đo lường và phân tích**
  + **Môi trường tổ chức để tích hợp**
  + **Phân tích và phân tích quyết định**
  + **Phân tích và phân tích nhân quả**

# Cấp độ cho các mô hình liên tục và được tổ chức trong CMM I

* **Cấp 5**
* **Cấp 4**
* **Cấp 3**
* **Cấp độ 2**
* **Cấp độ 1**
* **Cấp 0**
* **Liên tiếp**
* **( *Mức độ khả năng* )**
* **Đã tổ chức**
* **( *Mức độ trưởng thành* )**
* **chưa hoàn thiện**
* **Đã thực hiện**
* **Được quản lý**
* **Được định nghĩa**
* **Quản lý định lượng**
* **Tối ưu hóa**
* **Quản lý định lượng**
* **Ban đầu**
* **Được quản lý**
* **Được định nghĩa**
* **Tối ưu hóa**
* **- - - - - -**

# Mô hình liên tục và được tổ chức

* **Trong biểu diễn liên tục, mỗi quá trình bắt đầu ở mức năng lực 0 và nâng cao các mức năng lực dựa trên việc đạt được “mục tiêu chung” và “các mục tiêu phụ cụ thể”.**
  + **Cho phép tổ chức chọn và chọn quy trình để tập trung vào dựa trên nhu cầu của tổ chức**
  + **Cho phép so sánh khu vực quy trình theo khu vực quy trình giữa các tổ chức**
  + **Cho phép di chuyển dễ dàng hơn từ các tiêu chuẩn khác**
* **Trong biểu diễn theo giai đoạn, tổ chức bắt đầu ở mức trưởng thành 1 và di chuyển lên các cấp dựa trên việc nắm vững các quy trình.**
  + **Cho phép di chuyển dễ dàng từ mô hình CMM trước đó**
  + **Cung cấp hướng dẫn về trình tự trưởng thành theo khu vực quy trình**
  + **Cho phép so sánh dễ dàng hơn các tổ chức theo mức độ trưởng thành**

# Mối quan hệ của mục tiêu và thực tiễn

* **Quá trình**
* **Khu vực 1**
* **Quá trình**
* **Khu vực 25**
* **- - - - - - - - - - - - - - - - -**
* **Riêng**
* **Mục tiêu 1**
* **Riêng**
* **Mục tiêu x**
* **- - - -**
* **Riêng**
* **Thực hành 1**
* **Riêng**
* **Thực hành w**
* **- - -**
* **Riêng**
* **Mục tiêu 1**
* **- - - -**
* **Riêng**
* **Mục tiêu z**
* **Chung**
* **Mục tiêu 1**
* **Chung**
* **Mục tiêu 5**
* **- - - -**
* **Chung**
* **Thực hành 1**
* **Chung**
* **Thực hành n**
* **Chung**
* **Thực hành 1**
* **- - - -**
* **- - - -**
* **Riêng**
* **Thực hành 1**
* **Riêng**
* **Thực hành k**
* **Chung**
* **Thực hành p**
* **- - -**

# Đạt được “ *Cấp độ khả năng”* của từng Vùng quy trình trong Mô hình đại diện liên tục

* **chưa hoàn thiện**
* **Đã thực hiện**
* **Được quản lý**
* **Được định nghĩa**
* **Quản lý định lượng**
* **Tối ưu hóa**
* **CL0**
* **CL1**
* **CL2**
* **CL3**
* **CL4**
* **CL5**
* ***+ (Mục tiêu cụ thể)***
* ***+ (Mục tiêu chung 1)***
* ***+ (Mục tiêu chung 2)***
* ***+ (Mục tiêu chung 3)***
* ***+ (Mục tiêu chung 4)***
* ***+ (Mục tiêu chung 5)***

# Đạt được *“Độ trưởng thành”* (ML) trong mô hình Đại diện theo giai đoạn

* **ML1 (0 quy trình): không có quy trình**
* **ML2 (7 quy trình): 1) Yêu cầu Mgmt, 2) Lập kế hoạch dự án, 3) Theo dõi dự án, 4) Thỏa thuận nhà cung cấp, 5) Đo lường và phân tích, 6) Đảm bảo chất lượng sản phẩm và quy trình, 7) Cấu hình mgmt**
* **ML3 (14 quy trình): 1) Yêu cầu phát triển, 2) Giải pháp kỹ thuật, 3) Tích hợp sản phẩm, 4) Xác minh, 5) Xác nhận, 6) Tập trung tổ chức, 7) Định nghĩa quy trình tổ chức, 8) Đào tạo tổ chức, 9) quản lý, 10) Quản lý rủi ro, 11) Tích hợp nhóm, 12) Tích hợp nhà cung cấp mgmt, 13) Phân tích và giải quyết quyết định, 14) Môi trường tổ chức để tích hợp**
* **ML4 (2 quy trình): 1) Hiệu suất quy trình tổ chức, 2) Quản lý dự án định lượng**
* **ML5 (2 quy trình): 1) Đổi mới và triển khai tổ chức, 2) Phân tích và giải quyết nhân quả**

# Quy trình và định nghĩa quy trình

* **2 thành phần chính của quá trình định nghĩa:**
  + **Các hoạt động chính**
  + **Sắp xếp các hoạt động**
* **Hầu hết các tổ chức cần phải sửa đổi một quy trình hiện có để phù hợp hơn với nhu cầu của họ ----- do đó họ phải xác định chi tiết hơn và truyền đạt các định nghĩa quy trình đã sửa đổi (một nỗ lực lớn)**
* **Mở rộng định nghĩa quy trình thành cấp độ "tinh tế" hơn:**
  + **Mô tả chi tiết các *hoạt động***
  + ***Kiểm soát*** cần thiết cho *lối vào* và *lối ra* của từng hoạt động và thứ tự của các hoạt động
  + ***Các tạo phẩm*** phát sinh từ các hoạt động
  + ***Nguồn nhân lực*** cần thiết để thực hiện các hoạt động
  + ***Các công cụ*** có thể cần thiết để hỗ trợ hoạt động của các hoạt động

# Đạt được *“Độ trưởng thành”* (ML) trong mô hình Đại diện theo giai đoạn

* **ML1 (0 quy trình): không có quy trình**
* **ML2 (7 quy trình): Khu vực xử lý**
  + **1) Yêu cầu Mgmt, Kỹ thuật**
  + **2) Dự án lập kế hoạch dự án**
  + **3) Dự án giám sát Dự án Mgmt**
  + **4) Thỏa thuận nhà cung cấp Mgmt Project Mgmt**
  + **5) Hỗ trợ đo lường và phân tích**
  + **6) Quy trình và đảm bảo chất lượng sản phẩm**
  + **7) Cấu hình mgmt Hỗ trợ**

# Đạt được *“Độ trưởng thành”* (ML) trong mô hình Đại diện theo giai đoạn

* **ML3 (14 quy trình): Khu vực xử lý**
  + **1) Yêu cầu phát triển Kỹ thuật**
  + **2) Kỹ thuật giải pháp kỹ thuật**
  + **3) Kỹ thuật tích hợp sản phẩm**
  + **4) Kỹ thuật xác minh**
  + **5) Kỹ thuật xác nhận**
  + **6) Quá trình tổ chức tập trung quá trình Mgmt**
  + **7) Quy trình tổ chức định nghĩa Quy trình Mgmt**
  + **8) Quy trình đào tạo tổ chức Mgmt**
  + **9) Quản lý dự án tích hợp Dự án Mgmt**
  + **10) Quản lý rủi ro**
  + **11) Tích hợp Team Mgmt dự án**
  + **12) Nhà cung cấp Mgmt dự án tích hợp**
  + **13) Phân tích và phân tích quyết định**
  + **14) Môi trường tổ chức để hỗ trợ tích hợp**

# Đạt được *“Độ trưởng thành”* (ML) trong mô hình Đại diện theo giai đoạn

* **ML4 (2 quy trình): Khu vực xử lý**
  + **1) Quy trình tổ chức thực hiện Quy trình Mgmt**
  + **2) Dự án quản lý dự án định lượng Mgmt**
* **ML5 (2 quy trình): Khu vực xử lý**
  + **1) Đổi mới tổ chức và triển khai Quy trình Mgmt**
  + **2) Phân tích và phân tích nhân quả**

CHƯƠNG 5

* **Chương 5**
* **Phương pháp quy trình mới và mới nổi**
* **Mùa thu 2011**

# Mục tiêu

* Hiểu không
  + Hạn chế của phương pháp xử lý truyền thống
  + Khả năng áp dụng của các quy trình nhanh
  + Nguyên lý cơ bản của các quy trình nhanh
* Làm quen với một số quy trình nhanh

# Quy trình nhanh là gì

* Gia đình của phương pháp phát triển phần mềm
* Bản phát hành ngắn và lặp lại
  + Chia thành từng miếng nhỏ
  + Phát hành cho khách hàng thường xuyên nhất có thể
* Thiết kế gia tăng
  + Đừng cố gắng hoàn thành thiết kế
  + Chậm trễ các quyết định thiết kế càng lâu càng tốt
* Sự tham gia của người sử dụng
  + Nhận phản hồi của người dùng thường xuyên nhất có thể
* Tài liệu tối thiểu
  + Chỉ số lượng tài liệu cần thiết
* Giao tiếp không chính thức
  + Mọi người giao tiếp tốt hơn một cách không chính thức
* Thay đổi
  + Mọi thứ sẽ thay đổi, lập kế hoạch cho điều đó

# Tuyên ngôn nhanh nhẹn

* **Chúng tôi đang khám phá những cách tốt hơn để phát triển phần mềm bằng cách thực hiện nó và giúp những người khác làm điều đó.**
* **Đó là, trong khi có giá trị trong các mục bên phải, chúng tôi đánh giá các mục ở bên trái nhiều hơn.**
* http://agilemanifesto.org
* **1. Cá nhân và tương tác - -** qua các quy trình và công cụ **2** . **Phần mềm làm việc** **- -** trên tài liệu toàn diện **3** . **Hợp tác khách hàng** **-** **-** qua thương lượng hợp đồng **4** . **Trả lời thay đổi** **- -** hơn sau một kế hoạch

# Các vấn đề với quy trình truyền thống

* Thời gian phát triển dài
* Không có khả năng đối phó với những thay đổi về yêu cầu
* Giả định các yêu cầu được hiểu khi bắt đầu dự án
* Phụ thuộc vào nỗ lực phát triển anh hùng
* Phương pháp luận phức tạp
* Xử lý chất thải / sao chép nỗ lực

# Một số phương pháp nhanh nhẹn

* Không có quy trình nào sẽ hoạt động đối với tất cả các dự án
* Chọn quy trình phù hợp nhất với dự án
* Bốn trong số rất nhiều phương pháp nhanh nhẹn
  + Lập trình cực đoan (XP)
  + Crystal Clear / Orange
  + -Quy trình hợp nhất hợp lý
    - Khung
  + Microsoft Solutions Framework

# XP: Giá trị cốt lõi

* ***Giao tiếp*** (giữa nhóm và với khách hàng)
* ***Tính đơn giản*** (trong thiết kế và mã)
* ***Phản hồi*** (ở nhiều cấp độ)
* ***Can đảm****(để thực hiện và thực hiện quyết định khó khăn)*

# XP: Nguyên tắc cơ bản

* Phản hồi nhanh
  + Sử dụng lập trình ghép nối, kiểm tra đơn vị, tích hợp và các tương tác và bản phát hành ngắn
* Sự đơn giản
  + Hãy thử cách tiếp cận đơn giản nhất có thể
* Gia tăng
  + Hãy thử những thay đổi nhỏ sẽ tăng lên
  + Sử dụng tái cấu trúc mã - các sửa đổi nhỏ để cải thiện mã
* Nắm lấy thay đổi
  + Bảo tồn các tùy chọn cho tương lai
  + Trì hoãn các quyết định cam kết một đến một đường dẫn càng lâu càng tốt
* Công việc có chất lượng
  + Tạo ra sản phẩm tốt nhất có thể

# XP: Các nguyên tắc khác

* Học tập liên tục
* Đầu tư ban đầu nhỏ
* Chơi để giành chiến thắng
* Thí nghiệm bê tông
* Mở, giao tiếp trung thực
* Làm việc với bản năng của mọi người
* Chấp nhận trách nhiệm
* Thích ứng địa phương
* Sự làm sáng tỏ trong ngành du lịch
* Đo lường trung thực

# XP: 12 thực hành

* Lập kế hoạch
  + Xác định các tính năng được đưa vào bản phát hành tiếp theo
  + Ưu tiên kinh doanh và ước tính kỹ thuật
* Bản phát hành ngắn
  + Nhận hệ thống làm việc nhanh chóng
  + Phát hành phiên bản mới trong chu kỳ ngắn (2 đến 4 tuần)
  + Dựa vào các chi tiết mới về phản hồi của khách hàng
* Ẩn dụ
  + Sử dụng phép ẩn dụ thay vì kiến ​​trúc
* Thiết kế đơn giản
  + Loại bỏ sự phức tạp không cần thiết càng sớm càng tốt
  + Thiết kế có thể được thay đổi trong các phiên bản sau
* Hướng phát triển thử nghiệm
  + Liên tục và tự động
  + Viết các bài kiểm tra trước khi viết mã
* Cải tiến thiết kế (Refactoring)
  + Xóa trùng lặp, cải thiện giao tiếp và đơn giản hóa hoặc thêm tính linh hoạt cần thiết

# XP: 12 thực hành

* Lập trình ghép nối
  + Hai lập trình viên làm việc cùng một máy
* Quyền sở hữu tập thể
  + Bất kỳ ai cũng có thể thay đổi bất kỳ mã nào vào bất kỳ lúc nào
* Hội nhập liên tục
  + Tích hợp và kiểm tra mỗi khi công việc được hoàn thành (nhiều lần trong ngày)
* Bước đi bền vững
  + 40 giờ một tuần hợp lý
  + Không bao giờ làm thêm giờ hai tuần liên tiếp
* Khách hàng tại chỗ
  + Bao gồm khách hàng thực sự vào nhóm
* Tiêu chuẩn mã hóa
  + Mọi người đều cần sử dụng cùng một quy tắc

# XP: Lập kế hoạch

* Chỉ lập kế hoạch lặp lại ngay lập tức tiếp theo
* Bốn biến để điều chỉnh trong một dự án
  + Phạm vi
    - Cần phải làm gì để hệ thống hữu ích
  + Giá cả
  + Phẩm chất
  + Thời gian
* Phạm vi dễ thay đổi hơn (tính năng thả)
* Sử dụng các câu chuyện (đại diện cho một nhiệm vụ, tương tự như các ca sử dụng)

# XP: Lập kế hoạch (2)

* Khách hàng quyết định
  + Phạm vi: cần thiết cho hệ thống hữu ích
  + Ưu tiên: xác định và ưu tiên các đặc tính
  + Phạm vi phát hành: những gì được bao gồm trong mỗi bản phát hành
  + Ngày phát hành
* Bên kỹ thuật quyết định
  + Ước tính: mỗi tính năng sẽ mất bao lâu
  + Hậu quả: công nghệ và ngôn ngữ lập trình tốt nhất
  + Quy trình: các hoạt động công việc được thực hiện như thế nào; tổ chức nhóm
* Ai đưa ra quyết định?

# XP: Lập kế hoạch trò chơi

* Mục tiêu: Tối đa hóa giá trị của phần mềm được tạo ra
* Chiến lược: Đầu tư càng ít càng tốt để có được hầu hết các chức năng càng nhanh càng tốt
* Ba giai đoạn:
  + Thăm dò - phân định phạm vi có thể
  + Cam kết - chọn phạm vi, ngày phát hành
    - Kinh doanh sắp xếp các câu chuyện (tính năng) thành ba loại: cần thiết, quan trọng và tốt đẹp để có
    - Phát triển sắp xếp các câu chuyện theo rủi ro: có thể ước tính chính xác, có thể ước tính một cách hợp lý, những người không thể ước tính
    - Doanh nghiệp chọn phạm vi, các thẻ câu chuyện đó có thể đáp ứng lịch biểu hoặc lịch biểu được điều chỉnh để đáp ứng phạm vi
  + Chỉ đạo - điều chỉnh kế hoạch
    - Thêm, xóa hoặc điều chỉnh câu chuyện

# Trò chơi lập kế hoạch XP - Thăm dò

* Mục tiêu: Tìm hiểu / quyết định những gì hệ thống có thể làm
* Di chuyển
  + Viết một câu chuyện (khách hàng)
  + Ước tính một câu chuyện (nhà phát triển)
  + Tách một câu chuyện

# Trò chơi XP Planning - cam kết

* Doanh nghiệp chọn phạm vi và ngày phát hành tiếp theo
* Phát triển cam kết với nó
* Di chuyển
  + Sắp xếp các câu chuyện (thiết yếu, quan trọng, tốt đẹp)
  + Sắp xếp theo rủi ro (ước tính chính xác, đoán, không có đầu mối)
  + Bộ phát triển vận tốc (lý tưởng so với thời gian lịch)
  + Doanh nghiệp chọn phạm vi (câu chuyện / nhiệm vụ nào)

# Trò chơi lập kế hoạch XP - Chỉ đạo

* Cập nhật và điều chỉnh gói
* Di chuyển
  + Khách hàng có thể chọn câu chuyện để lặp lại
  + Phục hồi
    - Nhà phát triển yêu cầu khách hàng thực hiện tái cấp phép
  + Thêm / thay thế câu chuyện
  + Ước tính lại

# Pha lê

* Một phương pháp không thể phù hợp với tất cả các dự án
* Những gì / làm thế nào để thích ứng với dự án
* Phân loại dự án theo
  + Kích thước (số lượng nhà phát triển)
  + Tính nghiêm trọng: Mất mát mà sự cố sẽ gây ra
    - Đời sống
    - Tiền thiết yếu
    - Tiền tùy ý
    - Khách hàng thoải mái
  + Ưu tiên (áp lực thời gian)

# Ba phương pháp được xác định

* Màu tối hơn, phương pháp càng nặng
  + Tinh thể rõ ràng
    - Các dự án không quan trọng
    - Đội sáu đến tám
  + Tinh thể cam
    - Quan trọng nhưng không phải cuộc sống quan trọng
    - Đội lên đến 40 người
  + Trang web màu cam tinh thể
* Cần nhiều hơn ba
  + Cuộc sống quan trọng
  + Dự án quy mô lớn
  + Vv

# Điều chỉnh phương pháp luận

* Các phương pháp lớn hơn cho các nhóm lớn hơn
* Các phương pháp nặng hơn cho các dự án quan trọng hơn
* Ưu tiên giao tiếp tương tác qua tài liệu chính thức
* Hiểu rằng mọi người khác nhau. Các quy trình kỷ luật cao khó áp dụng hơn
* Giả sử mọi người muốn trở thành 'công dân tốt'

# Tinh thể - Thuộc tính

* Giao hàng thường xuyên
  + Ít nhất cứ sau vài tháng
  + Các loại phân phối: tất cả người dùng, người dùng bị giới hạn, chỉ giới thiệu
* Cải thiện phản xạ
  + Luôn luôn suy nghĩ những gì có thể được cải thiện
* Đóng giao tiếp
  + Các thành viên trong nhóm ở gần
* An toàn cá nhân
  + Không trả thù để nói lên
* Tiêu điểm
  + Giảm thiểu gián đoạn
* Dễ dàng truy cập vào người dùng chuyên gia
  + Được tư vấn chuyên gia khi cần
* Môi trường kỹ thuật tốt
  + Kiểm tra tự động, quản lý cấu hình và tích hợp thường xuyên

|  |  |  |
| --- | --- | --- |
|  | * **Thông thoáng** | * **trái cam** |
| * Đội | * Một đội, cùng một phòng | * Các nhóm khác nhau để lập kế hoạch hệ thống, giám sát dự án, kiến ​​trúc, công nghệ, chức năng, cơ sở hạ tầng và thử nghiệm bên ngoài |
| * Vai trò / Người riêng biệt | * Ít nhất 4 người, đóng vai trò của nhà tài trợ, nhà thiết kế cao cấp, lập trình viên, người dùng. * Các vai trò khác có thể được điền bởi cùng một người, bao gồm người quản lý dự án, chuyên gia kinh doanh, người thu thập yêu cầu | * 14 vai trò khác nhau, được chơi bởi những người khác nhau, bao gồm (bên cạnh những người trong Crystal Clear), Giám đốc dự án, Nhà tài trợ, Chuyên gia kinh doanh, Kiến trúc sư, Thiết kế Mentor, Người thử nghiệm và Nhà thiết kế giao diện người dùng. |
| * Công việc sản xuất | * 9 mục, bao gồm lịch biểu, trường hợp sử dụng, bản phác thảo thiết kế, trường hợp kiểm tra và hướng dẫn sử dụng. | * 13 mục, bao gồm các tài liệu trong các tài liệu yêu cầu Crystal Clear cộng, báo cáo trạng thái, tài liệu thiết kế giao diện người dùng và thông số kỹ thuật của Inter-team. * Các sản phẩm công việc được phát triển cho đến khi chúng đủ dễ hiểu, chính xác và ổn định để đánh giá ngang hàng. |
| * Độ dài phát hành tối đa | * 2 tháng | * 2-4 tháng, hai lượt xem của người dùng trên mỗi bản phát hành |

* **Hình 5.2: So sánh Crystal Clear và Crystal Orange**

# Crystal - Các tính năng phổ biến

* Tiến độ được theo dõi bằng cách phân phối chứ không phải theo tài liệu
* Thử nghiệm hồi quy tự động
* Sự tham gia trực tiếp của người dùng
* Hai lượt xem của người dùng trên mỗi bản phát hành
* Hội thảo điều chỉnh phương pháp
* Tiêu chuẩn chính sách bắt buộc
* Mã hóa phong cách, vv còn lại để tiêu chuẩn địa phương
* Kỹ thuật cho các vai trò cá nhân được để lại cho từng cá nhân
* Chúng tương tự như XP

# Quy trình hợp nhất dưới dạng Agile

* LÊN thường được coi là nặng
  + Tất cả các yêu cầu khi bắt đầu
  + Kiến trúc, thiết kế trả trước
  + Rational Unified Process (RUP) yêu cầu nhiều tài liệu / hiện vật
* Nhưng
  + lặp đi lặp lại và gia tăng
  + UP không thực sự yêu cầu tất cả các sản phẩm, RUP
* Hạn chế sản phẩm, loại bỏ vai trò, thêm lặp lại

# Microsoft Solutions Framework

* Khung, không phải phương pháp luận (một cấp)
* Có thể được thực hiện như nhanh nhẹn
* Các thành phần
  + Nguyên lý cơ bản
  + Mô hình
  + Môn học
  + Ý chính
  + Thực hành đã được chứng minh
  + khuyến nghị

# MSF: Nguyên tắc cơ bản

* Thúc đẩy truyền thông mở
* Làm việc hướng tới một tầm nhìn chung
* Trao quyền cho các thành viên trong nhóm
* Thiết lập trách nhiệm giải trình rõ ràng và trách nhiệm chung
* Tập trung vào giá trị kinh doanh
* Ở lại nhanh nhẹn, mong đợi thay đổi
* Đầu tư vào chất lượng
* Học hỏi từ mọi kinh nghiệm

# Mô hình nhóm MSF

|  |  |
| --- | --- |
| * **Vai trò** | * **Mục tiêu chất lượng chính** |
| * Quản lý chương trình | * Phân phối trong các ràng buộc của dự án |
| * Phát triển | * Cung cấp thông số kỹ thuật của sản phẩm |
| * Kiểm tra | * Đảm bảo mỗi bản phát hành giải quyết tất cả các vấn đề |
| * Quản lý phát hành | * Triển khai mượt mà và quản lý liên tục |
| * Kinh nghiệm người dùng | * Hiệu suất người dùng nâng cao |
| * Quản lý sản phẩm | * Khách hàng hài lòng |

# Mô hình quy trình MSF

|  |  |  |
| --- | --- | --- |
| * **Giai đoạn** | * **Milestone (có thể phân phối)** | * **Tư duy** |
| * Hình dung | * Tầm nhìn / phạm vi được phê duyệt | * Thám hiểm |
| * Lập kế hoạch | * Kế hoạch dự án được phê duyệt | * Điều tra |
| * Đang phát triển | * Phạm vi hoàn thành (tất cả các tính năng được triển khai) | * Thiết kế sáng tạo) |
| * Ổn định | * Phê duyệt Phát hành Sẵn sàng (chức năng gỡ rối hoàn tất) | * Single-minded (tìm và sửa lỗi) |
| * Triển khai | * Triển khai hoàn tất (cho bản phát hành này) | * Bị xử lý kỷ luật |

# MSF - Các ngành quản lý

* Quản lý dự án
* Quản lý rủi ro
* Quản lý sẵn sàng

# Phát triển nguồn mở

* Mở nguồn, cho phép người dùng sửa đổi
* Nhiều nhà phát triển cung cấp mã trở lại, cải thiện chương trình
* Câu chuyện thành công: Linux, Apache, MySQL
* Không thực sự nhanh nhẹn

# Mã nguồn mở so với Agile

* Điểm tương đồng
  + Bản phát hành nhỏ
  + Giao tiếp không chính thức
  + Tính khả dụng của khách hàng
  + Hội nhập liên tục
  + Chia sẻ tầm nhìn
* Sự khác biệt
  + Nhóm lớn hơn
  + Nhóm phân phối
  + Chia tỷ lệ

# Agile vs Traditional

|  |  |  |
| --- | --- | --- |
|  | * Nhanh nhẹn | * Truyền thống / Nặng |
| * Yêu cầu | * Giả định thay đổi * Yêu cầu không chính thức * Tương tác người dùng không đổi | * Giả sử không có thay đổi * Tài liệu yêu cầu đầy đủ, chi tiết, chính thức |
| * Thiết kế | * Không chính thức * Lặp lại | * Chính thức * Trả trước |
| * Sự tham gia của người sử dụng | * Quan trọng * Thường xuyên | * Bắt đầu (Yêu cầu) * Kết thúc (Thử nghiệm chấp nhận) |
| * Tài liệu | * Tối thiểu, chỉ khi cần * Mã nguồn. | * tài liệu nặng, chính thức |
| * Giao tiếp | * Không chính thức * Trong suốt dự án. | * Các tài liệu * Các bản ghi nhớ và cuộc họp chính thức |
| * Độ phức tạp | * Thấp | * Cao. |
| * Trên không | * Thấp | * Cao |

# Quy trình so với dự án

|  |  |  |
| --- | --- | --- |
|  | * *Agile (XP)* | * *Truyền thống (RUP)* |
| * Phạm vi / kích thước | * Nhỏ bé. * Một nhóm, 3-10 người. | * Các dự án lớn. * Có thể được thu nhỏ |
| * Tính nghiêm trọng | * Tương đối thấp. * Không phải cho các hệ thống sống còn | * Cao hơn |
| * Những người | * Đội chơi * 'Công dân tốt' * Nhà phát triển tốt | * Hầu hết mọi người * Các vai trò khác nhau |
| * Văn hóa công ty | * Các công ty có quy mô nhỏ * Nền văn hóa thư giãn. | * Các công ty lớn hơn * Trang web địa lý từ xa * Văn hóa chính thức. |
| * Tính ổn định | * Không cần thiết * Đối phó với những thay đổi về yêu cầu hoặc môi trường. | * Giả định môi trường và yêu cầu ổn định * Ít phù hợp hơn để đối phó với những thay đổi. |

# Nhanh nhẹn và truyền thống

* Ưu điểm
  + Đơn giản hơn
  + Chi phí thấp, chi phí thấp
  + Giao dịch với các thay đổi
  + Kết quả nhanh
  + Hệ thống có thể sử dụng
* Rủi ro, Nhược điểm
  + Không thể mở rộng
  + Phụ thuộc vào tinh thần đồng đội
  + Dựa vào quyền truy cập vào khách hàng
  + Xung đột văn hóa

CHƯƠNG 6

* **Chương 6**
* **Kỹ thuật yêu cầu**
* **Mùa thu 2011**

# Chuẩn bị kỹ thuật yêu cầu

* **Kế hoạch**
* **Dành cho**
* **Yêu cầu**
* **Hoạt động**
* **Đồng ý trên**
* **Tài nguyên,**
* **Quy trình và**
* **Kế hoạch cho**
* **Yêu cầu**
* **Hoạt động**
* **Có được và**
* **Tổ chức**
* **Đồng ý khi**
* **Tài nguyên và**
* **Quá trình**
* **1. Trước khi thực sự thực hiện các hoạt động kỹ thuật yêu cầu,**
* **điều quan trọng là lập kế hoạch cho các nguồn lực và thời gian cần thiết để thực hiện**
* **bước quan trọng này trong kỹ nghệ phần mềm.**
* **2. Một số tổ chức thậm chí còn thực hiện các yêu cầu kỹ thuật riêng biệt,**
* **hoạt động độc lập và định giá riêng, với tùy chọn gấp chi phí**
* **Vào toàn bộ dự án nếu họ nhận được cuộc gọi để hoàn thành dự án phần mềm.**

# Các yêu cầu chính

* **Sự khám phá**
* **Tài liệu và định nghĩa**
* **Thông số kỹ thuật**
* **Tạo mẫu**
* **Phân tích**
* **Xem lại và xác thực**
* **Thoả thuận và chấp nhận**

# Yêu cầu Kỹ thuật Hoạt động

* **Yêu cầu**
* **Sự khám phá**
* **Yêu cầu**
* **Phân tích**
* **Yêu cầu**
* **Định nghĩa**
* **Yêu cầu**
* **Tạo mẫu**
* **Yêu cầu**
* **Ôn tập**
* **Yêu cầu**
* **Đặc điểm kỹ thuật**
* **Yêu cầu**
* **Thỏa thuận &**
* **chấp thuận**

# Tại sao bộ hoạt động này lại quan trọng và tại sao các yêu cầu phải được ghi lại?

* ***Yêu cầu rõ ràng là cần thiết cho các hoạt động thiết kế và thực hiện.***
* ***Yêu cầu tài liệu là cần thiết để tạo ra các trường hợp thử nghiệm và các kịch bản thử nghiệm - - - đặc biệt là cho các hệ thống lớn, nơi nhóm thử nghiệm là một nhóm người riêng biệt từ các nhà phát triển.***
* ***Yêu cầu tài liệu là cần thiết để kiểm soát phạm vi tiềm năng-creep.***
* ***Yêu cầu tài liệu là cần thiết để tạo tài liệu đào tạo người dùng, tài liệu tiếp thị và tài liệu để hỗ trợ và bảo trì.***
* ***Tài liệu yêu cầu cung cấp một cách để phân đoạn một dự án lớn, ưu tiên các bản phát hành và quản lý dự án dễ dàng hơn***
* **Hãy suy nghĩ về các quy trình nhanh khi bước quan trọng này đôi khi có thể là**
* **bị tổn hại bởi các kỹ sư phần mềm mới làm quen.**

# Yêu cầu

* **Yêu cầu có thể:**
  + **Được trao cho các kỹ sư phần mềm**
    - **Bản phát hành tiếp theo thứ hai hoặc thứ ba của chuỗi sản phẩm phần mềm “được lên kế hoạch” trong đó yêu cầu đã được thiết lập**
    - **Các yêu cầu được cung cấp như là một phần của yêu cầu báo giá cho một dự án phát triển phần mềm**
  + **Phải được các kỹ sư phần mềm thiết lập**
    - **Người dùng chỉ hiểu được các yêu cầu liên quan đến nhiệm vụ công việc cụ thể của họ**
    - **Lý do và mục tiêu kinh doanh không rõ ràng**
    - **Có thể có các yêu cầu mâu thuẫn và không đầy đủ được người dùng và khách hàng nêu**

# Yêu cầu cấp cao

* **Cần tìm kiếm các nhận thức và mục tiêu kinh doanh và quản lý cho dự án phần mềm**
  + **Cơ hội kinh doanh và nhu cầu kinh doanh**
  + **Sự biện minh cho dự án**
  + **Phạm vi**
  + **Ràng buộc chính**
  + **Chức năng chính**
  + **Yếu tố thành công**
  + **Đặc điểm người dùng**

# 6-Kích thước của yêu cầu chi tiết

* **Yêu cầu**
* **Chức năng cá nhân**
* **Quy trình làm việc**
* **Định dạng dữ liệu và dữ liệu**
* **Giao diện người dùng**
* **Giao diện hiện tại & hệ thống**
* **Các ràng buộc khác: Hiệu suất,**
* **Độ tin cậy, v.v.**

# Phân tích yêu cầu

* **Phân tích yêu cầu bao gồm:**
  + **Phân loại các yêu cầu**
  + **Ưu tiên các yêu cầu**

# Phân loại yêu cầu

* **Theo khu vực yêu cầu chi tiết:**
  + **Chức năng cá nhân**
  + **Dòng chảy kinh doanh**
  + **Nhu cầu dữ liệu và thông tin**
  + **Giao diện người dùng**
  + **Các giao diện khác cho các hệ thống bên ngoài**
  + **Các ràng buộc khác nhau**
    - **Hiệu suất**
    - **Bảo vệ**
    - **độ tin cậy**
    - **Vv**

# Yêu cầu Phân loại (cont.)

* **Sử dụng Case sử dụng của OO, trong đó xác định:**
  + **Chức năng cơ bản**
  + **Pre-điều kiện của chức năng**
  + **Dòng sự kiện hoặc kịch bản**
  + **Các điều kiện sau cho chức năng**
  + **Điều kiện lỗi và luồng thay thế**
* **Sử dụng phương pháp OO Use Case xác định:**
  + **Các diễn viên (hoặc tất cả các giao diện bên ngoài với hệ thống, bao gồm cả người dùng)**
  + **Các trường hợp sử dụng liên quan**
  + **Điều kiện biên**

# Yêu cầu Phân loại (cont.)

* **Xem định nghĩa yêu cầu định hướng (VORD) dựa trên khái niệm rằng các yêu cầu được xem khác nhau bởi những người khác nhau:**
  + **Xác định các bên liên quan và quan điểm của họ về các yêu cầu**
  + **Phân loại các quan điểm của các yêu cầu và loại bỏ bất kỳ sự trùng lặp nào**
  + **Tinh chỉnh các quan điểm được xác định của các yêu cầu**
  + **Lập bản đồ các quan điểm về các yêu cầu đối với hệ thống và các dịch vụ mà hệ thống phải cung cấp**

# Yêu cầu ưu tiên

* **Hầu hết thời gian chúng tôi có những hạn chế của:**
  + **Thời gian**
  + **Tài nguyên**
  + **Năng lực kỹ thuật**
* **Chúng tôi cần ưu tiên các yêu cầu để đáp ứng các giới hạn này**

# Yêu cầu ưu tiên

* **Tiêu chí ưu tiên:**
  + **Nhu cầu hoặc nhu cầu của người dùng / khách hàng hiện tại**
  + **Cạnh tranh và điều kiện thị trường hiện tại**
  + **Nhu cầu khách hàng mới trong tương lai và dự kiến**
  + **Lợi thế bán hàng**
  + **Các vấn đề nghiêm trọng trong sản phẩm hiện tại**
* **Đây thường là chủ quan và yêu cầu cần được ưu tiên với sự giúp đỡ của nhiều bên liên quan.**

# Danh sách ưu tiên yêu cầu đơn giản

* **Req.** **#**
* **Tóm tắt Req.**
* **sự miêu tả**
* **Req.** **nguồn**
* **Req.** **sự ưu tiên**
* **Req.** **Trạng thái**
* **# 1**
* **Một trang**
* **truy vấn phải**
* **trả lời trong**
* **ít hơn**
* **1 giây**
* **Chuyên ngành**
* **tài khoản**
* **Tiếp thị**
* **Đại diện**
* **Mức độ ưu tiên 1 \***
* **Được chấp nhận cho**
* **bản phát hành này**
* **# 2**
* **Văn bản trợ giúp**
* **cần phải**
* **lĩnh vực nhạy cảm**
* **Tài khoản lớn**
* **người dùng**
* **Ưu tiên 2**
* **Đã hoãn**
* **Cho tiếp theo**
* **giải phóng**
* ***\* Mức độ ưu tiên có thể là 1, 2, 3 hoặc 4, với 1 là mức cao nhất***

# Yêu cầu So sánh và ưu tiên

* **Yêu cầu ưu tiên là một hoạt động so sánh các yêu cầu và đặt chúng theo thứ tự tương đối với nhau.**
  + **Đã xong chỉ với trải nghiệm và cảm giác ruột**
  + **Thực hiện với một chút nghiêm khắc hơn:**
    - **Sắp xếp theo nhóm ưu tiên (ví dụ: 2 biểu đồ trước đó) trong đó các nhóm ưu tiên dựa trên một số danh sách tiêu chí ưu tiên**
    - **So sánh khôn ngoan, chuẩn hóa và tính toán giá trị tương đối bằng cách sử dụng Quy trình phân tích phân cấp (AHP) - xem các trang 159-161 của sách giáo khoa**

# Ma trận so sánh cặp đôi

* <số>

|  |  |  |  |
| --- | --- | --- | --- |
|  | * **Req1** | * **Req2** | * **Req3** |
| * Req1 | * 1 | * 3 | * 5 |
| * Req2 | * 1/3 | * 1 | * 1/2 |
| * Req3 | * 1/5 | * 2 | * 1 |

* **Tổng mỗi cột và chia mỗi giá trị trong cột theo tổng**

|  |  |  |  |
| --- | --- | --- | --- |
|  | * **Req1** | * **Req2** | * **Req3** |
| * Req1 | * .65 | * .5 | * .77 |
| * Req2 | * .22 | * .17 | * .08 |
| * Req3 | * .13 | * .33 | * .15 |

# Ma trận so sánh cặp đôi

* <số>
* **Tổng mỗi hàng:**
  + **Hàng 1 (Req1) = 1,92**
  + **Hàng 2 (Req2) = .47**
  + **Hàng 3 (Req) 3 = .61**
* **Chia mỗi số theo số lượng yêu cầu:**
  + **1,92 / 3 = .64**
  + **0,47 / 3 = 0,15**
  + **0,61 / 3 = 0,20**
* **Các kết quả**
  + **Req1 cao nhất (64% tổng giá trị req)**
  + **Req3 highet thứ hai (20%)**
  + **Req2 thấp nhất (15%)**
* **Chỉ tốt cho số lượng nhỏ các yêu cầu**

# Yêu cầu Định nghĩa / Tạo mẫu / Xem lại

* **Một khi các yêu cầu được trưng cầu, phân tích và ưu tiên, phải nêu rõ thêm chi tiết.** **Ba hoạt động chính có thể đan xen phải được thực hiện:**
  + **Yêu cầu định nghĩa**
  + **Yêu cầu tạo mẫu**
  + **Yêu cầu xem xét**

# Yêu cầu

* **Các định nghĩa yêu cầu có thể được viết dưới các hình thức khác nhau:**
  + **Các mô tả đầu vào / quy trình / đầu ra đơn giản bằng tiếng Anh**
  + **Biểu đồ Dataflow (DFD)**
  + **Biểu đồ Quan hệ thực thể (ERD)**
  + **Sử dụng Sơ đồ Ca từ Ngôn ngữ Mô hình Thống nhất (UML)**
* **Khi yêu cầu được xác định chi tiết bằng cách sử dụng bất kỳ biểu mẫu nào ở trên, chúng vẫn cần phải được *xem xét* bởi người dùng / khách hàng và các bên liên quan khác.**

# *Định nghĩa yêu cầu bằng cách sử dụng biểu mẫu sơ đồ đầu vào-quá trình nhập và tiếng Anh*

* **Req.** **#**
* **Đầu vào**
* **Đầu ra**
* **Quá trình**
* **# 12: khách hàng**
* **gọi món**
* **- Mục theo loại**
* **và số lượng**
* **- Gửi lời đề nghị**
* **- Đảm nhận**
* **mục và**
* **tương ứng**
* **số lượng**
* **- Trưng bày**
* **chấp thuận**
* **tin nhắn và**
* **- Hỏi về**
* **xác nhận**
* **thông điệp**

# *Định nghĩa sử dụng DFD*

* **Đơn đặt hàng**
* **Xử lý đơn đặt hàng**
* **Thông tin khách hàng DB**
* **Tín dụng của khách hàng,**
* **địa chỉ, v.v.**
* **Thông tin khoảng không quảng cáo.**
* **Bao bì**
* **Hóa đơn**
* **Sản phẩm avail.**
* **Thông tin.**
* **Dữ liệu gói**
* **Bao bì**
* **chi tiết**
* **Đang chuyển hàng**
* **Dạy.**
* **khách hàng**

# *Yêu cầu Định nghĩa bằng cách sử dụng Entity-Relation-Diagram (ERD)*

* **Tác giả**
* **Sách**
* **viết**
* **1**
* **m**
* ***Cardinality*** : xác định số lần xuất hiện của các thực thể
* **Tác giả**
* **Sách**
* ***Phương thức*** : xác định nhu cầu cần thiết của mối quan hệ tồn tại

# *Yêu cầu Định nghĩa xác định thực thể và thuộc tính*

* **Nhân viên**
* **Địa chỉ nhà**
* **Tên**
* **Tuổi tác**
* **đường phố**
* **Thành phố**
* **Tiểu bang**
* **Zip**
* **(a) Hình thức đồ họa**
* **(a) dạng bảng**
* **Nhân viên**
* **- Tên**
* **- Địa chỉ nhà**
* **- Tuổi tác**
* **- Đường phố**
* **- Thành phố**
* **- Tiểu bang**
* **- Zip**
* **Đăng ký phần**
* **Thêm khóa học**
* **Thêm mục**
* **Thêm sinh viên**
* **Chọn mục**
* ***Yêu cầu Định nghĩa bằng cách sử dụng Sơ đồ Ca sử dụng***
* **Sinh viên**
* **Nhà đăng ký**

# Yêu cầu tạo mẫu

* **Yêu cầu prototyping chủ yếu là địa chỉ giao diện người dùng (UI) một phần của yêu cầu về:**
  + ***Hình ảnh (kích thước, hình dạng, vị trí, màu sắc)***
  + ***Luồng (kiểm soát và luồng logic)***
* **Việc tạo mẫu có thể được thực hiện ở một trong hai chế độ:**
  + **Độ trung thực thấp** : sử dụng giấy / bìa cứng để thể hiện màn hình và con người để di chuyển bảng
  + **Độ trung thực cao** : sử dụng các công cụ tự động như Visual Basic để mã hóa màn hình và hướng luồng logic của các màn hình này

# Yêu cầu kỹ thuật

* **Một khi các yêu cầu được xác định, tạo mẫu và xem xét, nó phải được đặt vào tài liệu Yêu cầu Đặc tả**
* **Có thể mua IEEE / EIA Standard 12207.1-1997 từ IEEE.** **Nó phác thảo hướng dẫn cho 3 phần chính của tài liệu đặc tả yêu cầu.**
  + **Giới thiệu**
  + **Mô tả cấp cao**
  + **Mô tả chi tiết**
    - **Chi tiết của từng chức năng (quá trình nhập-xuất)**
    - **Giao diện, bao gồm giao diện người dùng và giao diện mạng**
    - **Yêu cầu hiệu suất (thời gian đáp ứng, thông lượng, v.v.)**
    - **Các ràng buộc, tiêu chuẩn thiết kế, v.v.**
    - **Các thuộc tính bổ sung như bảo mật, độ tin cậy, v.v.**
    - **Bất kỳ yêu cầu duy nhất bổ sung nào**

# Phần mềm Yêu cầu kỹ thuật

* <số>
* **Đặc tả yêu cầu phần mềm (SRS) quy định các yêu cầu cho một mục cấu hình phần mềm máy tính (CSCI) và các phương thức được sử dụng để đảm bảo rằng mỗi yêu cầu đã được đáp ứng.Các yêu cầu liên quan đến các giao diện bên ngoài CSCIs có thể được trình bày trong SRS hoặc trong một hoặc nhiều đặc tả yêu cầu giao diện (IRS).**
* **1.0 Giới thiệu về tài liệu**
  + **1.1 Mục đích của dự án**
* *Nêu rõ mục đích của hệ thống hoặc hệ thống phụ mà tài liệu này áp dụng.*
  + **1.2 Phạm vi dự án**
  + **1.3 Từ viết tắt, Chữ viết tắt, Định nghĩa**
  + **1.4 Tài liệu tham khảo**
  + **1.5 Phác thảo phần còn lại của SRS**
* **2.0 Mô tả chung về sản phẩm**
  + **2.1 Bối cảnh sản phẩm**
  + **2.2 Chức năng sản phẩm**
  + **2.3 Đặc điểm người dùng**
  + **2.4 Hạn chế**
  + **2.5 Giả định và phụ thuộc**

# Phần mềm Yêu cầu kỹ thuật

* <số>
* **3.0 Yêu cầu cụ thể**
  + **3.1 Yêu cầu giao diện bên ngoài**
    - **3.1.1 Giao diện người dùng**
    - **3.1.2 Giao diện phần cứng**
    - **3.1.3 Giao diện phần mềm**
    - **3.1.4 Giao diện truyền thông**
  + **3.2 Yêu cầu chức năng**
    - **3.2.1 Lớp 1**
      * **3.2.1.1 Câu ngắn, bắt buộc nêu rõ yêu cầu chức năng được xếp hạng cao nhất.**
      * **3.2.1.2 Mô tả Mô tả đầy đủ về yêu cầu.**
      * **3.2.1.3 Tính phê phán Mô tả mức độ cần thiết của yêu cầu này đối với hệ thống tổng thể.**
      * **3.2.1.4 Các vấn đề kỹ thuật Mô tả mọi vấn đề thiết kế hoặc thực hiện liên quan đến việc đáp ứng yêu cầu này.**
      * **3.1.2.5 Chi phí và lịch biểu Mô tả chi phí tương đối hoặc tuyệt đối liên quan đến vấn đề này.**

# Phần mềm Yêu cầu kỹ thuật

* <số>
  + - * **3.2.1.6 Rủi ro Mô tả các trường hợp mà theo đó yêu cầu này có thể không được thỏa mãn, và những hành động nào có thể được thực hiện để giảm xác suất xảy ra này.**
      * **3.1.2.7 Các phụ thuộc với các yêu cầu khác Mô tả các tương tác với các yêu cầu khác.**
      * **3.1.2.8 ... những người khác thích hợp**
    - **3.2.2 Lớp 2**
    - **3.2.3 Lớp 3**
* **.** **.** **.**
  + **3.3 Yêu cầu về hiệu suất**
  + **3.4 Ràng buộc thiết kế**
  + **3.5 Yêu cầu chất lượng**
  + **3.6 Yêu cầu an toàn**
  + **3.7 Yêu cầu an toàn và an toàn**
  + **3.8 Yêu cầu tài nguyên máy tính**
  + **3.9 Yêu cầu nhân sự**
  + **3,10 Đào tạo Yêu cầu liên quan**
  + **3.11 Logistics Yêu cầu liên quan**
  + **3.12 Yêu cầu khác**

# Phần mềm Yêu cầu kỹ thuật

* <số>
* **4.0 Ghi chú**
  + **4.1 Mục đích sử dụng**
  + **4.2 Định nghĩa được sử dụng trong tài liệu này**
  + **4,3 từ viết tắt được sử dụng trong tài liệu này**
  + **4.4 Thay đổi từ vấn đề trước**
* **5.0 Phụ lục**

# Yêu cầu “Đăng xuất”

* **Có một đặc điểm kỹ thuật yêu cầu đã đồng ý và ký kết là rất quan trọng:**
  + **Phục vụ như một điểm mốc quan trọng và chính thức thoát khỏi một giai đoạn của kỹ nghệ phần mềm**
  + **Phục vụ như là đường cơ sở mà từ đó bất kỳ thay đổi nào trong tương lai có thể được theo dõi và kiểm soát**

# Yêu cầu Truy xuất nguồn gốc

* **Khả năng theo dõi một yêu cầu là cần thiết để đảm bảo rằng sản phẩm đã thực hiện đầy đủ các yêu cầu.**
* **Chúng ta cần phải theo dõi các yêu cầu:**
  + **Yêu cầu *từ* nguồn (con người và tài liệu)**
  + **Yêu cầu đối *với* các bước sau (thiết kế, triển khai)**
* **Chúng tôi cũng cần phải liên kết các yêu cầu với các yêu cầu “tiền điều kiện tiên quyết” khác.**

# Bảng truy nguyên được điền một phần

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| * ***Yêu cầu*** | * ***Thiết kế*** | * ***Mã số*** | * ***Kiểm tra*** | * ***Khác*** * ***Reqs.*** |
| * **1** | * **comp.** **X** | * **Mô-đun 3** | * **Trường hợp thử nghiệm 32** | * **2,3** |
| * **2** | * **comp.** **w** | * **Học phần 5** | * **Trường hợp thử nghiệm 16** | * **1** |
| * **3** | * **comp.** **X** | * **Học phần 2** | * **Trường hợp thử nghiệm 27** | * **1** |
| * **4** |  |  |  |  |
| * **5** |  |  |  |  |
| * **6** |  |  |  |  |
| * **7** |  |  |  |  |

CHUONG 7

* Chương 7
* **Thiết kế: Kiến trúc và phương pháp luận**
* Mùa thu 2011

# Mục tiêu

* Kiến trúc và thiết kế chi tiết
* Các kiểu kiến ​​trúc, chiến thuật và kiến ​​trúc tham chiếu chung
* Kỹ thuật cơ bản cho thiết kế chi tiết
* Các vấn đề cơ bản với thiết kế giao diện người dùng

# Giới thiệu

* Bắt đầu từ yêu cầu
* Phần mềm sẽ được cấu trúc như thế nào
  + Các thành phần là gì
  + Các thành phần này có liên quan như thế nào
* Hai phần
  + Kiến trúc (cao cấp)
  + Thiết kế chi tiết
* Ký hiệu
* **Mối quan hệ giữa Kiến trúc và Thiết kế**
* **Thiết kế chi tiết**
* **đến từ**
* **yêu cầu &**
* **kiến trúc**

# Kiến trúc phần mềm

* Cấu trúc của hệ thống, bao gồm:
  + Phần mềm
  + Thuộc tính hiển thị bên ngoài của chúng
  + Mối quan hệ giữa các phần tử
* Mỗi hệ thống đều có kiến ​​trúc
* Nhiều cấu trúc
  + nhiều cách tổ chức các yếu tố
* Thuộc tính bên ngoài của mô-đun

# Lý lịch…

* Kiến trúc phần mềm
* <số>
* Kiến trúc là thiết kế hệ thống ở mức cao nhất
* Các lựa chọn về công nghệ, sản phẩm để sử dụng, máy chủ, vv được thực hiện ở cấp kiến ​​trúc
  + Không thể thiết kế chi tiết hệ thống và sau đó điều chỉnh các lựa chọn này
  + Kiến trúc phải được tạo sức chứa chúng
* Là nơi sớm nhất khi các đặc tính như hiệu suất tin cậy có thể được đánh giá

# Kiến trúc

* Kiến trúc phần mềm
* <số>
* Kiến trúc là một thiết kế của sw cung cấp một cái nhìn rất cao về các bộ phận và chúng liên quan như thế nào để tạo thành toàn bộ
  + Phân vùng hệ thống thành các phần sao cho mỗi phần có thể được hiểu một cách độc lập
  + Và mô tả mối quan hệ giữa các bộ phận
* Một hệ thống phức tạp có thể được phân chia theo nhiều cách khác nhau, mỗi cách cung cấp một khung nhìn hữu ích
  + Tương tự cũng đúng đối với phần mềm
  + Không có cấu trúc độc nhất; nhiều khả năng

# Kiến trúc

* Kiến trúc phần mềm
* <số>
* Định nghĩa: Kiến trúc phần mềm là cấu trúc hoặc cấu trúc bao gồm các phần tử, các thuộc tính hiển thị bên ngoài của chúng và các mối quan hệ giữa chúng
  + Các phần tử: chỉ quan tâm đến các đặc tính bên ngoài cần thiết cho đặc tả mối quan hệ
  + Chi tiết về cách các thuộc tính được hỗ trợ không quan trọng đối với kiến ​​trúc
  + Định nghĩa không nói bất cứ điều gì về việc liệu một kiến ​​trúc có tốt hay không - phân tích cần thiết cho điều đó
* Một kiến ​​trúc mô tả các cấu trúc khác nhau của hệ thống

# Sử dụng chính của kiến ​​trúc

* Kiến trúc phần mềm
* <số>
* Hiểu và giao tiếp
  + Bằng cách hiển thị một hệ thống ở mức cao và che giấu sự phức tạp của các phần, mô tả kiến ​​trúc tạo điều kiện cho giao tiếp
  + Để có được sự hiểu biết chung giữa các bên liên quan khác nhau (người dùng, khách hàng, kiến ​​trúc sư, nhà thiết kế, người triển khai, người kiểm tra)
  + Đối với thương lượng và thỏa thuận
  + Mô tả kiến ​​trúc cũng có thể giúp hiểu về các hệ thống hiện có

# Sử dụng ...

* Kiến trúc phần mềm
* <số>
* Tái sử dụng
  + Một phương pháp tái sử dụng là soạn các hệ thống từ các bộ phận và tái sử dụng các phần hiện có
  + Mô hình này được tạo điều kiện bằng cách sử dụng lại các thành phần ở mức cao cung cấp các dịch vụ hoàn chỉnh
  + Để sử dụng lại các thành phần hiện có, kiến ​​trúc phải được chọn sao cho các thành phần này phù hợp với các thành phần khác
  + Do đó, quyết định về việc sử dụng các thành phần hiện có được thực hiện tại thời điểm thiết kế kiến ​​trúc

# Sử dụng ..

* Kiến trúc phần mềm
* <số>
* Xây dựng và tiến hóa
  + Một số cấu trúc trong mô tả kiến ​​trúc sẽ được sử dụng để hướng dẫn phát triển hệ thống
  + Việc phân vùng ở cấp kiến ​​trúc cũng có thể được sử dụng để phân bổ công việc cho các nhóm vì các phần tương đối độc lập
  + Trong quá trình tiến hóa sw, kiến ​​trúc giúp quyết định những gì cần phải được thay đổi để kết hợp các thay đổi / tính năng mới
  + Kiến trúc có thể giúp quyết định tác động của những thay đổi đối với các thành phần hiện có trên các thành phần khác

# Sử dụng ...

* Kiến trúc phần mềm
* <số>
* Phân tích
  + Nếu các thuộc tính như hiệu suất và độ tin cậy có thể được xác định từ thiết kế, các lựa chọn thay thế có thể được xem xét trong quá trình thiết kế để đạt được mức hiệu suất mong muốn
  + Kiến trúc Sw mở ra các khả năng như vậy cho phần mềm (các ngành kỹ thuật khác thường có thể làm được điều này)
  + Ví dụ: độ tin cậy và hiệu suất của một hệ thống có thể được dự đoán từ kiến ​​trúc của nó, nếu ước tính cho các thông số như tải được cung cấp
  + Sẽ yêu cầu mô tả chính xác về kiến ​​trúc, cũng như các thuộc tính của các phần tử trong mô tả

# Lượt xem và quan điểm

* Xem - Biểu diễn cấu trúc hệ thống
* 4 + 1 lượt xem
  + Hợp lý (phân tách OO)
  + Quy trình (thành phần thời gian chạy)
  + Phân tích hệ thống phụ
  + Kiến trúc vật lý
  + +1: trường hợp sử dụng
* Phân loại khác (Bass, Clements, Kazman)
  + Mô-đun
  + Thời gian chạy
  + Phân bổ
* Quan điểm khác nhau cho những người khác nhau

# Lý lịch…

* Kiến trúc phần mềm
* <số>
* Kiến trúc là thiết kế hệ thống ở mức cao nhất
* Các lựa chọn về công nghệ, sản phẩm để sử dụng, máy chủ, vv được thực hiện ở cấp kiến ​​trúc
  + Không thể thiết kế chi tiết hệ thống và sau đó điều chỉnh các lựa chọn này
  + Kiến trúc phải được tạo sức chứa chúng
* Là nơi sớm nhất khi các đặc tính như hiệu suất tin cậy có thể được đánh giá

# Lượt xem…

* Kiến trúc phần mềm
* <số>
* Nhiều loại lượt xem đã được đề xuất
* Hầu hết thuộc một trong ba loại này
  + Mô-đun
  + Thành phần và trình kết nối
  + Phân bổ
* Các quan điểm khác nhau không phải là không liên quan - tất cả chúng đều đại diện cho cùng một hệ thống
  + Có mối quan hệ giữa các yếu tố của các quan điểm khác nhau; các mối quan hệ có thể phức tạp

# Lượt xem…

* Kiến trúc phần mềm
* <số>
* Chế độ xem mô-đun
  + Một hệ thống là tập hợp các đơn vị mã tức là chúng không đại diện cho các thực thể thời gian chạy
  + Các yếu tố tức là các mô-đun, ví dụ: lớp, gói, hàm, thủ tục, phương thức,…
  + Mối quan hệ giữa chúng là dựa trên mã
    - Phụ thuộc vào cách mã của mô-đun tương tác với mô-đun khác
    - Ví dụ về mối quan hệ
      * “Mô-đun A là một phần của mô-đun B”
      * “Mô-đun A phụ thuộc vào các dịch vụ của mô-đun B”
      * “Mô-đun B là sự tổng quát hóa mô-đun A”

# Phong cách kiến ​​trúc

* Ống và Bộ lọc
* Event-Driven
* Client-Server
* Model-View-Controller (MVC)
* Xếp lớp
* Cơ sở dữ liệu Centric
* Ba cấp

# Ống và lọc

* Kiến trúc phần mềm
* <số>
* Rất thích hợp cho các hệ thống chủ yếu thực hiện chuyển đổi dữ liệu
* Một hệ thống sử dụng kiểu này sử dụng mạng chuyển đổi để đạt được kết quả mong muốn
* Có một loại thành phần - bộ lọc
* Có một loại đầu nối - ống
* Bộ lọc thực hiện một số phép chuyển đổi và chuyển dữ liệu đến các bộ lọc khác thông qua đường ống

# Ống và bộ lọc…

* Kiến trúc phần mềm
* <số>
* Một bộ lọc độc lập; không cần biết id của bộ lọc gửi / nhận dữ liệu
* Bộ lọc có thể không đồng bộ và là nhà sản xuất hoặc người tiêu dùng dữ liệu
* Một đường ống là một chiều kênh chuyển luồng dữ liệu từ bộ lọc này sang bộ lọc khác
* Một đường ống là một kết nối 2 chiều
* Bộ lọc phải thực hiện lưu vào bộ đệm và đồng bộ hóa giữa các bộ lọc

# Ống và bộ lọc…

* Kiến trúc phần mềm
* <số>
* Bộ lọc nên hoạt động mà không cần biết xác định nhà sản xuất / người tiêu dùng
* Một đường ống phải kết nối cổng đầu ra của một bộ lọc với cổng đầu vào của một bộ lọc khác
* Bộ lọc có thể có chuỗi điều khiển độc lập

# Thí dụ

* Kiến trúc phần mềm
* <số>
* Một hệ thống cần thiết để đếm tần số của các từ khác nhau trong một tập tin
* Một cách tiếp cận: đầu tiên chia tệp thành chuỗi các từ, sắp xếp chúng, sau đó đếm số lần xuất hiện
* Kiến trúc của hệ thống này có thể tự nhiên sử dụng kiểu ống và bộ lọc

# Thí dụ..

* Kiến trúc phần mềm
* <số>

# Kiểu ống và bộ lọc

# Kiểu máy khách-máy chủ

* Kiến trúc phần mềm
* <số>
* Hai loại thành phần - máy khách và máy chủ
* Khách hàng chỉ có thể giao tiếp với máy chủ, nhưng không chỉ với các khách hàng khác
* Giao tiếp được bắt đầu bởi một khách hàng gửi yêu cầu và máy chủ phản hồi
* Một loại trình kết nối - yêu cầu / trả lời, không đối xứng
* Thường thì máy khách và máy chủ nằm trên các máy khác nhau

# Kiểu máy khách-chủ…

* Kiến trúc phần mềm
* <số>
* Một hình thức chung của phong cách này là cấu trúc n-tier
* Cấu trúc 3 tầng thường được sử dụng bởi nhiều hệ thống ứng dụng và web
  + Client-tier chứa các máy khách
  + Tầng giữa chứa quy tắc kinh doanh
  + Tầng cơ sở dữ liệu có thông tin

# Kiểu máy khách-máy chủ cơ bản

# Kiểu máy khách-máy chủ

* Khách hàng có thể kết nối với nhiều máy chủ

# Phong cách MVC

* **Mô hình**
* **Bộ điều khiển 1**
* **Xem 1**
* **Bộ điều khiển 2**
* **Xem 2**
* Phân tách mô hình (dữ liệu) khỏi chế độ xem
* Bộ điều khiển tích hợp với chế độ xem ngày nay

# Layered Style

* **Hình 7.5: Kiểu lớp.** **Triển khai Java API**
* **gọi API OS mà lần lượt gọi các hàm hạt nhân**

# Kiểu dữ liệu được chia sẻ

* Kiến trúc phần mềm
* <số>
* Hai loại thành phần - kho dữ liệu và trình truy cập dữ liệu
* Kho dữ liệu - cung cấp bộ nhớ vĩnh viễn đáng tin cậy
* Truy cập dữ liệu - truy cập dữ liệu trong kho, thực hiện tính toán và có thể đưa kết quả trở lại
* Giao tiếp giữa những người truy cập dữ liệu chỉ thông qua kho lưu trữ

# Kiểu dữ liệu được chia sẻ ...

* Kiến trúc phần mềm
* <số>
* Hai biến thể có thể
  + Kiểu bảng đen: nếu dữ liệu được đăng trong một kho lưu trữ, tất cả các người truy cập đều được thông báo; tức là nguồn dữ liệu được chia sẻ là một tác nhân hoạt động
  + Kiểu kho lưu trữ: kho lưu trữ thụ động
* Ví dụ. hệ thống định hướng cơ sở dữ liệu; hệ thống web; môi trường lập trình, ..

# Thí dụ

* Kiến trúc phần mềm
* <số>
* Hệ thống đăng ký sinh viên của một trường đại học
* Kho lưu trữ chứa tất cả các dữ liệu về sinh viên, các khóa học, lịch trình,…
* Người truy cập như quản trị, phê duyệt, đăng ký, báo cáo thực hiện các thao tác trên dữ liệu

# Thí dụ…

* Kiến trúc phần mềm
* <số>

# Thí dụ..

* Kiến trúc phần mềm
* <số>
* Các thành phần không trực tiếp giao tiếp với nhau
* Dễ mở rộng - nếu cần một bộ lập lịch, nó sẽ được thêm vào như một bộ truy cập mới
  + Không cần thay đổi thành phần hiện có
* Chỉ có một kiểu kết nối trong này - đọc / ghi

# Kiểu trung tâm cơ sở dữ liệu

* DB
* Client1a
* Client1b
* Client2

# Phong cách ba tầng

* DB
* Client1a
* Client1b
* Client2
* Kinh doanh
* Tầng
* Khách hàng không truy cập trực tiếp vào DB
* Tính linh hoạt, toàn vẹn

# Chiến thuật kiến ​​trúc

* Giải quyết các vấn đề nhỏ hơn, cụ thể
* Không ảnh hưởng đến cấu trúc tổng thể của hệ thống
* Ví dụ: Phát hiện lỗi phân tán
  + Nhịp tim
  + Ping / echo

# Kiến trúc tham khảo

* Các kiến ​​trúc chính thức
* Mẫu cho một lớp hệ thống
* Ví dụ: Kiến trúc tham khảo J2EE (MVC2)
* DB
* Client1a
* Client1b
* Client2
* Web
* Tầng
* Kinh doanh
* Bậc (EJB)

# Thiết kế chi tiết

* Tinh chỉnh Kiến trúc và khớp với Yêu cầu
* Chi tiết như thế nào?
* Làm thế nào chính thức?
* Có thể mức độ chi tiết khác nhau cho các chế độ xem khác nhau

# Suy giảm chức năng

* Chủ yếu cho lập trình có cấu trúc (nay là hệ thống Web)
* Ban đầu: module cho mỗi nhiệm vụ / yêu cầu
* Tinh chỉnh thành các mô-đun phụ
* Có những phân tách thay thế

# Phân tích mô-đun có thể có

* **0. Chính**
* **1.Student**
* **1.3 Xóa**
* **1.2 Sửa đổi**
* **1.1 Thêm**
* **2.Giáo dục**
* **3. Mục**
* **4. Đăng ký**
* **2.1 Thêm**
* **3.1 Thêm**
* **4.1 Đăng ký**
* **2.2 Sửa đổi**
* **3.2 Sửa đổi**
* **4,2 thả**
* **2.3 Xóa**
* **3.3 Xóa**

# Phân tích thay thế

# Thiết kế cơ sở dữ liệu quan hệ

* Hầu hết các cơ sở dữ liệu sử dụng công nghệ quan hệ
* Quan hệ (bảng)
  + Bộ hai chiều
  + Hàng (tuple), Cột (thuộc tính)
  + Hàng có thể là thực thể, mối quan hệ hoặc thuộc tính
  + Khóa chính
  + Khóa ngoại

# Thiết kế cơ sở dữ liệu

* Mô hình khái niệm (được thực hiện trong giai đoạn phân tích / yêu cầu) tạo ra sơ đồ ER
* Thiết kế hợp lý (quan hệ)
* Thiết kế vật lý (quyết định kiểu dữ liệu, v.v ...)
* Triển khai / bảo trì
  + Vật lý ở mức độ thấp (ổ đĩa cứng, v.v ...)
  + Điều chỉnh chỉ mục

# Sơ đồ quan hệ thực thể

* Thực thể (hình chữ nhật)
  + Yếu - đường đôi
* Mối quan hệ (kim cương)
* Thuộc tính (hình bầu dục)
  + Đa giá trị - đường đôi
  + Xác định - gạch chân

# Sơ đồ ER

# Logical DB Design- Thực thể

* Bảng cho mỗi thực thể
* Làm phẳng các thuộc tính tổng hợp
* Đối với các thực thể yếu, hãy thêm khóa chính của thực thể mạnh
* **Con số**
* **Chức vụ**
* **CreditHours**
* **Số của khóa học**
* **Số phần**
* **Học kỳ**
* **Năm**
* **Thời gian**
* **Vị trí**
* **Khóa học**
* **Phần**

# Thiết kế DB hợp lý - Đa giá trị

* Bảng mới cần thiết cho các thuộc tính đa giá trị

# Thiết kế DB hợp lý - Các mối quan hệ

* Nếu một bên liên quan đến chỉ một thực thể, hãy thêm khóa ngoại vào bên đó
* Đối với nhiều người, cần có bảng mới
* Đối với ternary, cần bảng mới

# Thiết kế DB vật lý

* Kiểu dữ liệu cho mỗi thuộc tính
  + Kiểm tra xem hỗ trợ DBMS của bạn
  + Mã hóa
* Quyết định chỉ mục
  + Tìm kiếm nhanh hơn, cập nhật chậm hơn
  + Chỉ mục tiêu thụ không gian
  + Luôn có thể điều chỉnh trong khi triển khai
* Đôi khi không chuẩn hóa (tránh)

# Thiết kế OO

* Quyết định
  + Lớp học nào cần tạo
  + Họ có liên quan với nhau như thê nào
* UML
* Bước đầu tiên: Tinh chỉnh các trường hợp sử dụng

# Sử dụng sơ đồ trường hợp

* **Đăng ký phần**
* **Thêm khóa học**
* **Thêm mục**
* **Thêm sinh viên**
* **Chọn mục**
* **Sinh viên**
* **Nhà đăng ký**

# Thiết kế lớp học

* Đối tượng đại diện cho các thực thể hoặc khái niệm hệ thống trong thế giới thực
* Được tổ chức thành các lớp học. Các đối tượng trong lớp có đặc điểm tương tự
* Đối tượng có thuộc tính (thuộc tính)
* Các đối tượng cũng có các phương thức
* **Sinh viên**
* **dateOfBirth: Ngày**
* **tên: Chuỗi**
* **getAgeInYears (): int**
* **getAgeInDays (): int**

# Sơ đồ lớp UML

* **Sinh viên**
* **Trường học**
* **0 .. \***
* **1..1**
* **Được đăng ký**
* Hiệp hội
* Thành phần
* **Sinh viên**
* **Địa chỉ nhà**
* **streetName: Chuỗi**
* **streetNumber: int**
* **thành phố: String**
* **trạng thái: Chuỗi**
* **Mã zip: int**

# Sơ đồ lớp UML - Thừa kế

* **Sinh viên**
* **Người**
* **Nhân viên**

# Sơ đồ trạng thái UML

* **Đã chấp nhận**
* **Hoạt động**
* **Cựu sinh viên**
* **Bị trục xuất**
* **Không hoạt động**
* **ghi danh:**
* **tốt nghiệp:**
* **expell:**
* **thất bại**
* **ghi danh:**
* **ghi danh:**

# Biểu đồ trình tự UML

# Thiết kế giao diện người dùng

* Rõ ràng nhất đối với người dùng
* Hai vấn đề chính
  + Luồng tương tác
  + Xem và cảm nhận
* Các loại giao diện
  + Dòng lệnh
  + Menu văn bản
  + Đồ họa (GUI)

# Mô hình GOMS

* Xem xét các loại người dùng khác nhau
* Bốn yếu tố
  + Những mục tiêu
  + Hoạt động
  + Phương pháp
  + Quy tắc lựa chọn

# Các vấn đề về giao diện người dùng khác

* Các loại người dùng
* Heuristics
* Nguyên tắc giao diện người dùng
* Các vấn đề đa văn hóa
* Ẩn dụ
* Phần mềm đa nền tảng
* Trợ năng
* Giao diện đa phương tiện

# Trở kháng đối tượng-quan hệ không khớp

* **Elem 1**
* **Elem 2**
* **Elem 3**
* **4 phút 4**
* **Elem 5**
* **Bảng quan hệ**
* **Đối tượng đóng gói**
* ***?***
* **Làm cách nào để chúng tôi xử lý sự không khớp giữa các khái niệm hướng đối tượng và**
* **DB quan hệ như:**
* **- đánh máy**
* **- riêng tư và công cộng**
* **- thừa kế và đa hình**
* **- Cấu trúc lồng nhau so với cấu trúc bảng**

CHUONG 8

* *Chương 8*
* **Thiết kế đặc điểm và số liệu**
* *Mùa thu 2011*

# Đặc trưng thiết kế tốt

* **Bên cạnh đó thiết kế rõ ràng - - - phải phù hợp với yêu cầu - - - có hai đặc điểm cơ bản:**
  + **C onsistency qua thiết kế:**
    - **Giao diện người dùng chung**
      * **nhìn**
      * **Dòng chảy tự nhiên**
    - **Xử lý lỗi phổ biến**
    - **Báo cáo chung**
    - **Giao diện hệ thống phổ biến**
    - **Trợ giúp chung**
    - **Tất cả thiết kế được mang đến cùng một mức độ sâu**
  + **Hoàn thành** thiết kế
    - **Tất cả các yêu cầu được tính cho**
    - **Tất cả các phần của thiết kế được mang đến khi hoàn thành, đến cùng một mức độ sâu**

# Intuitively Complexity liên quan đến thiết kế “Good”

* **Một số đặc điểm di truyền của thiết kế phức tạp**
  + **Chỉ số phức tạp Halstead**
  + **Chỉ số độ phức tạp của Cyclomatic McCabe**
  + **Số liệu luồng thông tin Henry-Kafura**
  + **Card và thiết kế Glass phức tạp metrcis**

# Chỉ số Halstead

* **Phát triển bởi Maurice Halstead của Purdue trong những năm 1970 để chủ yếu phân tích sự phức tạp của mã nguồn chương trình.**
* **Đã sử dụng 4 đơn vị đo lường cơ bản từ mã:**
  + **n1 = số toán tử riêng biệt**
  + **n2 = số toán hạng riêng biệt**
  + **N1 = tổng của tất cả các lần xuất hiện của n1**
  + **N2 = tổng của tất cả các lần xuất hiện của n2**
* **Từ vựng chương trình, n = n1 + n2**
* **Độ dài chương trình, N = N1 + N2**
* **Sử dụng những điều này, anh đã xác định 4 chỉ số:**
  + **Âm lượng, V = N \* (Log2 n)**
  + **Khối lượng tiềm năng, V @ = (2 + n2 @) log2 (2 + n2 @)**
  + **Cấp độ thực hiện chương trình, L = V @ / V**
  + **Nỗ lực, E = V / L**
* **Chỉ số Halstead thực sự chỉ đo lường sự phức tạp từ vựng, thay vì cấu trúc**
* **sự phức tạp của mã nguồn.**
* *Khối lượng của hàm phải>> 20 và <1000*
* *Khối lượng của một chức năng một dòng không có tham số không rỗng là khoảng 20;* *một volume> 1000 cho biết hàm có thể làm quá nhiều thứ*
* *Khối lượng của một tập tin nên> 100 & <8000*
* *01 lớp công khai Class1 {*
* *02 public void x (boolean v) {*
* *03 int i;*
* *04 nếu (v) {i = 1;}*
* *05 người khác {i = 2;}*
* *06 công tắc (i) {*
* *07 trường hợp 1:*
* *08 trường hợp 2:*
* *09 mặc định :;*
* *10}*
* *11 thử {*
* *12 trong khi (v) {*
* *13 v = sai;*
* *14 int r = 1;*
* *15 Boolean b = new Boolean (true);*
* *16 i = i \* i + r;*
* *17 giờ nghỉ;*
* *18}*
* *19} catch (Exception e) {*
* *20 ném e;*
* *21}*
* *22}*
* *23}*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| * **Hàng** | * **N1** | * **n1** | * **N2** | * **n2** |
| * 04 | * if, = | * if, = | * v, i, 1 | * v, i, 1 |
| * 05 | * = |  | * i, 2 | * 2 |
| * 06 | * công tắc điện | * công tắc điện | * tôi |  |
| * 12 | * vòng lặp | * vòng lặp | * v |  |
| * 13 | * = |  | * v, sai | * sai |
| * 14 | * = |  | * 1 |  |
| * 15 | * =, mới | * Mới | * thật | * thật |
| * 16 | * =, \*, + | * \*, + | * tôi, tôi, tôi, r | * r |
| * 20 | * ném | * ném | * e | * e |
| * **Toàn bộ** | * **13** | * **số 8** | * **16** | * **số 8** |

* ***Chỉ số Halstead - Ví dụ***
* *N1: Mỗi lần cùng một toán tử*
* *n1: Lần đầu tiên toán tử xuất hiện*
* *N2: Mỗi lần cùng một toán hạng xảy ra*
* *n2 :: Lần đầu tiên toán hạng xuất hiện*

# Chỉ số phức tạp của Cyclomatic TJ McCabe dựa trên niềm tin rằng chất lượng chương trình liên quan đến sự phức tạp của luồng điều khiển chương trình .

* **Chỉ số phức tạp của Cyclomatic TJ McCabe dựa trên niềm tin rằng chất lượng chương trình liên quan đến sự phức tạp của luồng điều khiển chương trình .**
* **n1**
* **n2**
* **n3**
* **n5**
* **n6**
* **n4**
* **e1**
* **e2**
* **e7**
* **e3**
* **e 4**
* **e5**
* **e6**
* **Vùng 1**
* **Vùng 2**
* ***Độ phức tạp của Cyclomatic = E - N + 2p***
* ***trong đó E = số cạnh***
* ***N = số nút***
* ***p = số kết nối***
* ***thành phần (thường là 1)***
* ***Vì vậy, đối với luồng điều khiển này:***
* ***7 cạnh - 6 nút + 2 = 3***
* **Số comlexity cyclomatic cũng có thể**
* **được tính như sau:**
* **- số lượng quyết định nhị phân +1**
* **- số vùng đóng + 1**
* ***Độ phức tạp của Cyclomatic Cyclomatic***

# Độ phức tạp của Cyclomatic Cyclomatic

* Các biện pháp phức tạp về thiết kế kết cấu
* Áp dụng cho thiết kế và phân tích rủi ro mã
* Cũng được sử dụng để xác định số lượng các trường hợp thử nghiệm cần thiết để lái xe qua các đường dẫn độc lập tuyến tính trong hệ thống
* Lớn hơn số lượng, nhiều rủi ro hơn, yêu cầu kiểm tra nhiều hơn
  + 1 - 10: rủi ro thấp và đơn giản
  + > 50: nguy cơ cao
* Giữ luồng kiểm soát để độ phức tạp chu trình nhỏ hơn 50

# Henry-Kafura (Fan-in và Fan-out)

* **Số liệu của Henry và Kafura đo lưu lượng liên mô-đun, bao gồm:**
  + **Chuyển thông số**
  + **Truy cập biến toàn cục**
  + **đầu vào**
  + **đầu ra**
* **Fan-in** : số luồng mô-đun liên kết vào chương trình
* **Fan-out** : số lượng mô-đun liên tục ra khỏi chương trình
* **Độ phức tạp của mô-đun, Cp = (fan-in x fan-out) 2**
* **ví dụ ở trên: Cp = (3 + 1) 2 = 16**
* **Module, P**

# Henry-Kafura (Fan-in và Fan-out)

* Đo lưu lượng liên tục
* Dựa trên luồng thông tin vào và ra khỏi mô-đun
* Bao gồm
  + Chuyển thông số
  + Truy cập biến toàn cục
  + Đầu vào
  + Đầu ra

# Henry-Kafura (Fan-in và Fan-out)

* Tổng độ phức tạp của cấu trúc Henry-Kafura
  + Tổng hợp cấu trúc phức tạp của Henry-Kafura cho tất cả các mô-đun chương trình.
  + CT = ∑Cpi
* *i = 1*
* *n*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| * **Mô-đun** | * **Mod-A** | * **Mod-B** | * **Mod-C** | * **Mod-D** |
| * Người hâm mộ | * 3 | * 4 | * 2 | * 2 |
| * Quạt ra | * 1 | * 2 | * 3 | * 2 |

* *Cp cho Mod-A = (3 \* 1) 2 = 9*
* *Cp cho Mod-B = (4 \* 2) 2 = 64*
* *Cp cho Mod-C = (2 \* 3) 2 = 36*
* *Cp cho Mod-D = (2 \* 2) 2 = 16*
* *CT cho toàn bộ chương trình = 125*
* *Cp = (fan-in X fan-out) 2*

# Thiết kế tiềm năng hoặc những khó khăn trong thực hiện Dự đoán của Henry-Kafura

* Cấu trúc dữ liệu kém chất lượng (quá tải)
* Mô đun hóa không đúng cách
  + Thủ tục phải ở trong một và chỉ một mô-đun
* Xây dựng mô-đun nội bộ kém
  + Nhiều thủ tục truy cập trực tiếp vào cấu trúc dữ liệu
* Phân tích chức năng kém

# Thẻ và Thủy tinh (Mức độ phức tạp cao hơn)

* **Card và Glass sử dụng cùng một khái niệm về fan-in và fan-out để mô tả sự phức tạp của thiết kế:**
  + **Kết cấu phức tạp của mô-đun x**
    - **Sx = (fan-out) 2**
  + **Độ phức tạp của dữ liệu**
    - **Dx = Px / (fan-out +1), trong đó Px là số biến được truyền đến và đi từ mô-đun**
  + **Hệ thống phức tạp**
    - **Cx = Sx + Dx**
* ***Lưu ý: Ngoại trừ Px, người hâm mộ không phải là***
* ***xem xét ở đây***

# Thuộc tính thiết kế “Tốt”

* **Dễ dàng:**
  + **Hiểu không**
  + **Thay đổi**
  + **Tái sử dụng**
  + **Kiểm tra**
  + **Tích hợp**
  + **Mã số**
* **Hãy tin rằng chúng tôi có thể nhận được nhiều "dễ dàng" nếu chúng tôi xem xét:**
  + ***Sự gắn kết***
  + ***Khớp nối***

# Modularity

* Một khái niệm gắn liền với trừu tượng
* Modularity hỗ trợ độc lập các mô hình
* Mô-đun hỗ trợ trừu tượng trong phần mềm
* Hỗ trợ cấu trúc phân cấp của chương trình
* Modularity tăng cường độ rõ ràng của thiết kế, giảm bớt việc triển khai
* Giảm chi phí kiểm tra, gỡ lỗi và bảo trì
* Không thể chỉ cần cắt một chương trình thành các mô-đun để lấy mô đun
* Cần một số tiêu chí để phân hủy

# Khớp nối

* Các mô-đun độc lập: nếu một người có thể hoạt động hoàn toàn mà không có sự hiện diện của người khác
* Độc lập giữa các mô-đun là mong muốn
  + Mô-đun có thể được sửa đổi riêng
  + Có thể được thực hiện và kiểm tra riêng
  + Chi phí lập trình giảm
* Trong một hệ thống tất cả các mô-đun không thể độc lập
* Mô-đun phải hợp tác với nhau
* Kết nối khác giữa các mô-đun
  + Họ phụ thuộc nhiều hơn
  + Cần có thêm kiến ​​thức về một mô-đun để hiểu mô-đun kia.
* Khớp nối nắm bắt được khái niệm phụ thuộc

# Khớp nối…

* Khớp nối giữa các mô-đun là sức mạnh của các kết nối giữa các mô-đun
* Nói chung, chúng ta càng biết nhiều về mô-đun A để hiểu mô đun B thì kết nối chặt chẽ hơn là A đến B
* Các mô-đun "kết hợp cao" được nối với nhau bằng kết nối mạnh mẽ
* Mô-đun "được ghép nối lỏng lẻo" có liên kết yếu
* *Uncoupled -*
* *không phụ thuộc*
* *Rất kết hợp -*
* *nhiều phụ thuộc*
* *Kết hợp lỏng lẻo -*
* *một số phụ thuộc*
* *Khớp nối…*

# Khớp nối…

* Mục tiêu: các mô-đun được kết hợp lỏng lẻo nhất có thể
* Nếu có thể, có các mô-đun độc lập
* Khớp nối được quyết định trong quá trình thiết kế kiến ​​trúc
* Không thể giảm trong khi triển khai
* Khớp nối là khái niệm liên mô-đun
* Các yếu tố chính ảnh hưởng đến khớp nối
  + Loại kết nối giữa các mô-đun
  + Độ phức tạp của giao diện
  + Loại luồng thông tin giữa các mô-đun

# Khớp nối…

* Sự phức tạp và tối tăm của giao diện tăng sự ghép nối
* Giảm thiểu số lượng giao diện cho mỗi mô-đun
* Giảm thiểu sự phức tạp của từng giao diện
* Khớp nối được giảm thiểu nếu
  + Chỉ mục nhập của một mô-đun được sử dụng bởi những người khác
  + Thông tin được truyền độc quyền thông qua các tham số
* Khớp nối tăng nếu
  + Giao diện gián tiếp và tối nghĩa được sử dụng
  + Nội bộ của một mô-đun được sử dụng trực tiếp
  + Biến được chia sẻ được sử dụng cho giao tiếp

# Khớp nối…

* Khớp nối tăng với độ phức tạp của các giao diện, ví dụ như số và độ phức tạp của các tham số
* Giao diện là cần thiết để hỗ trợ giao tiếp cần thiết
* Thường xuyên hơn cần thiết được sử dụng, ví dụ như đi qua toàn bộ hồ sơ khi chỉ cần một trường
* Giữ giao diện của một mô-đun càng đơn giản càng tốt

# Khớp nối…

* Khớp nối phụ thuộc vào loại luồng thông tin
* Hai loại thông tin: dữ liệu hoặc kiểm soát.
* Chuyển thông tin kiểm soát
  + Hành động của mô-đun phụ thuộc vào thông tin
  + Làm cho các mô-đun khó hiểu hơn
* Chuyển thông tin dữ liệu
  + Mô-đun có thể được coi là chức năng đầu vào-đầu ra

# Khớp nối…

* Khớp nối thấp nhất: giao diện chỉ với giao tiếp dữ liệu
* Cao nhất: giao diện lai (một số dữ liệu, một số điều khiển)
* Loại giao diện khớp nối Loại
* sự kết nối phức tạp
* Low Simple To module Dữ liệu
* Hiển thị theo tên
* Cao phức tạp để kiểm soát nội bộ
* Yếu tố tối nghĩa

# Khớp nối

* **Khớp nối địa chỉ các thuộc tính của " mức độ *phụ thuộc lẫn nhau* " giữa các đơn vị phần mềm, mô-đun hoặc các thành phần.**
* **Khớp nối nội dung**
* **Khớp nối chung**
* **Khớp nối điều khiển**
* **Khớp nối tem**
* **Khớp nối dữ liệu**
* **Chỉ truyền thông tin cần thiết**
* **Không có khớp nối**
* **Lý tưởng, nhưng không thực tế**
* **Truy cập dữ liệu nội bộ hoặc thông tin thủ tục**
* **Cấp độ, mức độ**
* **khớp nối ở đâu**
* ***Ghép dữ liệu*** là
* **thấp nhất**
* **Hạ thấp hơn**
* *CAO COUPLING*
* *THẤP*
* *LOOSE*
* *Ghép dữ liệu*
* *Ghép tem*
* *Đã bỏ*
* *Khớp nối điều khiển*
* *Khớp nối chung*
* *Ghép nội dung*
* *Một thành phần sửa đổi dữ liệu nội bộ của một thành phần khác: Không tốt,*
* *Thay đổi trong một thành phần sẽ không ảnh hưởng đến bất kỳ thành phần nào khác*
* *Sắp xếp dữ liệu trong kho dữ liệu chung: Tốt hơn, nhưng phụ thuộc vẫn tồn tại*
* *Một thành phần chuyển các tham số để kiểm soát thành phần khác*
* *Chuyển cấu trúc dữ liệu từ thành phần này sang thành phần khác*
* *Chỉ dữ liệu được truyền, không phải cấu trúc dữ liệu*
* *Khớp nối…*
* *A*
* *B*
* *C*
* *D*
* *E*
* *Hợp phần B*
* *Đến D1*
* *Hợp phần D*
* *Đến D1*
* *D1:*
* *Các nhánh B thành D mặc dù D nằm dưới sự kiểm soát của C*
* *Khớp nối nội dung*
* *Toàn cầu: A1*
* *A2*
* *A3*
* *Biến: V1*
* *V2*
* *Khu vực dữ liệu chung*
* *và tên biến*
* *Thành phần X*
* *Thành phần Y*
* *Hợp phần Z*
* *V1 = V2 + A1*
* *Thay đổi VI thành 0*
* *Tăng VI*
* *Cả ba thành phần đều thay đổi thành V1.* *Có thể là một vấn đề.*
* *Khớp nối chung*

# Sự gắn kết

* Khớp nối đặc trưng liên kết mô-đun liên
* Giảm bằng cách giảm thiểu mối quan hệ giữa các phần tử của các mô-đun khác nhau
* Một phương pháp khác để đạt được điều này là bằng cách tối đa hóa mối quan hệ giữa các phần tử của cùng một mô-đun
* Sự gắn kết xem xét mối quan hệ này
* Quan tâm đến việc xác định các yếu tố của một mô-đun có liên quan chặt chẽ với nhau như thế nào
* Trong thực tế cả hai đều được sử dụng

# Sự gắn kết…

* Sự gắn kết của một mô-đun thể hiện mức độ ràng buộc chặt chẽ của các thành phần của mô-đun
* Cung cấp cho một xử lý về việc liệu các yếu tố khác nhau của một mô-đun thuộc về nhau
* Sự gắn kết cao là mục tiêu
* Sự gắn kết và ghép nối có liên quan với nhau
* Sự gắn kết lớn hơn của các mô-đun, khớp nối thấp hơn giữa mô-đun
* Tương quan không hoàn hảo

# Mức độ gắn kết

* Có nhiều mức độ gắn kết.
  + Trùng hợp ngẫu nhiên
  + Hợp lý
  + Temporal
  + Giao tiếp
  + Tuần tự
  + Chức năng
* Trùng hợp là thấp nhất, chức năng cao nhất
* Thang tỷ lệ không tuyến tính
* Chức năng được coi là rất mạnh

# Sự gắn kết

* **Sự gắn kết của một đơn vị, của một mô-đun, của một đối tượng hoặc một thành phần địa chỉ thuộc tính của "mức độ *liên quan* " trong đơn vị đó, mô-đun, đối tượng hoặc thành phần.**
* **Chức năng**
* **Tuần tự**
* **Giao tiếp**
* **Thủ tục**
* **Temporal**
* **Hợp lý**
* **Trùng hợp ngẫu nhiên**
* **Mức độ gắn kết**
* **nơi *chức năng* là**
* **"Cao nhất"**
* **Biểu diễn nhiều hơn 1**
* **hàm không liên quan**
* **Thực hiện 1 chức năng duy nhất**
* **Càng cao càng tốt**
* *COHESION CAO*
* *THẤP*
* *Hợp lý*
* *Temporal*
* *Trùng hợp ngẫu nhiên*
* *Thủ tục*
* *Giao tiếp*
* *Tuần tự*
* *Chức năng*
* *Các thành phần có bộ phận không liên quan đến nhau*
* *Các hàm hoặc thành phần dữ liệu có liên quan về mặt logic được đặt trong cùng một thành phần*
* *Chức năng chỉ liên quan đến vấn đề thời gian: khó thực hiện thay đổi*
* *Chức năng nhóm trong một thành phần duy nhất chỉ để đảm bảo thứ tự thực hiện thích hợp*
* *Các hàm nhóm trong một thành phần duy nhất chỉ vì chúng tạo ra cùng một tập dữ liệu*
* *Đầu ra từ một phần tử đầu vào thành phần đến một phần tử khác của thành phần*
* *Sự gắn kết lý tưởng: mọi yếu tố xử lý đều cần thiết cho hiệu suất của một hàm duy nhất*
* *Mức độ gắn kết ...*
* *CHỨC NĂNG*
* *D*
* *CHỨC NĂNG*
* *E*
* ***Thủ tục***
* *Liên quan theo thứ tự của*
* *chức năng*
* ***Hợp lý***
* *Các hàm tương tự*
* *CHỨC NĂNG A*
* *CHỨC NĂNG A '*
* *CHỨC NĂNG A ”*
* *logic*
* ***Temporal***
* *Liên quan theo thời gian*
* *TIME T0*
* *TIME T0 + X*
* *TIME T0 + 2X*
* *CHỨC NĂNG A*
* *CHỨC NĂNG B*
* *CHỨC NĂNG C*
* *CHỨC NĂNG A*
* *CHỨC NĂNG B*
* *CHỨC NĂNG C*
* ***Giao tiếp***
* *Truy cập cùng một dữ liệu*
* ***Tuần tự***
* *Đầu ra của một phần là đầu vào tiếp theo*
* *DỮ LIỆU*
* *CHỨC NĂNG A*
* *CHỨC NĂNG B*
* *CHỨC NĂNG C*
* ***Chức năng***
* *Tuần tự với*
* *đầy đủ, chức năng liên quan*
* *CHỨC NĂNG A - phần 1*
* *CHỨC NĂNG A - phần 2*
* *CHỨC NĂNG A - phần 3*
* ***Trùng hợp ngẫu nhiên***
* *Các bộ phận không liên quan*
* *CHỨC NĂNG A*
* *CHỨC NĂNG*
* *B*
* *CHỨC NĂNG*
* *C*
* *Mức độ gắn kết ...*

# Xác định sự gắn kết

* Mô tả mục đích của một mô-đun trong một câu
* Thực hiện các bài kiểm tra sau
  + 1. Nếu câu phải là một câu ghép, chứa nhiều hơn một động từ, thì mô-đun này có thể thực hiện nhiều hơn một hàm. Có thể có sự liên kết tuần tự hoặc giao tiếp.
  + 2. Nếu câu có chứa các từ liên quan đến thời gian, như "đầu tiên", "tiếp theo", "sau", "bắt đầu", vv, mô-đun có thể có liên kết tuần tự hoặc thời gian.

# Đang xác định sự gắn kết ...

* 3. Nếu biến vị ngữ của câu không chứa một đối tượng cụ thể duy nhất theo động từ, thì mô-đun này có lẽ là gắn kết logic. Ví dụ: "chỉnh sửa tất cả dữ liệu", trong khi "chỉnh sửa dữ liệu nguồn" có thể có sự gắn kết chức năng.
* 4. Các từ như "khởi tạo", "dọn dẹp" thường ngụ ý sự gắn kết thời gian.
* Mô-đun gắn kết chức năng luôn có thể được mô tả bằng một câu lệnh đơn giản
* *E*
* *A*
* *B*
* *C*
* *F*
* *D*
* *G*
* *Hệ thống 1*
* *E*
* *A*
* *B*
* *C*
* *F*
* *D*
* *Hệ thống 2*
* *G*
* *Phạm vi kiểm soát: có quyền kiểm soát các yếu tố khác*
* *Phạm vi ảnh hưởng: được kiểm soát bởi các yếu tố khác*
* *Không có thành phần nào trong phạm vi hiệu lực nếu nó không nằm trong phạm vi kiểm soát.* *Nếu phạm vi ảnh hưởng rộng hơn phạm vi kiểm soát, hầu như không thể đảm bảo rằng thay đổi sẽ không phá hủy hệ thống.*
* *Hệ thống 1 có thể tốt hơn vì điều này.*
* *Phạm vi kiểm soát*

# Sử dụng chương trình và dữ liệu lát để đo lường chương trình gắn kết

* **Bieman và Ott đã giới thiệu một thước đo sự gắn kết chương trình bằng cách sử dụng các khái niệm sau đây từ chương trình và các lát dữ liệu:**
  + **Mã thông báo dữ liệu là bất kỳ biến hoặc hằng số nào trong chương trình**
  + **Một lát trong một chương trình là tập hợp tất cả các câu lệnh có thể ảnh hưởng đến giá trị của một số biến quan tâm cụ thể.**
  + **Một slice dữ liệu là tập hợp tất cả các thẻ dữ liệu trong slice sẽ ảnh hưởng đến giá trị của một biến quan tâm cụ thể.**
  + **Mã thông báo** là các thẻ dữ liệu trong chương trình nằm trong nhiều hơn một lát dữ liệu.
  + **Các thẻ siêu dính** là các thẻ dữ liệu trong chương trình nằm trong mọi phần dữ liệu của chương trình
* **Đo lường Chương trình gắn kết thông qua 2 chỉ số:**
* **- sự gắn kết chức năng yếu = (# của mã thông báo) / (tổng số mã thông báo dữ liệu)**
* **- Gắn kết chức năng mạnh mẽ = (#of mã thông báo siêu dính) / (tổng số 3 mã thông báo dữ liệu)**

# Ví dụ về mã giả của hàm đo liên kết chức năng

* **Tìm quy trình giá trị tối đa và tối thiểu:**
* **MinMax (z, n)**
* **Số nguyên cuối, phút, tối đa, i;**
* **end = n;**
* **max = z [0];**
* **min = z [0];**
* **Đối với (i = 0, i = <end, i ++) {**
* **nếu z [i]> max thì max = z [i];**
* **nếu z [i] <min thì min = z [i];**
* **}**
* **trả về max, min;**
* **Mã dữ liệu** :
* **z1**
* **n1**
* **kết thúc1**
* **min1**
* **max1**
* **i1**
* **kết thúc2**
* **n2**
* **max2**
* **z2**
* **01**
* **min2**
* **z3**
* **02**
* **i2**
* **03**
* **i3**
* **end3**
* **i4**
* **z4**
* **i5**
* **max3**
* **max4**
* **z5**
* **i6**
* **z6**
* **i7**
* **min3**
* **min4**
* **z7**
* **i8**
* **max5**
* **min5 (33)**
* **Slice max** :
* **z1**
* **n1**
* **kết thúc1**
* **max1**
* **i1**
* **kết thúc2**
* **n2**
* **max2**
* **z2**
* **01**
* **i2**
* **03**
* **i3**
* **end3**
* **i4**
* **z4**
* **i5**
* **max3**
* **max4**
* **z5**
* **i6**
* **max5**
* **(22)**
* **Slice min** :
* **z1**
* **n1**
* **kết thúc1**
* **min1**
* **i1**
* **kết thúc2**
* **n2**
* **min2**
* **z3**
* **02**
* **i2**
* **03**
* **i3**
* **end3**
* **i4**
* **z6**
* **i7**
* **min3**
* **min4**
* **z7**
* **i8**
* **min5**
* **(22)**
* **Keo thẻ** :
* **z1**
* **n1**
* **kết thúc1**
* **i1**
* **kết thúc2**
* **n2**
* **i2**
* **03**
* **i3**
* **end3**
* **i4 (11)**
* **Keo siêu dính** :
* **z1**
* **n1**
* **kết thúc1**
* **i1**
* **kết thúc2**
* **n2**
* **i2**
* **03**
* **i3**
* **end3**
* **i4 (11)**

# Ví dụ về chỉ số gắn kết mã giả

* **Đối với ví dụ về việc tìm min và max, các mã thông báo keo giống như các thẻ siêu keo.**
  + **Mã keo siêu dính = 11**
  + **Mã thông báo keo = 11**
* **Các slice dữ liệu cho min và slice dữ liệu cho max biến thành cùng một số, 22**
* **Tổng số mã thông báo là 33**
* **Số liệu liên kết cho ví dụ về min-max là:**
* **liên kết chức năng yếu = 11/33 = 1/3**
* **liên kết chức năng mạnh mẽ = 11/33 = 1/3**
* **Nếu chúng tôi chỉ tính một hàm (ví dụ: max), thì:**
* **liên kết chức năng yếu = 22/22 = 1**
* **liên kết chức năng mạnh mẽ = 22/22 = 1**

# Chỉ số OO Chidamber và Kemerer (CK) OO

* **Phương pháp trọng số cho mỗi lớp (WMC)**
* **Độ sâu của cây thừa kế (DIT)**
* **Số trẻ em (NOC)**
* **Khớp nối giữa các lớp đối tượng (CBO)**
* **Trả lời cho một lớp (RFC)**
* **Thiếu sự gắn kết trong các phương pháp (LCOM)**
* ***Lưu ý rằng LCOM là số đo âm trong LCOM cao cho biết***
* ***sự gắn kết thấp và độ phức tạp cao.*** ***Sự gắn kết dựa trên***
* ***kết hợp các phương thức với các biến mẫu chung.***

# Sự gắn kết và khớp nối

* **Sự gắn kết**
* **Khớp nối**
* ***Cấp độ cao***
* ***Cấp thấp***
* **Mạnh**
* **Yếu**
* **Lỏng lẻo**
* **Chặt chẽ**

# Giao diện người dùng

* **3 "quy tắc vàng" của Mandel cho thiết kế giao diện người dùng**
  + ***Đặt người dùng trong tầm kiểm soát***
  + ***Giảm tải bộ nhớ của người dùng (G. Miller 7 + hoặc - 2)***
  + ***Tính nhất quán (trước đó - thiết kế đầy đủ và nhất quán)***
* **Shneiderman và Plaisant (8 quy tắc cho thiết kế)**
  + ***Tính nhất quán***
  + ***Cắt ngắn cho người dùng thường xuyên (hoặc có kinh nghiệm)***
  + ***Phản hồi thông tin***
  + ***Đối thoại sẽ dẫn đến việc đóng***
  + ***Phấn đấu để phòng ngừa lỗi và xử lý lỗi đơn giản***
  + ***Dễ dàng đảo ngược hành động ("hoàn tác" hành động)***
  + ***Kiểm soát nội bộ***
  + ***Giảm bộ nhớ ngắn hạn***

# Nguyên mẫu thiết kế giao diện người dùng và “Thử nghiệm”

* **Nguyên mẫu thiết kế giao diện người dùng:**
  + **Độ trung thực thấp (với bo mạch chủ)**
  + **Độ trung thực cao (với các công cụ "bảng câu chuyện")**
* **Khả năng sử dụng "phòng thí nghiệm" và phân tích thống kê**
  + **Số đối tượng có thể hoàn thành các tác vụ trong một thời gian nhất định**
  + **Thời gian cần thiết để hoàn thành các tác vụ khác nhau**
  + **Số lần cần các chức năng “trợ giúp”**
  + **Số lần "làm lại" được sử dụng và ở đâu**
  + **Số lần "cắt ngắn" được sử dụng**

# Nguồn gốc của Demeter

* **Một "hướng dẫn" thiết kế cho các hệ thống OO có nguồn gốc từ dự án Hệ thống Demeter tại:**
  + **Đại học Northeastern vào những năm 1980**
  + **Dự án lập trình hướng-khía cạnh**
* **Giải quyết vấn đề ghép nối thiết kế thông qua việc đặt các ràng buộc về nhắn tin giữa các đối tượng**
  + **Giới hạn việc gửi tin nhắn đến các đối tượng được trực tiếp biết đến**

# Luật Demeter

* **Một đối tượng cần gửi tin nhắn để chỉ các loại sau đây của các đối tượng:**
  + ***bản thân đối tượng***
  + ***các thuộc tính của đối tượng (các biến mẫu)***
  + ***các tham số của các phương thức trong đối tượng***
  + ***bất kỳ đối tượng nào được tạo bởi một phương thức trong đối tượng***
  + ***bất kỳ đối tượng nào được trả về từ một cuộc gọi đến một trong các phương thức của đối tượng***
  + ***bất kỳ đối tượng nào trong bất kỳ bộ sưu tập nào thuộc một trong các danh mục trên***