**BỘ TÀI LIỆU CÔNG NGHỆ PHẦN MỀM**

Mục Lục

[Mục Tiêu Môn Học: 11](#_Toc518424320)

[I. Giới thiệu về Kỹ thuật phần mềm: 12](#_Toc518424321)

[II. Phát triển phần mềm chuyên nghiệp 12](#_Toc518424322)

[III. Chuyên sâu về kĩ thuật phần mềm: 15](#_Toc518424323)

[IV. Đa dạng về công nghệ phần mềm: 16](#_Toc518424324)

[V. Đạo đức kỹ thuật phần mềm: 18](#_Toc518424325)

[Chapter 3: Software Process 21](#_Toc518424326)

[3.1 Đặc điểm kỹ thuật phần mềm 21](#_Toc518424327)

[3.2. Thiết kế & triển khai phần mềm 21](#_Toc518424328)

[3.3. Xác nhận phần mềm 21](#_Toc518424329)

[3.4. Tiến hóa phần mềm 21](#_Toc518424330)

[3.5 Software Process Model 22](#_Toc518424331)

[3.5.1 Mô hình tuyến tính 22](#_Toc518424332)

[3.5.2 Mô hình thác nước(Waterfall model) 22](#_Toc518424333)

[3.5.2 aCác giai đoạn chính của mô hình thác nước 23](#_Toc518424334)

[3.5.3 Mô hình lặp đi lặp lại(Iterative Models) 24](#_Toc518424335)

[3.5.4 The Spiral Model 24](#_Toc518424336)

[3.6 giai đoạn của mô hình xoắn ốc 25](#_Toc518424337)

[3.7 Mô hình phát triển dựa trên thành phần 26](#_Toc518424338)

[3.8 Đặc điểm phần mềm 27](#_Toc518424339)

[3.9 Software Design & Implementation 28](#_Toc518424340)

[3.10 Activities may be part of the design process 29](#_Toc518424341)

[3.11 Xác nhận phần mềm 30](#_Toc518424342)

[3.12 Quy trình thử nghiệm ba giai đoạn 30](#_Toc518424343)

[3.13 Các giai đoạn trong quá trình thử nghiệm 31](#_Toc518424344)

[3.14 Tiến hóa phần mềm 32](#_Toc518424345)

[3.15 Xác định vai trò và trách nhiệm 32](#_Toc518424346)

[3.15.1 Vai trò & trách nhiệm tại các giai đoạn SDLC 33](#_Toc518424347)

[Chapter 4 :Software Project Management 34](#_Toc518424348)

[4.1 Phân biệt quản lý phần mềm 35](#_Toc518424349)

[4.2 Hoạt động quản lý 35](#_Toc518424350)

[4.3 Quản lý rủi ro 36](#_Toc518424351)

[4.4 Quy trình quản lý rủi ro 36](#_Toc518424352)

[4.5 The Risk Management Process 36](#_Toc518424353)

[4.6 Examples of Different Risk Types 37](#_Toc518424354)

[4.7 Phân tích rủi ro 37](#_Toc518424355)

[4.8 Các loại rủi ro & ví dụ 37](#_Toc518424356)

[4.9 Lập kế hoạch rủi ro 38](#_Toc518424357)

[4.10 Các chiến lược để giúp quản lý rủi ro 38](#_Toc518424358)

[4.11 Giám sát rủi ro 39](#_Toc518424359)

[4.12 Chỉ số rủi ro 39](#_Toc518424360)

[4.12.1 Những điểm chính 39](#_Toc518424361)

[4.13 Quản lý con người 40](#_Toc518424362)

[4.13.1 Nhân tố quản lý con người 40](#_Toc518424363)

[4.14 Thúc đẩy mọi người 40](#_Toc518424364)

[4.15 Phân cấp nhu cầu con người 41](#_Toc518424365)

[4.15.1 Kiểu tính cách 42](#_Toc518424366)

[4.15.2 Các loại tính cách tự định hướng 42](#_Toc518424367)

[4.16 Cân bằng động lực 42](#_Toc518424368)

[4.17 Teamwork 42](#_Toc518424369)

[4.18 Tính gắn kết nhóm 42](#_Toc518424370)

[4.19 Những lợi thế của một nhóm cố kết 43](#_Toc518424371)

[4.20 Hiệu quả của một đội 43](#_Toc518424372)

[4.21 Chọn thành viên nhóm 43](#_Toc518424373)

[4.22 Lắp ráp một đội 43](#_Toc518424374)

[4.23 Thành phần nhóm 44](#_Toc518424375)

[4.24 Tổ chức nhóm 44](#_Toc518424376)

[4.25 Nhóm không chính thức 44](#_Toc518424377)

[4.26 Nhóm truyền thông 45](#_Toc518424378)

[Chapter 5: Requirement Engineering 45](#_Toc518424379)

[5.1 Topics covered 45](#_Toc518424380)

[5.2 Requirements Engineering 46](#_Toc518424381)

[5.3 What Is a Requirement? 46](#_Toc518424382)

[5.4 Yêu cầu trừu tượng (Davis) 46](#_Toc518424383)

[5.5 Các loại yêu cầu 46](#_Toc518424384)

[5.6 Yêu cầu của người dùng và hệ thống 47](#_Toc518424385)

[5.7 Độc giả của các loại yêu cầu khác nhau 47](#_Toc518424386)

[5.8 Yêu cầu chức năng và phi chức năng 47](#_Toc518424387)

[5.9 Yêu cầu chức năng 48](#_Toc518424388)

[5.10 Yêu cầu chức năng cho MHC-PMS 48](#_Toc518424389)

[5.11 Yêu cầu Imprecision(NULL) 48](#_Toc518424390)

[5.12 Yêu cầu đầy đủ và nhất quán 48](#_Toc518424391)

[5.13 Những yêu cầu phi lý 49](#_Toc518424392)

[5.16 Các loại yêu cầu phi chức năng 49](#_Toc518424393)

[5.17 Thực hiện các yêu cầu phi chức năng 49](#_Toc518424394)

[5.18 Phân loại phi chức năng 49](#_Toc518424395)

[5.19 Ví dụ về các yêu cầu phi chức năng trong MHC-PMS 50](#_Toc518424396)

[5.20 Mục tiêu và yêu cầu 50](#_Toc518424397)

[5.21 Yêu cầu về tính khả dụng 50](#_Toc518424398)

[5.22 Số liệu để chỉ định các yêu cầu phi chức năng 50](#_Toc518424399)

[5.23 Yêu cầu về miền 51](#_Toc518424400)

[5.24 Hệ thống bảo vệ tàu hỏa 51](#_Toc518424401)

[5.25 Các vấn đề về yêu cầu miền 51](#_Toc518424402)

[5.26 Key points (chốt bài) 52](#_Toc518424403)

[5.27 Phương thức và yêu cầu nhanh 52](#_Toc518424404)

[5.28 Người dùng tài liệu yêu cầu 53](#_Toc518424405)

[5.29 Yêu cầu thay đổi tài liệu 53](#_Toc518424406)

[5.30 Cấu trúc của tài liệu yêu cầu 53](#_Toc518424407)

[5.31 Cấu trúc của tài liệu yêu cầu 54](#_Toc518424408)

[5.32 Yêu cầu kỹ thuật 54](#_Toc518424409)

[5.33 Cách viết đặc tả yêu cầu hệ thống 55](#_Toc518424410)

[5.34 Yêu cầu và thiết kế 55](#_Toc518424411)

[5.35 Đặc điểm ngôn ngữ tự nhiên 55](#_Toc518424412)

[5.36 Hướng dẫn viết yêu cầu 56](#_Toc518424413)

[5.37 Vấn đề với ngôn ngữ tự nhiên 56](#_Toc518424414)

[5.38 Thông số kỹ thuật có cấu trúc 56](#_Toc518424415)

[5.39 Thông số dựa trên biểu mẫu 56](#_Toc518424416)

[5.40 Yêu cầu quy trình kỹ thuật 58](#_Toc518424417)

[5.41 Một cái nhìn xoắn ốc của quá trình kỹ thuật yêu cầu 58](#_Toc518424418)

[5.42 Yêu cầu gợi ý và phân tích 58](#_Toc518424419)

[5.43 Vấn đề phân tích yêu cầu 58](#_Toc518424420)

[5.44 Quy trình phân tích và yêu cầu yêu cầu 59](#_Toc518424421)

[5.45 Quy trình hoạt động 59](#_Toc518424422)

[5.46 Các vấn đề về yêu cầu 60](#_Toc518424423)

[Chapter 6: Requirement Engineer Process 60](#_Toc518424424)

[6.1 Yêu cầu quy trình kỹ thuật: 60](#_Toc518424425)

[6.1.1 Nghiên cứu khả thi 61](#_Toc518424426)

[6.2 Quan điểm: 64](#_Toc518424427)

[6.3 LIBSYS viewpoint hierarchy 65](#_Toc518424428)

[6.3.1 Họp bàn ý kiến 65](#_Toc518424429)

[6.4 Article printing use-case: 67](#_Toc518424430)

[6.5 LIBSYS use cases 67](#_Toc518424431)

[6.6 Article Printing 68](#_Toc518424432)

[6.7 Print article sequences: 68](#_Toc518424433)

[6.8 Các yếu tố xã hội và tổ chức 68](#_Toc518424434)

[6.9 Ethnography and Prototyping 69](#_Toc518424435)

[6.10 Xác nhận yêu cầu 69](#_Toc518424436)

[6.11 Phân loại yêu cầu 71](#_Toc518424437)

[6.12 Yêu cầu lập kế hoạch quản lý 72](#_Toc518424438)

[6.13 Truy xuất nguồn gốc 72](#_Toc518424439)

[6.14 Ma trận truy xuất nguồn gốc(A traceability matrix) 72](#_Toc518424440)

[6.15 CASE Tool Support 73](#_Toc518424441)

[Chapter 7 : Design & Implement 74](#_Toc518424442)

[7.1 Thiết kế và thực hiện 74](#_Toc518424443)

[7.2 Xây dựng hoặc mua 74](#_Toc518424444)

[7.3 Hướng đối tượng 74](#_Toc518424445)

[7.4 Quy trình giai đoạn 75](#_Toc518424446)

[7.5 Ngữ cảnh hệ thống và tương tác 75](#_Toc518424447)

[7.6 System Context For The Weather Station 76](#_Toc518424448)

[7.7 Weather station use cases 76](#_Toc518424449)

[7.8 Use Case Description—Report Weather 76](#_Toc518424450)

[7.9 Thiết kế kiến trúc 77](#_Toc518424451)

[7.10 Kiến trúc cấp cao của Trạm thời tiết 77](#_Toc518424452)

[7.11 Kiến trúc của hệ thống thu thập dữ liệu 78](#_Toc518424453)

[7.12 Phương pháp tiếp cận để xác định 78](#_Toc518424454)

[7.13 Các lớp đối tượng trạm thời tiết 79](#_Toc518424455)

[7.14 Các lớp đối tượng trạm thời tiết 80](#_Toc518424456)

[7.15 Mô hình thiết kế 80](#_Toc518424457)

[7.16 Mô hình hệ thống con 80](#_Toc518424458)

[7.17 Mô hình trình tự(Sequence Model) 81](#_Toc518424459)

[7.18 Biểu đồ trình tự mô tả thu thập dữ liệu(Sequence Diagram Describing Data Collection) 81](#_Toc518424460)

[7.19 Sơ đồ nhà nước, Sơ đồ trạng thái(State Diagrams)(vcc sơ đồ nhà nước ☺) 82](#_Toc518424461)

[7.20 Đặc điểm giao diện(Interface Specification) 82](#_Toc518424462)

[7.21 Mẫu thiết kế(Design Pattern) 83](#_Toc518424463)

[7.22 Mẫu Observer 84](#_Toc518424464)

[7.23 Nhiều màn hình sử dụng mẫu Observer(Multiple Displays Using the Observer Pattern) 86](#_Toc518424465)

[7.24 Mô hình UML của mẫu Observer 86](#_Toc518424466)

[7.25 Vấn đề thiết kế 86](#_Toc518424467)

[7.26 Vấn đề thực hiện 87](#_Toc518424468)

[7.27 Quản lý cấu hình 88](#_Toc518424469)

[7.28 Host-Target Development 89](#_Toc518424470)

[7.29 Môi trường phát triển tích hợp (IDE) 90](#_Toc518424471)

[7.30 Thành phần / Hệ thống Các yếu tố triển khai 90](#_Toc518424472)

[7.31 Phát triển nguồn mở 91](#_Toc518424473)

[7.32 Mô hình giấy phép(License Model) 92](#_Toc518424474)

[Chương 8: Mô hình hệ thống. 93](#_Toc518424475)

[8.1 Kế hoạch cho mô hình hệ thống. 94](#_Toc518424476)

[8.2 Quan điểm hệ thống 94](#_Toc518424477)

[8.3 Các sơ đồ uml. 94](#_Toc518424478)

[Biểu đồ Use case (Use Case Diagram): 94](#_Toc518424479)

[Biểu đồ lớp (Class Diagram): 94](#_Toc518424480)

[Biểu đồ trạng thái (State Diagram): 95](#_Toc518424481)

[Biểu đồ trình tự (Sequence Diagram): 95](#_Toc518424482)

[8.4 Context Models 95](#_Toc518424483)

[8.5 Ranh giới hệ thống. 96](#_Toc518424484)

[8.6 Phối cảnh quy trình. 96](#_Toc518424485)

[8.7Mô hình tương tác. 96](#_Toc518424486)

[Use Case Modeling. 96](#_Toc518424487)

[8.8 Các điểm chính. 97](#_Toc518424488)

[8.9 Tổng quát. 97](#_Toc518424489)

[8.10 Mô hình hành vi. 97](#_Toc518424490)

[8.11 Mô hình hướng sự kiện 98](#_Toc518424491)

[8.12 Kỹ thuật theo mô hình. 98](#_Toc518424492)

[8.13 Sử dụng kỹ thuật theo mô hình 98](#_Toc518424493)

[8.14 Các loại mô hình. 98](#_Toc518424494)

[Chương 9 99](#_Toc518424495)

[9.1 Kỹ thuật phần mềm dựa trên thành phần. 99](#_Toc518424496)

[9.2 CBSE Essentials 99](#_Toc518424497)

[9.3 Các thành phần. 100](#_Toc518424498)

[9.4 Định nghĩa thành phần 100](#_Toc518424499)

[9.5 Thành phần đặc điểm. 101](#_Toc518424500)

[9.6 Thành phần với tư cách là nhà cung cấp dịch vụ. 101](#_Toc518424501)

[9.7 Mô hình thành phần. 102](#_Toc518424502)

[9.7.1 Các thành phần của mô hình thành phần 102](#_Toc518424503)

[9.8 Middleware Hỗ trợ. 102](#_Toc518424504)

[9.8.1 Dịch vụ Middleware được định nghĩa trong một mô hình thành phần 103](#_Toc518424505)

[9.9 Qúa Trinh CBSE. 103](#_Toc518424506)

[9.10 Quy trình hỗ trợ. 104](#_Toc518424507)

[9.11 CBSE để tái sử dụng. 104](#_Toc518424508)

[9.12 Nên phản ánh trừu tượng miền ổn định. 105](#_Toc518424509)

[9.13 Phát triển thành phần để tái sử dụng. 105](#_Toc518424510)

[9.14 Xử lý ngoại lệ. 106](#_Toc518424511)

[9.15 Các thành phần hệ thống cũ. 106](#_Toc518424512)

[9.16 CBSE với tái sử dụng. 106](#_Toc518424513)

[9.17 Quy trình nhận dạng thành phần. 107](#_Toc518424514)

[9.18 Các vấn đề về nhận dạng thành phần. 107](#_Toc518424515)

[9.19 Xác thực thành phần. 108](#_Toc518424516)

[Chapter 10: Thiết kế kiến ​​trúc 109](#_Toc518424517)

[10.1 Kiến trúc phần mềm 109](#_Toc518424518)

[10.2 Ưu điểm của kiến ​​trúc rõ ràng 110](#_Toc518424519)

[10.3 Đại diện kiến ​​trúc 111](#_Toc518424520)

[10.4 Biểu đồ hộp & đường 111](#_Toc518424521)

[10.5 Sử dụng mô hình kiến ​​trúc 111](#_Toc518424522)

[10.6 Quyết định thiết kế kiến ​​trúc 112](#_Toc518424523)

[10.7 Kiến trúc tái sử dụng 112](#_Toc518424524)

[10.8 Kiến trúc & đặc điểm hệ thống 113](#_Toc518424525)

[10.9 Quan điểm kiến ​​trúc 114](#_Toc518424526)

[10.10 4 + 1 mô hình xem kiến ​​trúc phần mềm 114](#_Toc518424527)

[10.11 Mẫu kiến ​​trúc 115](#_Toc518424528)

[10.12 Mẫu Model-View-Controller (MVC) 115](#_Toc518424529)

[10.13 Tổ chức Model-View-Controller 116](#_Toc518424530)

[10.14 Kiến trúc ứng dụng web sử dụng mẫu MVC 116](#_Toc518424531)

[10.15 Kiến trúc lớp 117](#_Toc518424532)

[10.16 Mô hình kiến ​​trúc Layered 117](#_Toc518424533)

[10.17 Một kiến ​​trúc lớp chung 118](#_Toc518424534)

[10.18 Kiến trúc của hệ thống LIBSYS 118](#_Toc518424535)

[10.19 Kiến trúc kho lưu trữ 119](#_Toc518424536)

[10.20 Mẫu Repository 119](#_Toc518424537)

[10.21 Kiến trúc kho lưu trữ cho một IDE 120](#_Toc518424538)

[10.22 Kiến trúc máy khách-máy chủ 120](#_Toc518424539)

[10.23 Mẫu Client-server 121](#_Toc518424540)

[10.24 Kiến trúc máy khách-máy chủ cho thư viện phim 122](#_Toc518424541)

[10.25 Kiến trúc ứng dụng 123](#_Toc518424542)

[10.26 Hệ thống xử lý giao dịch 124](#_Toc518424543)

[10.28 Cấu trúc của các ứng dụng xử lý giao dịch 125](#_Toc518424544)

[10.29 Kiến trúc phần mềm của một hệ thống ATM 125](#_Toc518424545)

[10.30 Kiến trúc hệ thống thông tin 126](#_Toc518424546)

[10.31 Kiến trúc hệ thống thông tin lớp 126](#_Toc518424547)

[10.32 Kiến trúc của MHC-PMS 127](#_Toc518424548)

[10.33 Hệ thống thông tin dựa trên web 127](#_Toc518424549)

[10.34 Triển khai máy chủ 127](#_Toc518424550)

[10.35 Hệ thống xử lý ngôn ngữ 128](#_Toc518424551)

[10.36 Thành phần trình biên dịch 128](#_Toc518424552)

[10.37 Kiến trúc trình biên dịch ống và bộ lọc 129](#_Toc518424553)

[10.38 Kiến trúc kho lưu trữ cho hệ thống xử lý ngôn ngữ 129](#_Toc518424554)

[Chapter 11: Kiểm thử phần mềm 130](#_Toc518424555)

[11.1 Kiểm tra chương trình 130](#_Toc518424556)

[11.2 Mục tiêu kiểm tra chương trình 131](#_Toc518424557)

[11.3 Kiểm tra xác thực và lỗi 131](#_Toc518424558)

[11.4 Mục tiêu quy trình thử nghiệm 131](#_Toc518424559)

[11.5 Một mô hình đầu vào-đầu ra của kiểm thử chương trình 132](#_Toc518424560)

[11.6 Xác minh và xác thực 132](#_Toc518424561)

[11.7 Tự tin V & V 132](#_Toc518424562)

[11.8 Kiểm tra & Kiểm tra 133](#_Toc518424563)

[11.9 Kiểm tra phần mềm 133](#_Toc518424564)

[11.9.1Ưu điểm của kiểm tra 134](#_Toc518424565)

[11.10 Một mô hình của quá trình kiểm thử phần mềm 134](#_Toc518424566)

[11.10.1 Các giai đoạn thử nghiệm 134](#_Toc518424567)

[11.11 Thử nghiệm phát triển 135](#_Toc518424568)

[11.12 Kiểm tra đơn vị 135](#_Toc518424569)

[11.13 Kiểm tra lớp đối tượng 135](#_Toc518424570)

[11.14 Giao diện đối tượng trạm thời tiết 136](#_Toc518424571)

[11.14.1 Kiểm tra trạm thời tiết 136](#_Toc518424572)

[11.15 Thử nghiệm tự động 136](#_Toc518424573)

[11.15.1 Các thành phần kiểm tra tự động 137](#_Toc518424574)

[11.16 Đơn vị kiểm tra hiệu quả 137](#_Toc518424575)

[11.17 Chiến lược thử nghiệm 137](#_Toc518424576)

[11.18 Kiểm tra phân vùng 138](#_Toc518424577)

[11.19 Nguyên tắc kiểm tra (chuỗi) 138](#_Toc518424578)

[11.19.1 Nguyên tắc kiểm tra chung 138](#_Toc518424579)

[11.20 Kiểm tra thành phần 139](#_Toc518424580)

[11.21 Kiểm tra giao diện 140](#_Toc518424581)

[11.22 Lỗi giao diện 140](#_Toc518424582)

[11.23 Nguyên tắc kiểm tra giao diện 141](#_Toc518424583)

[11.24 Thử nghiệm hệ thống 141](#_Toc518424584)

[11.25 Kiểm tra hệ thống và thành phần 141](#_Toc518424585)

[11.27 Thử nghiệm Usecase 142](#_Toc518424586)

[11.28 Thu thập biểu đồ trình tự dữ liệu thời tiết 142](#_Toc518424587)

[11.29 Chính sách thử nghiệm 142](#_Toc518424588)

[11.30 Hướng phát triển thử nghiệm 142](#_Toc518424589)

[11.31 TDD Process Các hoạt động 143](#_Toc518424590)

[11.31.1 Lợi ích của phát triển theo hướng thử nghiệm 143](#_Toc518424591)

[11.32 Kiểm tra hồi quy 144](#_Toc518424592)

[11.33 Kiểm tra phát hành 144](#_Toc518424593)

[11.34 Phát hành thử nghiệm và kiểm tra hệ thống 145](#_Toc518424594)

[11.35 Yêu cầu dựa trên thử nghiệm 145](#_Toc518424595)

[11.35.1 Yêu cầu kiểm tra 145](#_Toc518424596)

[11.36 Tính năng thử nghiệm theo kịch bản 146](#_Toc518424597)

[11.37 Kịch bản sử dụng cho MHC-PMS 146](#_Toc518424598)

[11.38 Kiểm tra năng suất 147](#_Toc518424599)

[11.39 Kiểm tra người dùng 147](#_Toc518424600)

[11.39.1 Các loại thử nghiệm người dùng 147](#_Toc518424601)

[11.40 Quá trình kiểm tra nghiệm thu 148](#_Toc518424602)

[11.40.1 Các giai đoạn trong quá trình kiểm tra nghiệm thu 148](#_Toc518424603)

[11.41 Phương thức nhanh và thử nghiệm chấp nhận 148](#_Toc518424604)

[Chapter 12: Project Planning(lập kế hoạch dự án) 149](#_Toc518424605)

[I. Chủ đề được đề cập: 149](#_Toc518424606)

[II. Lập kế hoạch dự án 150](#_Toc518424607)

[III. Phát triển theo kế hoạch: 151](#_Toc518424608)

[IV. Agile planning 154](#_Toc518424609)

[***Những điểm chính*** 163](#_Toc518424610)

# CHƯƠNG 1: Giới thiệu về CNPM

# Mục Tiêu Môn Học:

* Để giới thiệu kỹ thuật phần mềm và giải thích tầm quan trọng của nó
* Để đặt ra câu trả lời cho các câu hỏi chính về kỹ nghệ phần mề
* Giới thiệu các vấn đề đạo đức và chuyên môn và giải thích tại sao họ quan tâm đến các kỹ sư phần mềm

🡪 Chủ đề được đề cập:

* + - * Câu hỏi thường gặp về kỹ thuật phần mềm
      * Trách nhiệm chuyên môn và đạo đức

🡪Giới Thiệu:

* Chúng ta không thể chạy thế giới hiện đại mà không cần phần mềm
* Cơ sở hạ tầng quốc gia và các tiện ích được kiểm soát bởi các hệ thống dựa trên máy tính và hầu hết các sản phẩm điện bao gồm máy tính và phần mềm điều khiển

Ví dụ: 1/Sản xuất & phân phối công nghiệp hoàn toàn được vi tính hóa

2/Giải trí

**Hệ thống phần mềm:**

* Tóm tắt & vô hình
* KHÔNG bị ràng buộc bởi các tính chất của vật liệu, được điều chỉnh bởi các luật vật lý, hoặc bởi các quy trình sản xuất

*Tuy nhiên: Vì thiếu các ràng buộc vật lý nên:*

* Hệ thống phần mềm có thể
* Nhanh chóng trở nên vô cùng phức tạp
* Khó để hiểu
* Đắt tiền để thay đổi

Có nhiều loại hệ thống phần mềm khác biệt và nhiều trong số này được gọi là Thất bại phần mềm là hậu quả của 2 yếu tố:

1/ Tăng nhu cầu

2/ Kỳ vọng thấp

Tuy nhiên, Kỹ sư phần mềm có thể làm Phát triển phần mềm chuyên nghiệp:

* Rất nhiều người viết chương trình
* Những người trong doanh nghiệp viết chương trình bảng tính
* Các nhà khoa học và kỹ sư viết chương trình để xử lý dữ liệu thử nghiệm của họ
* Người yêu thích viết các chương trình vì lợi ích và sự thích thú của riêng họ

*Tuy nhiên:*

* *Phần lớn phát triển phần mềm là một hoạt động chuyên nghiệp, nơi phần mềm được phát triển cho các mục đích kinh doanh cụ thể (ví dụ: Hệ thống thông tin, hệ thống CAD)*
* *Phần mềm chuyên nghiệp, nhằm mục đích sử dụng bởi một người ngoài nhà phát triển, thường được phát triển bởi các nhóm chứ không phải cá nhân. Nó được duy trì và thay đổi trong suốt cuộc đời của nó.*

# Giới thiệu về Kỹ thuật phần mềm:

* Kỹ thuật phần mềm nhằm hỗ trợ phát triển phần mềm chuyên nghiệp, thay vì lập trình riêng lẻ.
* Nó bao gồm các kỹ thuật hỗ trợ đặc tả, thiết kế và tiến hóa của chương trình, không có điều nào trong số đó thường có liên quan đến phát triển phần mềm cá nhân

# Phát triển phần mềm chuyên nghiệp

* Nhiều người nghĩ rằng phần mềm chỉ là một từ khác cho các chương trình máy tính.
* Tuy nhiên, khi chúng ta nói về kĩ nghệ phần mềm, phần mềm không chỉ là bản thân các chương trình mà còn tất cả các tài liệu và tài liệu cấu hình liên quan được yêu cầu để làm cho các chương trình này hoạt động chính xác.
* Một hệ thống phần mềm chuyên nghiệp được phát triển thường là nhiều hơn một chương trình duy nhất

* Nó có thể bao gồm tài liệu hệ thống, mô tả cấu trúc của hệ thống; tài liệu người dùng, giải thích cách sử dụng hệ thống và trang web để người dùng tải xuống thông tin sản phẩm gần đây

*Các loại sản phẩm phần mềm:*

1. Sản phẩm chung
2. Xử lý văn bản
3. Bảng tính
4. Cơ sở dữ liệu
5. Gói vẽ
6. …
7. Sản phẩm tùy chỉnh (hoặc đặt riêng)
8. Kiểm soát không lưu
9. ERP
10. Mua sắm trực tuyến

*Chất lượng phần mềm:*

* Nó phải bao gồm hành vi của phần mềm trong khi nó đang thực hiện và cấu trúc và tổ chức của các chương trình hệ thống và tài liệu liên quan.
* Điều này được phản ánh trong cái gọi là thuộc tính phần mềm chất lượng hoặc phi chức năng. Ví dụ về các thuộc tính này là thời gian phản hồi của phần mềm đối với truy vấn của người dùng và tính dễ hiểu của mã chương trình
* Tập hợp các thuộc tính cụ thể mà bạn có thể mong đợi từ một hệ thống phần mềm rõ ràng phụ thuộc vào ứng dụng của nó:
* Hệ thống ngân hàng
* Trò chơi tương tác
* Hệ thống chuyển mạch điện thoại

🡪 *Chúng có thể được tổng quát thành tập hợp các thuộc tính*

|  |  |
| --- | --- |
| Đặc tính sản phẩm | Mô tả |
| Duy trì | Phần mềm phải được viết theo cách sao cho nó có thể phát triển thành  đáp ứng nhu cầu thay đổi của khách hàng. Đây là thuộc tính quan trọng  bởi vì thay đổi phần mềm là một yêu cầu không thể tránh khỏi của  thay đổi môi trường kinh doanh. |
| Tính phụ thuộc và bảo mật | Phần mềm đáng tin cậy bao gồm một loạt các đặc tính bao gồm độ tin cậy, an ninh và an toàn. Phần mềm đáng tin cậy không nên gây thiệt hại về thể chất hoặc kinh tế trong trường hợp lỗi hệ thống. Người dùng độc hại không thể truy cập hoặc làm hỏng hệ thống. |
| Hiệu quả | Phần mềm không nên sử dụng lãng phí tài nguyên hệ thống như vậy  như bộ nhớ và chu kỳ xử lý. Do đó hiệu quả bao gồm  đáp ứng, thời gian xử lý, sử dụng bộ nhớ, v.v. |
| Khả năng chấp nhận | Phần mềm phải được chấp nhận đối với loại người dùng mà nó là  được thiết kế. Điều này có nghĩa là nó phải dễ hiểu, có thể sử dụng và  tương thích với các hệ thống khác mà họ sử dụng. |

# Chuyên sâu về kĩ thuật phần mềm:

🡪*Là một kỷ luật kỹ thuật có liên quan đến tất cả các khía cạnh của sản xuất phần mềm từ các giai đoạn đầu của đặc tả hệ thống thông qua để duy trì hệ thống sau khi nó đã đi vào sử dụng*

Có 2 cụm từ khóa:

1. *Engineering Discipline(*Kỷ luật kỹ thuật):
   * Kỹ sư làm cho mọi thứ hoạt động. Họ áp dụng các lý thuyết, phương pháp và công cụ trong đó chúng phù hợp
   * Tuy nhiên, họ sử dụng chúng một cách có chọn lọc và luôn cố gắng khám phá các giải pháp cho các vấn đề ngay cả khi không có lý thuyết và phương pháp có thể áp dụng.
   * Các kỹ sư cũng nhận ra rằng họ phải làm việc với các ràng buộc về tài chính và tổ chức để họ tìm kiếm các giải pháp trong các ràng buộc này.
2. *Tất cả các khía cạnh của sản xuất phần mềm:*

* Kỹ nghệ phần mềm không chỉ quan tâm tới các qui trình kĩ thuật của phát triển phần mềm.
* Nó cũng bao gồm các hoạt động như quản lý dự án phần mềm và phát triển các công cụ, phương pháp và lý thuyết để hỗ trợ sản xuất phần mềm.

***Kỹ thuật phần mềm rất quan trọng vì:***

1/Ngày càng nhiều, cá nhân và xã hội dựa vào các hệ thống phần mềm tiên tiến. Chúng ta cần có khả năng tạo ra các hệ thống đáng tin cậy và đáng tin cậy một cách kinh tế và nhanh chóng.

2.1/Nó thường rẻ hơn, về lâu dài, để sử dụng các phương pháp kỹ thuật phần mềm và các kỹ thuật cho các hệ thống phần mềm thay vì chỉ viết các chương trình như thể nó là một dự án lập trình cá nhân.

2.2/Đối với hầu hết các loại hệ thống, phần lớn chi phí là chi phí thay đổi phần mềm sau khi đã sử dụng.

*🡪Cách tiếp cận có hệ thống được sử dụng trong kĩ nghệ phần mềm đôi khi được gọi là qui trình phần mềm:*

* **Một quy trình phần mềm**: Là một chuỗi các hoạt động dẫn đến việc sản xuất một sản phẩm phần mềm.
* Có bốn hoạt động cơ bản phổ biến cho tất cả các quy trình phần mềm:

1/ Đặc điểm phần mềm là nơi khách hàng và kỹ sư xác định phần mềm sẽ được sản xuất và các ràng buộc trong hoạt động của nó.

2/ Phát triển phần mềm là phần mềm được thiết kế và lập trình ở đâu

3/ Xác thực phần mềm: Trường hợp phần mềm được kiểm tra để đảm bảo rằng đó là những gì khách hàng yêu cầu.

4/ Tiến hóa phần mềm: Khi phần mềm được sửa đổi để phản ánh các yêu cầu thay đổi của khách hàng và thị trường.

*Đặt câu hỏi: Sự khác biệt giữa kĩ nghệ phần mềm và khoa học máy tính là gì?*

* Khoa học máy tính liên quan đến lý thuyết và nguyên tắc cơ bản; kĩ nghệ phần mềm có liên quan với thực tiễn phát triển và cung cấp phần mềm hữu ích.
* Các lý thuyết khoa học máy tính vẫn chưa đủ để hoạt động như một nền tảng hoàn chỉnh cho kỹ nghệ phần mềm (không giống như vật lý và kỹ thuật điện).
* Hệ thống kỹ thuật có liên quan với tất cả các khía cạnh của phát triển hệ thống dựa trên máy tính bao gồm phần cứng, phần mềm và kỹ thuật quy trình. Kỹ nghệ phần mềm là một phần của quá trình này liên quan tới việc phát triển cơ sở hạ tầng phần mềm, kiểm soát, ứng dụng và cơ sở dữ liệu trong hệ thống.
* Các kỹ sư hệ thống có liên quan đến đặc tả hệ thống, thiết kế kiến ​​trúc, tích hợp và triển khai.

*Lưu ý:* *Không có phương pháp hoặc kỹ thuật phần mềm phổ quát nào được áp dụng cho tất cả các loại phần mềm.* *Tuy nhiên, có 03 vấn đề chung ảnh hưởng đến nhiều loại phần mềm khác nhau:*

*1/ Tính không đồng nhất:*🡪Các hệ thống được yêu cầu hoạt động như hệ thống phân tán trên các mạng bao gồm các loại máy tính và thiết bị di động khác nhau

*2/ Thay đổi kinh doanh & xã hội:*

*🡪* Kinh doanh và xã hội đang thay đổi cực kỳ nhanh chóng khi các nền kinh tế mới nổi phát triển và công nghệ mới trở nên có sẵn.

🡪Họ cần có khả năng thay đổi phần mềm hiện có của họ và phát triển nhanh phần mềm mới

*3/ Bảo mật và tin tưởng:*

🡪 Khi phần mềm được đan xen với tất cả các khía cạnh của cuộc sống của chúng ta, điều quan trọng là chúng ta có thể tin tưởng phần mềm đó.

🡪Điều này đặc biệt đúng với các hệ thống phần mềm từ xa được truy cập thông qua một trang web hoặc giao diện dịch vụ web.

🡪Chúng tôi phải đảm bảo rằng người dùng độc hại không thể tấn công phần mềm của chúng tôi và bảo mật thông tin đó được duy trì.

# Đa dạng về công nghệ phần mềm:

* Yếu tố quan trọng nhất trong việc xác định phương pháp và kỹ thuật phần mềm nào là quan trọng nhất là loại ứng dụng đang được phát triển
* Có nhiều loại ứng dụng:
* Ứng dụng độc lập
* Ứng dụng dựa trên giao dịch tương tác
* Hệ thống điều khiển nhúng

1/ *Ứng dụng độc lập:*

* Đây là những hệ thống ứng dụng chạy trên máy tính cục bộ, chẳng hạn như PC
* Chúng bao gồm tất cả các chức năng cần thiết và không cần kết nối với mạng.
* Ví dụ về các ứng dụng như vậy là các ứng dụng văn phòng trên PC, chương trình CAD, phần mềm thao tác ảnh, v.v.

*2/ Ứng dụng dựa trên giao dịch tương tác:*

* Đây là những ứng dụng thực thi trên máy tính từ xa và được người dùng truy cập từ máy tính hoặc thiết bị đầu cuối của riêng họ
* Lớp ứng dụng này cũng bao gồm các hệ thống kinh doanh, nơi một doanh nghiệp cung cấp quyền truy cập vào các hệ thống của nó thông qua trình duyệt web hoặc chương trình khách hàng đặc biệt và các dịch vụ dựa trên đám mây, chẳng hạn như chia sẻ thư và ảnh.
* Các ứng dụng tương tác thường kết hợp một kho dữ liệu lớn được truy cập và cập nhật trong mỗi giao dịch.

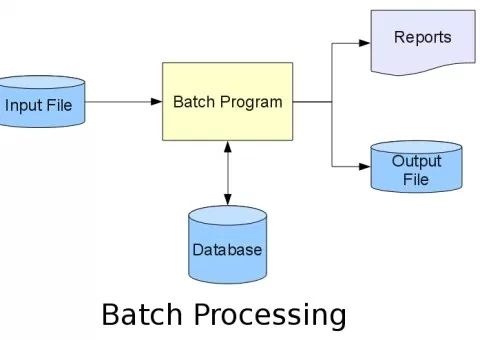
*3/* Các ứng dụng hệ thống điều khiển nhúng:

* Đây là những hệ thống kiểm soát phần mềm kiểm soát và quản lý các thiết bị phần cứng.
* Về mặt số lượng, có lẽ có nhiều hệ thống nhúng hơn bất kỳ loại hệ thống nào khác.

*Vd: máy ảnh, điện thoại,xe hơi,…*

*Các loại hệ thống xử lý phần mềm:*

***4/ Hệ thống xử lý hàng loạt:***

******

* Đây là những hệ thống kinh doanh được thiết kế để xử lý dữ liệu theo lô lớn.
* Chúng xử lý số lượng lớn các đầu vào riêng lẻ để tạo ra các đầu ra tương ứng.
* *Vd:* Ví dụ về các hệ thống lô bao gồm các hệ thống thanh toán định kỳ, chẳng hạn như hệ thống thanh toán qua điện thoại và hệ thống thanh toán tiền lương

***5/ Hệ thống giải trí:***

* Đây là những hệ thống chủ yếu cho mục đích sử dụng cá nhân và nhằm mục đích giải trí cho người dùng.
* Hầu hết các hệ thống này là trò chơi của loại này hay loại khác.
* Chất lượng của tương tác người dùng được cung cấp là đặc điểm phân biệt quan trọng nhất của hệ thống giải trí

***6/ Hệ thống mô hình hóa và mô phỏng***

* Đây là những hệ thống được phát triển bởi các nhà khoa học và kỹ sư để mô hình hóa các quá trình hoặc tình huống vật lý, bao gồm nhiều đối tượng tương tác riêng biệt.
* Chúng thường được tính toán chuyên sâu và yêu cầu các hệ thống song song hiệu năng cao để thực thi.

***7/ Hệ thống thu thập dữ liệu:***

* Đây là những hệ thống thu thập dữ liệu từ môi trường của chúng bằng cách sử dụng một bộ cảm biến và gửi dữ liệu đó đến các hệ thống khác để xử lý.
* Phần mềm phải tương tác với các cảm biến và thường được cài đặt trong môi trường thù địch như bên trong động cơ hoặc ở một vị trí ở xa.

***8/ Hệ thống các hệ thống***

* Đây là những hệ thống bao gồm một số hệ thống phần mềm khác.
* Một số trong số này có thể là các sản phẩm phần mềm chung, chẳng hạn như chương trình bảng tính. Các hệ thống khác trong hội đồng có thể được viết đặc biệt cho môi trường đó.

*🡪Trách nhiệm chuyên môn và đạo đức:*

* Kỹ nghệ phần mềm bao gồm trách nhiệm rộng hơn đơn giản là việc áp dụng các kỹ năng kỹ thuật.
* Các kỹ sư phần mềm phải hành xử một cách trung thực và có trách nhiệm về mặt đạo đức nếu họ được tôn trọng như các chuyên gia.
* Hành vi đạo đức hơn là đơn giản là duy trì luật.

# Đạo đức kỹ thuật phần mềm:

* Giống như các ngành kỹ thuật khác, kỹ nghệ phần mềm được thực hiện trong một khuôn khổ xã hội và pháp lý giới hạn sự tự do của những người làm việc trong khu vực đó.
* Là kỹ sư phần mềm, bạn phải chấp nhận rằng công việc của bạn liên quan đến trách nhiệm rộng lớn hơn là việc áp dụng các kỹ năng kỹ thuật.
* Bạn cũng phải cư xử một cách có trách nhiệm đạo đức và đạo đức nếu bạn được tôn trọng như một kỹ sư chuyên nghiệp.
* Nó đi mà không nói rằng bạn nên duy trì tiêu chuẩn bình thường của sự trung thực và toàn vẹn.
* Bạn không nên sử dụng các kỹ năng và khả năng của bạn để hành xử một cách không trung thực hoặc theo cách sẽ mang lại sự bất đồng cho nghề kỹ nghệ phần mềm.
* Tuy nhiên, có những lĩnh vực mà các tiêu chuẩn về hành vi chấp nhận được không bị ràng buộc bởi luật pháp mà bởi khái niệm trách nhiệm chuyên môn hơn

**\*Các vấn đề về trách nhiệm nghề nghiệp**:

1/Bảo mật: Thông thường nên tôn trọng tính bảo mật của nhà tuyển dụng hoặc khách hàng của bạn bất kể có hay không một thỏa thuận bảo mật chính thức đã được ký kết.

2/Năng lực:

* Bạn không nên xuyên tạc mức độ thẩm quyền của mình.
* Bạn không nên cố ý chấp nhận công việc nằm ngoài thẩm quyền của bạn.

3/Quyền sở hữu trí tuệ

* Các kỹ sư nên biết về luật pháp địa phương quản lý việc sử dụng tài sản trí tuệ như bằng sáng chế, bản quyền, v.v.
* Họ nên cẩn thận để đảm bảo rằng tài sản trí tuệ của nhà tuyển dụng và khách hàng được bảo vệ.

**\*Vấn đề về làm dụng máy tính:**

* Kỹ sư phần mềm không nên sử dụng các kỹ năng kỹ thuật của họ để lạm dụng máy tính của người khác.
* Phạm vi sử dụng sai máy tính từ tương đối tầm thường (trò chơi đang chơi trên máy của chủ nhân), cực kỳ nghiêm trọng (phổ biến vi-rút).

**\*Quy tắc đạo đức - lời mở đầu**

* Lời nói đầu:
* Phiên bản ngắn của mã tóm tắt nguyện vọng ở mức độ trừu tượng cao; các mệnh đề được bao gồm trong phiên bản đầy đủ đưa ra các ví dụ và chi tiết về cách những nguyện vọng này thay đổi cách chúng ta hành động như các chuyên gia kỹ thuật phần mềm. Nếu không có nguyện vọng, chi tiết có thể trở thành hợp pháp và tẻ nhạt; mà không có chi tiết, nguyện vọng có thể trở nên có vẻ cao nhưng trống rỗng; cùng với nhau, nguyện vọng và chi tiết tạo thành một mã gắn kết.
* Các kỹ sư phần mềm sẽ cam kết thực hiện việc phân tích, đặc tả, thiết kế, phát triển, kiểm thử và bảo trì phần mềm một nghề có lợi và được kính trọng. Theo cam kết của họ đối với sức khỏe, an toàn và phúc lợi của công chúng, các kỹ sư phần mềm phải tuân theo tám nguyên tắc sau đây:

***1/Quy tắc đạo đức ACM / IEEE***

* Các hiệp hội chuyên nghiệp ở Mỹ đã hợp tác để sản xuất một bộ luật thực hành đạo đức.
* Các thành viên của các tổ chức này đăng ký mã thực hành khi họ tham gia.
* Bộ luật gồm tám nguyên tắc liên quan đến hành vi và quyết định của các kỹ sư phần mềm chuyên nghiệp, bao gồm các học viên, nhà giáo dục, người quản lý, giám sát viên và nhà hoạch định chính sách, cũng như học viên và sinh viên nghề.

***2/Quy tắc đạo đức - nguyên tắc***

🡪CÔNG CỘNG:

Các kỹ sư phần mềm phải hành động nhất quán với lợi ích công cộng.

🡪KHÁCH HÀNG VÀ NGƯỜI LÀM VIỆC:

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.

🡪SẢN PHẨM:

Các kỹ sư phần mềm phải đảm bảo rằng các sản phẩm của họ và các sửa đổi liên quan đáp ứng các tiêu chuẩn chuyên nghiệp cao nhất có thể.

🡪JUDGMENT:

Các kỹ sư phần mềm sẽ duy trì tính toàn vẹn và độc lập trong bản án chuyên môn của họ.

* SỰ QUẢN LÝ:

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.

🡪CHUYÊN NGHIỆP:

Các kỹ sư phần mềm sẽ nâng cao tính toàn vẹn và danh tiếng của nghề phù hợp với lợi ích công cộng.

🡪ĐỒNG NGHIỆP

Các kỹ sư phần mềm phải công bằng và ủng hộ đồng nghiệp của họ.

🡪SELF

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.

**Bối cảnh đạo đức:**

* Bất đồng về nguyên tắc với các chính sách quản lý cấp cao.
* Chủ lao động của bạn hoạt động một cách phi đạo đức và phát hành một hệ thống an toàn quan trọng mà không hoàn thành việc kiểm tra hệ thống.
* Tham gia phát triển hệ thống vũ khí quân sự hoặc hệ thống hạt nhân.

# Chapter 3: Software Process

*🟑 4 Hoạt động cơ bản cho Kỹ thuật phần mềm*

## 3.1 Đặc điểm kỹ thuật phần mềm

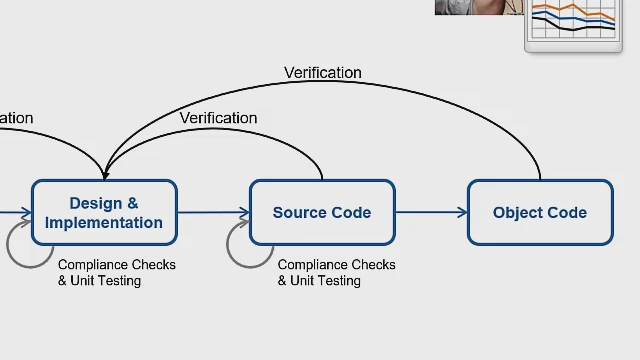
Chức năng của phần mềm và các ràng buộc trong hoạt động của nó phải được xác định.

## 3.2. Thiết kế & triển khai phần mềm

Phần mềm để đáp ứng các đặc điểm kỹ thuật phải được sản xuất

## 3.3. Xác nhận phần mềm

Phần mềm phải được xác thực để đảm bảo rằng phần mềm đó thực hiện những gì khách hàng muốn.



## 3.4. Tiến hóa phần mềm

Phần mềm phải phát triển để đáp ứng nhu cầu thay đổi của khách hàng.

Trong một số hình thức, các hoạt động này là một phần của tất cả các quy trình phần mềm.

Trong thực tế, tất nhiên, chúng là những hoạt động phức tạp trong bản thân và bao gồm các hoạt động phụ như xác nhận yêu cầu, thiết kế kiến trúc, thử nghiệm đơn vị, v.v.

Ngoài ra còn có các hoạt động hỗ trợ quy trình như quản lý cấu hình tài liệu và phần mềm

Khi chúng tôi mô tả và thảo luận các quy trình, chúng tôi thường nói về các hoạt động trong các quy trình này như chỉ định mô hình dữ liệu, thiết kế giao diện người dùng, v.v. và thứ tự của các hoạt động này.

Tuy nhiên, cũng như các hoạt động, mô tả quy trình cũng có thể bao gồm:

* Các sản phẩm
  + Đó là kết quả của một hoạt động quy trình.
  + Ví dụ, kết quả của hoạt động thiết kế kiến trúc có thể là một mô hình của kiến trúc phần mềm.
* Vai trò
  + Phản ánh trách nhiệm của những người liên quan đến quy trình.
  + Ví dụ về vai trò là người quản lý dự án, người quản lý cấu hình, lập trình viên, v.v.
* Trước và sau điều kiện

Đó là những tuyên bố đúng trước và sau khi một hoạt động quy trình đã được ban hành hoặc một sản phẩm được sản xuất.

Ví dụ, trước khi thiết kế kiến trúc bắt đầu, một điều kiện trước có thể là tất cả các yêu cầu đã được khách hàng chấp thuận; sau khi hoạt động này kết thúc, một điều kiện sau có thể là các mô hình UML mô tả kiến trúc đã được xem xét.

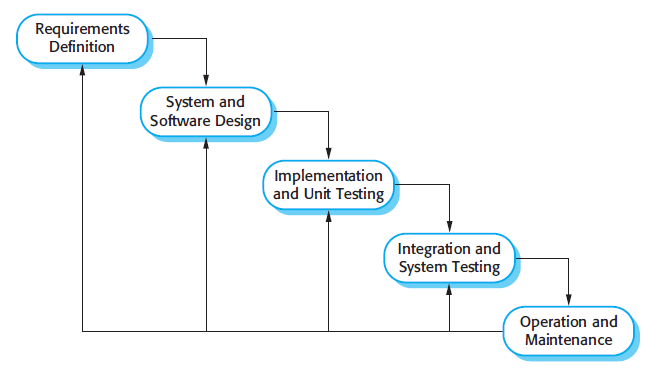
## 3.5 Software Process Model

### 3.5.1 Mô hình tuyến tính

Các mô hình tuyến tính phù hợp cho các dự án mà tất cả các yêu cầu được xác định và hiểu rõ trước khi thiết kế phần mềm bắt đầu.

Hai loại mô hình tuyến tính là:

### 3.5.2 Mô hình thác nước(Waterfall model)



Mô hình xuất bản đầu tiên của quá trình phát triển phần mềm được bắt nguồn từ các quy trình kỹ thuật hệ thống tổng quát hơn (Royce, 1970).

Do thác từ giai đoạn này sang pha khác, mô hình này được gọi là "mô hình thác nước" hoặc vòng đời phần mềm.

Mô hình thác nước là một ví dụ về quy trình định hướng kế hoạch — về nguyên tắc, bạn phải lập kế hoạch và lên lịch cho tất cả các hoạt động của quá trình trước khi bắt đầu công việc trên chúng.

### 3.5.2 aCác giai đoạn chính của mô hình thác nước

**Yêu cầu Phân tích & Định nghĩa**

Các dịch vụ, ràng buộc và mục tiêu của hệ thống được thiết lập bằng cách tham vấn với người dùng hệ thống.

Sau đó, chúng được xác định chi tiết và phục vụ như một đặc tả hệ thống.

**Thiết kế hệ thống và phần mềm**

Quy trình thiết kế hệ thống phân bổ các yêu cầu cho các hệ thống phần cứng hoặc phần mềm bằng cách thiết lập một kiến ​​trúc hệ thống tổng thể.

Thiết kế phần mềm liên quan đến việc xác định và mô tả trừu tượng hệ thống phần mềm cơ bản và các mối quan hệ của chúng

**Thực hiện và kiểm tra đơn vị**

Trong giai đoạn này, thiết kế phần mềm được thực hiện như một tập hợp các chương trình hoặc các đơn vị chương trình. Kiểm tra đơn vị liên quan đến việc xác minh rằng mỗi đơn vị đáp ứng đặc điểm kỹ thuật của nó.

**Tích hợp và kiểm tra hệ thống**

Các chương trình hoặc chương trình riêng lẻ được tích hợp và thử nghiệm như một hệ thống hoàn chỉnh để đảm bảo rằng các yêu cầu phần mềm đã được đáp ứng.

Sau khi thử nghiệm, hệ thống phần mềm được phân phối cho khách hàng.

**Vận hành và bảo trì**

Đây là giai đoạn vòng đời dài nhất.

Hệ thống được lắp đặt và đưa vào sử dụng thực tế.

Việc bảo trì liên quan đến sửa lỗi không được phát hiện trong các giai đoạn trước của vòng đời, cải thiện việc triển khai các đơn vị hệ thống và tăng cường các dịch vụ của hệ thống khi các yêu cầu mới được phát hiện.

Mô hình thác được sử dụng trong các dự án nhỏ, nơi các yêu cầu được xác định ngay từ đầu.

Mô hình này giả định rằng các yêu cầu được ổn định và đóng băng trong kế hoạch dự án.

Tuy nhiên, điều này thường không đúng trong trường hợp các dự án lớn, nơi mà các yêu cầu có thể phát triển trong suốt quá trình phát triển.

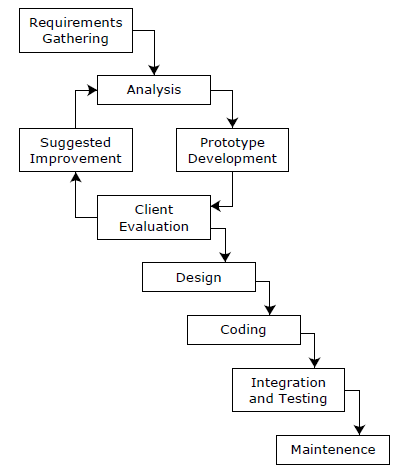
**Tạo mẫu(Prototyping Model)**

Mô hình Prototyping, còn được gọi là mô hình tiến hóa, đã đi vào hình ảnh vì những thất bại xảy ra trong phiên bản cuối cùng của ứng dụng phần mềm được phát triển bằng cách sử dụng mô hình thác nước.

Những thất bại thường xảy ra do sự thay đổi trong các yêu cầu của hệ thống được đề xuất hoặc do khoảng cách trong việc hiểu các yêu cầu của khách hàng bởi nhóm phát triển.

Một khoảng trống trong phiên bản đầu tiên của phần mềm ứng dụng được phát triển, chắc chắn, dẫn đến sự cần thiết phải làm lại ứng dụng.

Để khắc phục những hạn chế này, khái niệm tạo mẫu đã được giới thiệu.



Một hạn chế của mô hình Prototyping là nhóm phát triển đòi hỏi một chuyên gia cao cấp để phát triển một nguyên mẫu.

Nếu nhóm phát triển không đủ kinh nghiệm, nó có thể thỏa hiệp về chất lượng của hệ thống để có được nguyên mẫu hoạt động trong thời gian ngắn.

### 3.5.3 Mô hình lặp đi lặp lại(Iterative Models)

Các mô hình lặp đi lặp lại, một cải tiến so với mô hình Prototyping

Được sử dụng khi các yêu cầu cho phần mềm có khả năng phát triển trong suốt quá trình phát triển.

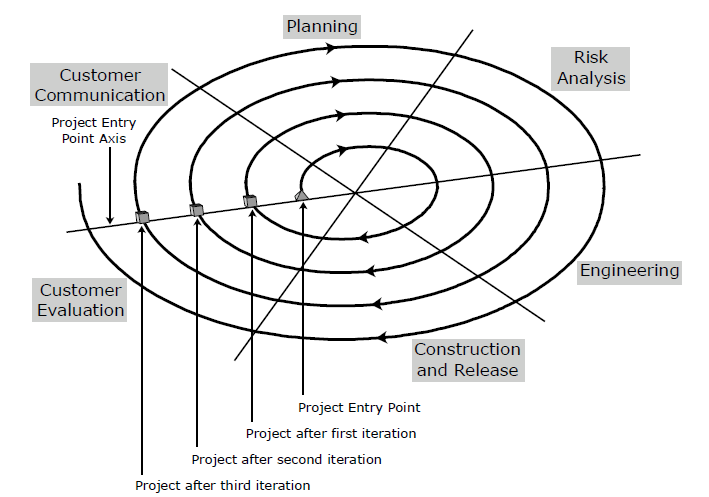
Các loại mô hình lặp lại là:

* Mô hình xoắn ốc(The Spiral Model)
* Mô hình xoắn ốc Win-win
* Mô hình phát triển dựa trên thành phần

### 3.5.4 The Spiral Model

Mô hình Xoắn ốc bao gồm bản chất lặp của mô hình Prototyping và bản chất tuyến tính của mô hình Thác.

Mô hình này lý tưởng cho việc phát triển phần mềm được phát hành trong các phiên bản khác nhau.



Trong mỗi lần lặp của mô hình Xoắn ốc, quy trình phát triển phần mềm tuân theo phương pháp tuyến tính phasewise.

Vào cuối lần lặp đầu tiên, khách hàng đánh giá phần mềm và cung cấp phản hồi.

Dựa trên đó, quy trình phát triển phần mềm đi vào bước lặp tiếp theo và sau đó theo cách tiếp cận tuyến tính để thực hiện phản hồi.

Quá trình lặp lại tiếp tục trong suốt vòng đời của phần mềm.

Một ví dụ về mô hình Xoắn ốc là sự phát triển của hệ điều hành Windows từ Window 3.1 đến Windows 10.

Microsoft đã phát triển hệ điều hành Windows 3.1 làm phiên bản đầu tiên của hệ điều hành.

Bạn có thể gọi đây là lần lặp đầu tiên trong mô hình Xoắn ốc. Sản phẩm được phát hành và đánh giá bởi khách hàng, trong đó bao gồm thị trường lớn.

Sau khi nhận được phản hồi từ khách hàng về Windows 3.1, Microsoft đã lên kế hoạch phát triển một phiên bản mới của hệ điều hành Windows.

Windows 95 được phát hành với những cải tiến và tính linh hoạt đồ họa. Tương tự như vậy, các phiên bản khác của hệ điều hành Windows đã được phát hành.

### 3.6 giai đoạn của mô hình xoắn ốc

***1.Giao tiếp khách hàng***

Bao gồm việc hiểu các yêu cầu hệ thống bằng cách liên lạc liên tục giữa khách hàng và nhà phân tích hệ thống.

***2.Lập kế hoạch***

Bao gồm ước tính lịch biểu, chi phí và tài nguyên cho dự án

***3.Phân tích rủi ro***

Bao gồm việc xác định, ước tính và theo dõi các rủi ro kỹ thuật và quản lý, chẳng hạn như trượt lịch và chi phí vượt quá.

***4.Kỹ thuật***

Bao gồm thu thập yêu cầu và thiết kế của hệ thống phần mềm

***5.Xây dựng và phát hành***

Bao gồm mã hóa, kiểm tra và triển khai phần mềm tại trang web của khách hàng và cung cấp tài liệu hỗ trợ người dùng.

***6.Đánh giá khách hàng***

Bao gồm đánh giá phần mềm của khách hàng và thực hiện phản hồi trong lần lặp tiếp theo về phát triển phần mềm

## 3.7 Mô hình phát triển dựa trên thành phần

Mô hình Phát triển dựa trên thành phần nhấn mạnh rộng rãi vào khả năng sử dụng lại của các thành phần.

Một thành phần là một phần có thể nhận dạng của một chương trình lớn hơn. Nó có thể là một hàm cụ thể hoặc một nhóm các hàm liên quan.

Trong mô hình Phát triển dựa trên thành phần, một hệ thống được chia thành các thành phần, lần lượt, được tạo thành từ các mô-đun.

Bạn có thể phát triển từng thành phần của một hệ thống phần mềm bằng cách sử dụng phương pháp tuyến tính hoặc lặp lại.

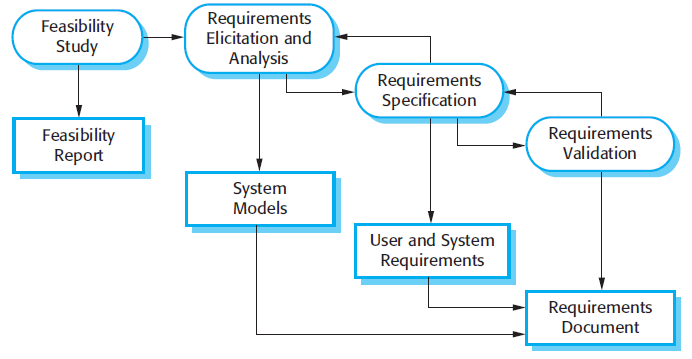
Tất cả các mô-đun liên quan tạo thành một thành phần được kiểm tra để đảm bảo rằng chúng hoạt động cùng nhau.

Bạn có thể tái sử dụng một thành phần và kết hợp nó với các thành phần khác trong cùng một hoặc các máy tính khác nhau trong một mạng phân phối để tạo thành một ứng dụng.

Ví dụ về các thành phần có thể bao gồm một máy tính lãi suất hiện có và thành phần tạo phần trả góp có thể được cắm vào phần mềm ứng dụng ngân hàng.

Tùy thuộc vào loại và số tiền cho vay và thời hạn mà khoản vay được yêu cầu, thành phần tính lãi suất tính lãi suất và thành phần tạo phần trả góp tạo ra gói trả góp theo chính sách ngân hàng.

**Process Activities**



## 3.8 Đặc điểm phần mềm

Đặc điểm kỹ thuật phần mềm hoặc kỹ thuật yêu cầu là quá trình hiểu và xác định các dịch vụ nào được yêu cầu từ hệ thống và xác định các ràng buộc về hoạt động và phát triển của hệ thống.

Yêu cầu kỹ thuật là một giai đoạn đặc biệt quan trọng của quá trình phần mềm như các lỗi ở giai đoạn này chắc chắn dẫn đến các vấn đề sau này trong việc thiết kế và thực hiện hệ thống

**Có 4 hoạt động chính trong quy trình kỹ thuật yêu cầu**

*Nghiên cứu khả thi*

Một ước tính được thực hiện cho dù nhu cầu người dùng được xác định có thể được thỏa mãn bằng cách sử dụng các công nghệ phần mềm và phần cứng hiện tại hay không.

Nghiên cứu xem xét liệu hệ thống được đề xuất sẽ có hiệu quả về mặt chi phí từ quan điểm kinh doanh hay không và liệu nó có thể được phát triển trong các ràng buộc ngân sách hiện có hay không.

Một nghiên cứu khả thi nên tương đối rẻ và nhanh chóng. Kết quả sẽ thông báo quyết định có nên tiến hành phân tích chi tiết hơn hay không

*Yêu cầu Phân tích & Phân tích*

Đây là quá trình tạo ra các yêu cầu hệ thống thông qua việc quan sát các hệ thống hiện có, các cuộc thảo luận với người dùng tiềm năng và người quản lý, phân tích nhiệm vụ, v.v.

Điều này có thể liên quan đến sự phát triển của một hoặc nhiều mô hình hệ thống và nguyên mẫu. Những điều này giúp bạn hiểu được hệ thống được chỉ định.

*Yêu cầu kỹ thuật*

Yêu cầu đặc điểm kỹ thuật là hoạt động dịch các thông tin được thu thập trong hoạt động phân tích thành một tài liệu xác định một tập hợp các yêu cầu

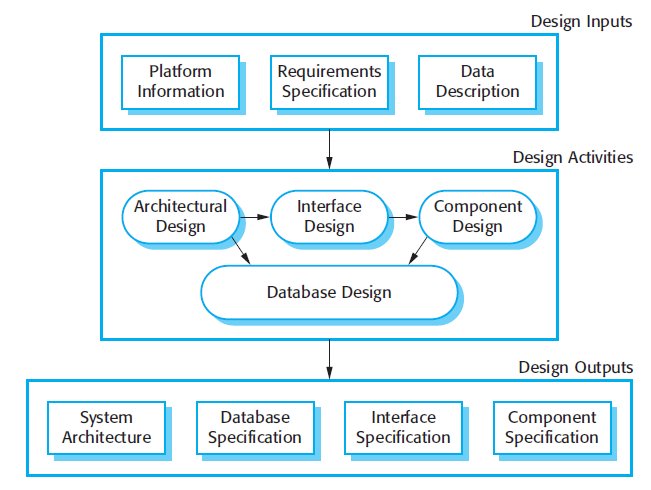
Hai loại yêu cầu có thể được bao gồm trong tài liệu này. Các yêu cầu của người dùng là các câu lệnh trừu tượng về các yêu cầu hệ thống cho khách hàng và người dùng cuối của hệ thống; yêu cầu hệ thống là mô tả chi tiết hơn về chức năng được cung cấp.

*Xác nhận yêu cầu*

Hoạt động này kiểm tra các yêu cầu về chủ nghĩa hiện thực, nhất quán và đầy đủ.

Trong quá trình này, các lỗi trong tài liệu yêu cầu chắc chắn được phát hiện. Sau đó nó phải được sửa đổi để sửa những vấn đề này.

## 3.9 Software Design & Implementation



Giai đoạn thực hiện phát triển phần mềm là quá trình chuyển đổi một đặc tả hệ thống thành một hệ thống thực thi.

Nó luôn luôn liên quan đến các quá trình thiết kế và lập trình phần mềm nhưng, nếu một cách tiếp cận gia tăng để phát triển được sử dụng, cũng có thể liên quan đến việc tinh chỉnh đặc tả phần mềm

Thiết kế phần mềm là mô tả cấu trúc của phần mềm được triển khai, các mô hình dữ liệu và cấu trúc được hệ thống sử dụng, các giao diện giữa các thành phần hệ thống và đôi khi, các thuật toán được sử dụng.

Các nhà thiết kế không đạt được thiết kế đã hoàn thành ngay lập tức mà phát triển thiết kế lặp đi lặp lại.

Họ thêm hình thức và chi tiết khi họ phát triển thiết kế của họ với backtracking liên tục để sửa thiết kế trước đó.

Hầu hết các giao diện phần mềm với các hệ thống phần mềm khác. Chúng bao gồm hệ điều hành, cơ sở dữ liệu, phần mềm trung gian và các hệ thống ứng dụng khác. Chúng tạo nên 'nền tảng phần mềm', môi trường mà phần mềm sẽ thực hiện.

Thông tin về nền tảng này là một đầu vào thiết yếu cho quá trình thiết kế, vì các nhà thiết kế phải quyết định cách tích hợp nó tốt nhất với môi trường của phần mềm

Các đặc tả yêu cầu là một mô tả các chức năng phần mềm phải cung cấp và hiệu suất và độ tin cậy yêu cầu của nó.

Nếu hệ thống xử lý dữ liệu hiện có, thì mô tả dữ liệu đó có thể được bao gồm trong đặc tả nền tảng; nếu không, mô tả dữ liệu phải là một đầu vào cho quá trình thiết kế để tổ chức hệ thống dữ liệu được xác định.

## 3.10 Activities may be part of the design process

*Thiết kế kiến ​​trúc*

Nơi bạn xác định cấu trúc tổng thể của hệ thống, các thành phần chính (đôi khi được gọi là các hệ thống con hoặc mô-đun), mối quan hệ của chúng và cách chúng được phân phối.

*Thiết kế giao diện*

Nơi bạn xác định giao diện giữa các thành phần hệ thống.

*Đặc tả giao diện này phải rõ ràng.*

Với giao diện chính xác, một thành phần có thể được sử dụng mà không cần các thành phần khác phải biết cách triển khai nó.

Khi các đặc tả giao diện được thỏa thuận, các thành phần có thể được thiết kế và phát triển đồng thời.

*Thiết kế thành phần*

Nơi bạn lấy từng thành phần hệ thống và thiết kế nó sẽ hoạt động như thế nào.

Đây có thể là một tuyên bố đơn giản về chức năng dự kiến ​​sẽ được thực hiện, với thiết kế cụ thể còn lại cho lập trình viên.

Ngoài ra, nó có thể là một danh sách các thay đổi được thực hiện cho một thành phần tái sử dụng hoặc một mô hình thiết kế chi tiết.

Mô hình thiết kế có thể được sử dụng để tự động tạo ra một triển khai thực hiện.

*Thiết kế cơ sở dữ liệu*

Nơi bạn thiết kế cấu trúc dữ liệu hệ thống và cách chúng được biểu diễn trong cơ sở dữ liệu.

Một lần nữa, công việc ở đây phụ thuộc vào việc một cơ sở dữ liệu hiện có có được tái sử dụng hay một cơ sở dữ liệu mới sẽ được tạo ra hay không.

Các hoạt động này dẫn đến một tập hợp các đầu ra thiết kế, cũng được hiển thị trong hình trước

Các chi tiết và đại diện của những thay đổi đáng kể. Đối với các hệ thống quan trọng, các tài liệu thiết kế chi tiết trình bày các mô tả chính xác và chính xác của hệ thống phải được tạo ra

Sự phát triển của một chương trình để thực hiện hệ thống sau một cách tự nhiên từ các quy trình thiết kế hệ thống.

Mặc dù một số lớp của chương trình, chẳng hạn như hệ thống an toàn quan trọng, thường được thiết kế chi tiết trước khi thực hiện bất kỳ bắt đầu, nó là phổ biến hơn cho các giai đoạn sau của thiết kế và phát triển chương trình được xen kẽ.

Các công cụ phát triển phần mềm có thể được sử dụng để tạo ra một chương trình khung sườn từ một thiết kế. Điều này bao gồm mã để xác định và thực hiện các giao diện, và trong nhiều trường hợp, nhà phát triển chỉ cần thêm chi tiết về hoạt động của từng thành phần chương trình

Lập trình là một hoạt động cá nhân và không có quy trình chung thường được tuân theo.

Một số lập trình viên bắt đầu với các thành phần mà họ hiểu, phát triển chúng, và sau đó chuyển sang các thành phần ít được hiểu hơn.

Những người khác có cách tiếp cận ngược lại, để lại các thành phần quen thuộc cho đến cuối cùng bởi vì họ biết làm thế nào để phát triển chúng.

Một số nhà phát triển muốn xác định dữ liệu sớm trong quá trình, sau đó sử dụng dữ liệu này để thúc đẩy phát triển chương trình; những người khác để dữ liệu không được chỉ định càng lâu càng tốt.

Thông thường, các lập trình viên thực hiện một số kiểm tra mã họ đã phát triển.

Điều này thường cho thấy các lỗi chương trình phải được loại bỏ khỏi chương trình.

Điều này được gọi là gỡ lỗi. Kiểm tra lỗi và gỡ lỗi là các quy trình khác nhau.

Kiểm tra thiết lập sự tồn tại của các khuyết tật. Gỡ lỗi có liên quan với việc định vị và sửa chữa các lỗi này.

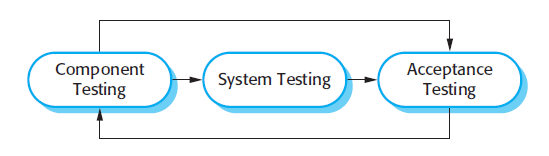
## 3.11 Xác nhận phần mềm

Xác nhận phần mềm hoặc, nói chung hơn, xác minh và xác thực (V & V) được dự định để cho thấy rằng một hệ thống vừa tuân theo đặc điểm kỹ thuật của nó và đáp ứng được sự mong đợi của khách hàng hệ thống

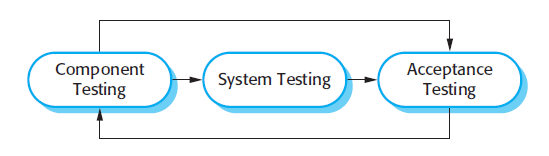
Kiểm thử chương trình, nơi hệ thống được thực hiện bằng cách sử dụng dữ liệu thử nghiệm mô phỏng, là kỹ thuật xác thực chính

Xác nhận cũng có thể bao gồm:

* Kiểm tra quá trình (như kiểm tra)
* Kiểm tra yêu cầu người dùng



## 3.12 Quy trình thử nghiệm ba giai đoạn



*Lý tưởng nhất là….*

Lỗi thành phần được phát hiện sớm trong quá trình và

Các vấn đề về giao diện được tìm thấy khi hệ thống được tích hợp

*Tuy nhiên….*

Khi các khiếm khuyết được phát hiện, chương trình phải được gỡ lỗi và điều này có thể yêu cầu các giai đoạn khác trong quá trình thử nghiệm được lặp lại.

Lỗi trong các thành phần chương trình có thể xuất hiện trong quá trình kiểm tra hệ thống. Do đó, quá trình này là một quá trình lặp đi lặp lại với thông tin được cung cấp từ các giai đoạn sau đến các phần trước của quy trình.

## 3.13 Các giai đoạn trong quá trình thử nghiệm

***Kiểm tra phát triển***

Các thành phần được kiểm tra bởi những người phát triển hệ thống

Mỗi thành phần được kiểm tra độc lập

Các thành phần có thể là các thực thể đơn giản như các hàm hoặc các lớp đối tượng

Các công cụ tự động hóa thử nghiệm như JUnit (Massol và Husted, 2003) có thể chạy lại các thử nghiệm thành phần khi các phiên bản mới của thành phần được tạo ra, thường được sử dụng

***Thử nghiệm hệ thống***

Các thành phần hệ thống được tích hợp để tạo ra một hệ thống hoàn chỉnh

Quá trình này liên quan đến việc tìm ra các lỗi do các tương tác không lường trước được giữa các thành phần và các vấn đề về giao diện thành phần.

Nó cũng quan tâm đến việc cho thấy rằng hệ thống đáp ứng các yêu cầu chức năng và phi chức năng của nó, và kiểm tra các thuộc tính hệ thống mới nổi.

Đối với các hệ thống lớn, đây có thể là một quá trình nhiều giai đoạn mà các thành phần được tích hợp để tạo thành các hệ thống con được kiểm tra riêng trước khi các hệ thống con này được tích hợp để tạo thành hệ thống cuối cùng.

***Kiểm tra chấp nhận***

Đây là giai đoạn cuối cùng trong quá trình thử nghiệm trước khi hệ thống được chấp nhận để sử dụng.

Hệ thống được kiểm tra với dữ liệu do khách hàng hệ thống cung cấp chứ không phải với dữ liệu thử nghiệm mô phỏng.

Kiểm tra chấp nhận có thể tiết lộ lỗi và thiếu sót trong định nghĩa yêu cầu hệ thống, bởi vì dữ liệu thực sự thực hiện hệ thống theo các cách khác nhau từ dữ liệu thử nghiệm.

Thử nghiệm chấp nhận cũng có thể tiết lộ các vấn đề về yêu cầu mà các cơ sở của hệ thống không thực sự đáp ứng nhu cầu của người dùng hoặc hiệu suất của hệ thống là không thể chấp nhận.

Kiểm tra chấp nhận đôi khi được gọi là 'alpha tesing'. Các hệ thống tùy chỉnh được phát triển cho một khách hàng.

Quá trình thử nghiệm alpha tiếp tục cho đến khi nhà phát triển hệ thống và khách hàng đồng ý rằng hệ thống được phân phối là việc triển khai các yêu cầu có thể chấp nhận được.

Khi một hệ thống được đưa ra thị trường như một sản phẩm phần mềm, một quá trình thử nghiệm được gọi là "beta testing" thường được sử dụng.

Thử nghiệm beta liên quan đến việc cung cấp hệ thống cho một số khách hàng tiềm năng đồng ý sử dụng hệ thống đó. Họ báo cáo vấn đề cho các nhà phát triển hệ thống

Điều này cho thấy sản phẩm sử dụng thực tế và phát hiện lỗi có thể không được dự đoán bởi các nhà xây dựng hệ thống.

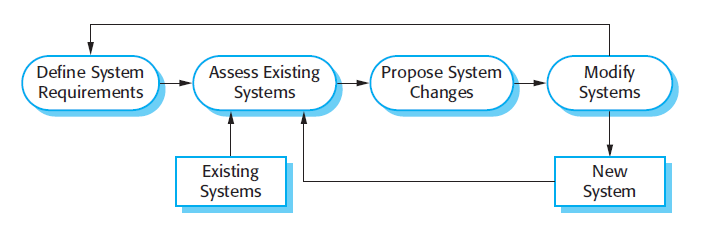
Sau phản hồi này, hệ thống được sửa đổi và phát hành để thử nghiệm beta tiếp theo hoặc để bán chung.

## 3.14 Tiến hóa phần mềm

Phần mềm vốn đã linh hoạt và có thể thay đổi.

Khi yêu cầu thay đổi thông qua việc thay đổi hoàn cảnh kinh doanh, phần mềm hỗ trợ doanh nghiệp cũng phải phát triển và thay đổi.

Mặc dù đã có một ranh giới giữa phát triển và tiến hóa (bảo trì) điều này ngày càng không thích hợp vì ít và ít hệ thống hoàn toàn mới.



Kỹ thuật phần mềm như là một quá trình tiến hóa, nơi phần mềm được liên tục thay đổi trong suốt cuộc đời của nó để đáp ứng với các yêu cầu thay đổi và nhu cầu của khách hàng.

## 3.15 Xác định vai trò và trách nhiệm

Những người khác nhau tương tác với nhau trong suốt giai đoạn phân tích, phát triển và triển khai của một dự án CNTT, thực hiện nhiều vai trò khác nhau trong các tổ chức tương ứng của họ

Nói chung, những người và vai trò này có thể được phân loại thành vai trò tổ chức khách hàng và vai trò tổ chức của nhà cung cấp.



### 3.15.1 Vai trò & trách nhiệm tại các giai đoạn SDLC

***Giai đoạn khởi đầu***

Giai đoạn bắt đầu tập trung vào các khía cạnh "những gì và tại sao" của quá trình chuyển đổi công nghệ.

Vai trò liên quan đến giai đoạn bắt đầu từ Tổ chức khách hàng là:

* Nhóm quản lý
* Đội ngũ tư vấn

Các vai trò khác nhau liên quan đến giai đoạn bắt đầu từ tổ chức nhà cung cấp là:

* Nhân viên bán hàng và tiếp thị
* Nhà phân tích thiết kế
* Kiến trúc sư

***Giai đoạn phân tích***

Giai đoạn phân tích tập trung vào việc thu thập thông tin, xác định các yêu cầu hệ thống và xây dựng các nguyên mẫu cho việc phát hiện các yêu cầu.

Giai đoạn phân tích cũng liên quan đến việc ưu tiên các yêu cầu, tạo ra và đánh giá các lựa chọn thay thế, và xem xét các khuyến nghị với ban quản lý.

Các vai trò khác nhau liên quan đến giai đoạn phân tích là:

* Người dùng
* Người quản lý
* Quản lý kinh doanh
* Nhà phân tích thiết kế

***Giai đoạn thiết kế***

Giai đoạn thiết kế bao gồm các chi tiết của các mô-đun khác nhau và logic nội bộ của chúng.

Hoạt động thiết kế thường được chia thành hai giai đoạn riêng biệt:

* Thiết kế hệ thống
* Thiết kế chi tiết

Các vai trò khác nhau liên quan đến giai đoạn thiết kế là:

* Kiến trúc sư kiến trúc / giải pháp
* Quản lý dự án
* Người dùng
* Người quản lý

***Giai đoạn xây dựng***

Mục tiêu của giai đoạn xây dựng là dịch thiết kế của hệ thống thành một mã trong một ngôn ngữ lập trình nhất định.

Các vai trò khác nhau liên quan đến giai đoạn xây dựng là:

* Quản lý dự án
* Trưởng nhóm
* Tư vấn kỹ thuật hoặc Chuyên gia
* Lập trình viên hoặc nhà phát triển

***Giai đoạn thực hiện***

Trong giai đoạn thực hiện dự án hoàn thành và giải pháp được cài đặt tại trang web của khách hàng.

Các vai trò khác nhau liên quan đến giai đoạn thực hiện là:

* Người dùng
* Người quản lý
* Quản lý dự án
* Đội ngũ lãnh đạo, lập trình viên và chuyên gia kỹ thuật

***Vai trò và năng lực***

Vai trò của cả tổ chức khách hàng và nhà cung cấp có năng lực kỹ thuật và hành vi cụ thể là:

* Nhà phân tích thiết kế
* Kiến trúc sư
* Giám đốc phát triển kinh doanh
* Lập trình viên
* Quản lý dự án
* Trưởng nhóm
* Chuyên gia kỹ thuật

***Đảm bảo chất lượng***

Để đảm bảo chất lượng trong quá trình phát triển phần mềm, bạn cần phải thực hiện các hoạt động đảm bảo chất lượng phần mềm ở từng giai đoạn của vòng đời phát triển phần mềm.

Hệ thống chất lượng thường bao gồm một bộ các nguyên tắc mô tả cấu trúc tổ chức, trách nhiệm, thủ tục, quy trình, tiêu chuẩn và định dạng để thực hiện đảm bảo chất lượng.

# Chapter 4 :Software Project Management

Quan tâm đến các hoạt động liên quan đến việc đảm bảo rằng phần mềm được phân phối đúng thời gian và đúng tiến độ và phù hợp với yêu cầu của các tổ chức phát triển và mua sắm phần mềm.

Quản lý dự án là cần thiết bởi vì phát triển phần mềm luôn chịu sự ràng buộc về ngân sách và lịch biểu được thiết lập bởi tổ chức phát triển phần mềm.

**Tiêu chí thành công**

1. Cung cấp phần mềm cho khách hàng theo thời gian đã thỏa thuận.
2. Giữ chi phí tổng thể trong phạm vi ngân sách.
3. Cung cấp phần mềm đáp ứng kỳ vọng của khách hàng.
4. Duy trì một đội ngũ phát triển hạnh phúc và hoạt động tốt.

## 4.1 Phân biệt quản lý phần mềm

**Sản phẩm là vô hình.(The product is intangible.)**

Phần mềm không thể được nhìn thấy hoặc chạm vào. Người quản lí dự án phần mềm không thể thấy tiến bộ bằng cách đơn giản nhìn vào vật phẩm đang được xây dựng.

**Nhiều dự án phần mềm là các dự án 'một lần'.(Many software projects are 'one-off' projects.)**

Các dự án phần mềm lớn thường khác nhau theo một số cách từ các dự án trước đó. Ngay cả những người quản lý có nhiều kinh nghiệm trước đây cũng có thể thấy khó dự đoán vấn đề.

**Các quy trình phần mềm là biến và tổ chức cụ thể.(Software processes are variable and organization specific.)**

Chúng tôi vẫn không thể dự đoán một cách đáng tin cậy khi một quy trình phần mềm cụ thể có thể dẫn đến các vấn đề phát triển.

## 4.2 Hoạt động quản lý

**Lập kế hoạch dự án**

Các nhà quản lý dự án chịu trách nhiệm lập kế hoạch, ước tính và lập kế hoạch phát triển dự án và phân công mọi người vào các nhiệm vụ.

**Báo cáo**

Quản lý dự án thường chịu trách nhiệm báo cáo về tiến độ của dự án cho khách hàng và cho người quản lý của công ty phát triển phần mềm.

**Quản lý rủi ro**

Người quản lí dự án đánh giá những rủi ro có thể ảnh hưởng đến một dự án, theo dõi những rủi ro này và hành động khi có vấn đề nảy sinh.

**Quản lý nhân sự**

Người quản lý dự án phải chọn mọi người cho nhóm của họ và thiết lập các cách làm việc dẫn đến hiệu suất nhóm hiệu quả

**Đề nghị bằng văn bản(Proposal writing)**

Giai đoạn đầu tiên trong một dự án phần mềm có thể liên quan đến việc viết một đề xuất để giành được hợp đồng để thực hiện một mục công việc. Đề xuất này mô tả các mục tiêu của dự án và nó sẽ được thực hiện như thế nào.

## 4.3 Quản lý rủi ro

Quản lý rủi ro liên quan đến việc xác định các rủi ro và lập các kế hoạch để giảm thiểu tác động của chúng đối với một dự án.

Rủi ro là xác suất mà một số trường hợp bất lợi sẽ xảy ra

1. Rủi ro dự án ảnh hưởng đến lịch biểu hoặc tài nguyên
2. Rủi ro sản phẩm ảnh hưởng đến chất lượng hoặc hiệu suất của phần mềm đang được phát triển;
3. Rủi ro kinh doanh ảnh hưởng đến việc tổ chức phát triển hoặc mua sắm phần mềm.

## 4.4 Quy trình quản lý rủi ro

**Nhận diện rủi ro**

Xác định các rủi ro của dự án, sản phẩm và kinh doanh

**Phân tích rủi ro**

Đánh giá khả năng và hậu quả của những rủi ro này

**Lập kế hoạch rủi ro**

Lập kế hoạch để tránh hoặc giảm thiểu tác động của rủi ro

**Giám sát rủi ro**

Theo dõi các rủi ro trong suốt dự án

## 4.5 The Risk Management Process

****

**Nhận diện rủi ro**

Có thể là một hoạt động của nhóm hoặc dựa trên trải nghiệm của người quản lý dự án riêng lẻ.

Một danh sách kiểm tra các rủi ro phổ biến có thể được sử dụng để xác định các rủi ro trong một dự án

* Rủi ro công nghệ.
* Rủi ro con người.
* Rủi ro tổ chức.
* Yêu cầu rủi ro.
* Rủi ro ước tính.

## 4.6 Examples of Different Risk Types

|  |  |
| --- | --- |
| **Loại rủi ro** | **Rủi ro có thể xảy ra** |
| Technology | 1. Cơ sở dữ liệu được sử dụng trong hệ thống không thể xử lý nhiều giao dịch mỗi giây như mong đợi. 2. Các thành phần phần mềm có thể tái sử dụng chứa các khiếm khuyết có nghĩa là chúng không thể được tái sử dụng theo kế hoạch. |
| People | 1. Nó là không thể tuyển dụng nhân viên với các kỹ năng cần thiết. 2. Nhân viên chủ chốt bị bệnh và không có mặt vào những thời điểm quan trọng. 3. Huấn luyện bắt buộc cho nhân viên không có sẵn. |
| Organizational | 1. Tổ chức được tái cơ cấu để quản lý khác nhau chịu trách nhiệm cho dự án. 2. Các vấn đề tài chính của tổ chức buộc phải cắt giảm ngân sách của dự án. |
| Tools | 1. Mã được tạo ra bởi các công cụ tạo mã phần mềm là không hiệu quả. 2. Các công cụ phần mềm không thể làm việc cùng nhau theo cách được tích hợp. |
| Requirements | 1. Các thay đổi đối với các yêu cầu yêu cầu làm lại thiết kế lớn được đề xuất. 2. Khách hàng không hiểu được tác động của các thay đổi yêu cầu. |
| Estimation | 1. Thời gian cần thiết để phát triển phần mềm được đánh giá thấp. 2. Tỷ lệ sửa chữa lỗi được đánh giá thấp. 3. Kích thước của phần mềm được đánh giá thấp. |

## 4.7 Phân tích rủi ro

Đánh giá xác suất và mức độ nghiêm trọng của từng rủi ro.

Xác suất có thể rất thấp, thấp, trung bình, cao hoặc rất cao.

Rủi ro hậu quả có thể là thảm họa, nghiêm trọng, chấp nhận được hoặc không đáng kể.

## 4.8 Các loại rủi ro & ví dụ

|  |  |  |
| --- | --- | --- |
| **Rủi ro** | **Xác suất** | **Hiệu ứng** |
| Các vấn đề tài chính tổ chức buộc giảm ngân sách dự án **(7).** | Thấp | Thảm khốc |
| Không thể tuyển dụng nhân viên với các kỹ năng cần thiết cho dự án **(3).** | Cao | Thảm khốc |
| Nhân viên chủ chốt bị ốm vào những thời điểm quan trọng trong dự án **(4).** | Vừa phải | Nghiêm trọng |
| Các lỗi trong các thành phần phần mềm có thể tái sử dụng phải được sửa chữa trước khi các thành phần này được tái sử dụng **(2).** |
| Các thay đổi đối với các yêu cầu yêu cầu làm lại thiết kế lớn được đề xuất **(10).** |
| Tổ chức được tái cấu trúc để quản lý khác nhau chịu trách nhiệm cho dự án **(6).** | Cao |
| Cơ sở dữ liệu được sử dụng trong hệ thống không thể xử lý nhiều giao dịch mỗi giây như mong đợi **(1).** | Vừa phải |
| Thời gian cần thiết để phát triển phần mềm được đánh giá thấp **(12).** | Cao |
| Không thể tích hợp các công cụ phần mềm **(9).** | Vừa vừa |
| Khách hàng không hiểu tác động của các thay đổi yêu cầu **(11).** | Vừa phải |
| Huấn luyện bắt buộc cho nhân viên không có sẵn **(5).** |
| Tỷ lệ sửa chữa khiếm khuyết được đánh giá thấp **(13).** |
| Kích thước của phần mềm được đánh giá thấp **(14).** | Cao |
| Mã được tạo ra bởi các công cụ tạo mã không hiệu quả **(8).** | Vừa phải | Không có gì |

## 4.9 Lập kế hoạch rủi ro

***Xem xét từng rủi ro và phát triển một chiến lược để quản lý rủi ro đó.***

***Chiến lược tránh***

Xác suất rủi ro phát sinh sẽ giảm

***Chiến lược giảm thiểu***

Tác động của rủi ro đối với dự án hoặc sản phẩm sẽ bị giảm

***Kế hoạch dự phòng***

Nếu rủi ro phát sinh, kế hoạch dự phòng là các kế hoạch để đối phó với rủi ro đó

## 4.10 Các chiến lược để giúp quản lý rủi ro

|  |  |
| --- | --- |
| Rủi ro | Chiến lược |
| Vấn đề tổ chức tài chính | Chuẩn bị một tài liệu tóm tắt cho quản lý cấp cao cho thấy dự án đang đóng góp rất quan trọng vào mục tiêu của doanh nghiệp và trình bày lý do tại sao cắt giảm ngân sách dự án sẽ không hiệu quả về chi phí. |
| Vấn đề tuyển dụng | Thông báo cho khách hàng về những khó khăn tiềm ẩn và khả năng chậm trễ; điều tra các thành phần mua vào. |
| Nhân viên bệnh | Tổ chức lại nhóm để có nhiều công việc chồng chéo hơn và do đó mọi người hiểu được công việc của nhau. |
| Thành phần bị lỗi | Thay thế các thành phần có khả năng bị lỗi với các thành phần đã mua trong độ tin cậy đã biết. |
| Yêu cầu thay đổi | Thông tin truy xuất nguồn gốc để đánh giá tác động thay đổi yêu cầu; tối đa hóa thông tin ẩn trong thiết kế. |
| Chuyển dịch cơ cấu tổ chức | Chuẩn bị một tài liệu tóm tắt cho quản lý cấp cao cho thấy dự án đang đóng góp rất quan trọng như thế nào đối với mục tiêu của doanh nghiệp. |
| Hiệu suất cơ sở dữ liệu | Điều tra khả năng mua cơ sở dữ liệu hiệu suất cao hơn. |
| Đánh giá thấp thời gian phát triển | Điều tra các thành phần mua vào; điều tra việc sử dụng trình tạo chương trình. |

## 4.11 Giám sát rủi ro

Đánh giá từng rủi ro được xác định thường xuyên để quyết định xem nó có trở nên ít có khả năng xảy ra hay không.

Đồng thời đánh giá xem tác động của rủi ro có thay đổi hay không.

Mỗi rủi ro chính cần được thảo luận tại các cuộc họp tiến độ quản lý.

## 4.12 Chỉ số rủi ro

|  |  |
| --- | --- |
| **Loại rủi ro** | **Chỉ số tiềm năng** |
| Technology | Giao hàng tận nơi phần cứng hoặc phần mềm hỗ trợ; nhiều vấn đề công nghệ được báo cáo. |
| People | Tinh thần nhân viên kém; mối quan hệ nghèo giữa các thành viên trong nhóm; kim ngạch nhân viên cao. |
| Organizational | Tin đồn về tổ chức; thiếu hành động của quản lý cấp cao. |
| Tools | Sự tự do của các thành viên trong nhóm để sử dụng các công cụ; khiếu nại về các công cụ CASE; nhu cầu cho các máy trạm có công suất cao hơn. |
| Requirements | Nhiều yêu cầu thay đổi yêu cầu; khiếu nại của khách hàng. |
| Estimation | Không đáp ứng được thỏa thuận lịch trình; không xóa các lỗi được báo cáo. |

### 4.12.1 Những điểm chính

Quản lý dự án tốt là điều cần thiết nếu các dự án kĩ nghệ phần mềm được phát triển đúng tiến độ và trong ngân sách.

Quản lý phần mềm khác với quản lý kỹ thuật khác. Phần mềm là vô hình. Các dự án có thể là cuốn tiểu thuyết hoặc sáng tạo mà không có kinh nghiệm để hướng dẫn quản lý của họ. Các quy trình phần mềm không trưởng thành như các quy trình kỹ thuật truyền thống.

Quản lý rủi ro hiện được công nhận là một trong những nhiệm vụ quản lý dự án quan trọng nhất.

Quản lý rủi ro liên quan đến việc xác định và đánh giá các rủi ro của dự án để thiết lập xác suất mà chúng sẽ xảy ra và hậu quả cho dự án nếu phát sinh rủi ro đó. Bạn nên lập kế hoạch để tránh, quản lý hoặc đối phó với các rủi ro có thể xảy ra nếu hoặc khi chúng phát sinh.

## 4.13 Quản lý con người

Con người là tài sản quan trọng nhất của tổ chức.

Nhiệm vụ của người quản lý về cơ bản là định hướng con người. Trừ khi có một số hiểu biết về con người, quản lý sẽ không thành công.

Quản lý người nghèo là một đóng góp quan trọng cho sự thất bại của dự án.

### 4.13.1 Nhân tố quản lý con người

***Tính nhất quán***

Các thành viên trong nhóm tất cả nên được đối xử theo cách tương tự mà không có yêu thích hoặc phân biệt đối xử.

***Sự tôn trọng***

Các thành viên nhóm khác nhau có kỹ năng khác nhau và những khác biệt này nên được tôn trọng.

***Bao gồm***

Liên kết tất cả các thành viên trong nhóm và đảm bảo rằng lượt xem của mọi người được xem xét.

***Trung thực***

Bạn nên luôn trung thực về những gì đang diễn ra tốt đẹp và những gì đang diễn ra tồi tệ trong một dự án.

## 4.14 Thúc đẩy mọi người

Một vai trò quan trọng của người quản lý là thúc đẩy những người làm việc trong một dự án.

Động lực có nghĩa là tổ chức công việc và môi trường làm việc để khuyến khích mọi người làm việc hiệu quả.

*Nếu mọi người không có động lực, họ sẽ không quan tâm đến công việc họ đang làm. Họ sẽ làm việc chậm chạp, có nhiều khả năng phạm sai lầm và sẽ không đóng góp vào các mục tiêu rộng lớn hơn của đội hoặc tổ chức.*

Động lực là một vấn đề phức tạp nhưng dường như chúng là các loại động lực khác nhau dựa trên:

* Nhu cầu cơ bản (ví dụ: thực phẩm, giấc ngủ, v.v.);
* Nhu cầu cá nhân (ví dụ: tôn trọng, tự tin);
* Nhu cầu xã hội (ví dụ: được chấp nhận là một phần của nhóm).

## 4.15 Phân cấp nhu cầu con người

****

****

**Cần sự hài lòng**

Trong các nhóm phát triển phần mềm, nhu cầu sinh lý và an toàn cơ bản không phải là vấn đề.

Xã hội

Cung cấp các tiện ích chung;

Cho phép liên lạc không chính thức, ví dụ: qua mạng xã hội

Kính trọng

Công nhận thành tựu;

Phần thưởng phù hợp.

Tự thực hiện

Đào tạo - mọi người muốn tìm hiểu thêm;

Trách nhiệm.

### 4.15.1 Kiểu tính cách

Hệ thống phân cấp nhu cầu gần như chắc chắn là một động lực quá đơn giản trong thực tế.

Động lực cũng nên tính đến các loại tính cách khác nhau:

* Nhiệm vụ định hướng.
* Tự định hướng.
* Tương tác theo định hướng.

Người định hướng công việc

* Tạo ra kết quả mong muốn là ưu tiên
* Hãy quan tâm đến năng suất và hiệu quả.
* Tập trung vào danh sách việc cần làm của họ và những điều họ hy vọng sẽ đạt được.
* Có mục tiêu cụ thể và danh sách chi tiết.

### 4.15.2 Các loại tính cách tự định hướng

Công việc là một phương tiện để kết thúc là thành tích của các mục tiêu cá nhân - ví dụ: làm giàu, chơi tennis, đi du lịch vv;

Động lực chính là sự hiện diện và hành động của đồng nghiệp. Mọi người đi làm vì họ thích đi làm.

## 4.16 Cân bằng động lực

Động lực cá nhân được tạo thành từ các yếu tố của mỗi lớp.

Số dư có thể thay đổi tùy theo hoàn cảnh cá nhân và sự kiện bên ngoài.

Tuy nhiên, con người không chỉ được thúc đẩy bởi các yếu tố cá nhân mà còn là một phần của một nhóm và văn hóa.

Mọi người đi làm vì họ được thúc đẩy bởi những người mà họ làm việc cùng.

## 4.17 Teamwork

Phần lớn kỹ nghệ phần mềm là hoạt động nhóm

*Lịch phát triển cho hầu hết các dự án phần mềm không tầm thường là không thể hoàn thành bởi một người làm việc một mình.*

 Một nhóm tốt là gắn kết và có tinh thần đồng đội. Những người tham gia được thúc đẩy bởi sự thành công của nhóm cũng như các mục tiêu cá nhân của riêng họ.

Tương tác nhóm là yếu tố quyết định chính của hiệu suất nhóm.

Tính linh hoạt trong thành phần nhóm bị giới hạn

*Người quản lý phải làm tốt nhất họ có thể với những người có sẵn.*

## 4.18 Tính gắn kết nhóm

Trong một nhóm cố kết, các thành viên coi nhóm là quan trọng hơn bất kỳ cá nhân nào trong đó.

Những lợi thế của một nhóm cố kết là:

* Các tiêu chuẩn chất lượng nhóm có thể được phát triển bởi các thành viên nhóm.
* Các thành viên trong nhóm học hỏi lẫn nhau và làm quen với công việc của nhau; Sự ức chế gây ra bởi sự thiếu hiểu biết được giảm xuống.
* Kiến thức được chia sẻ. Có thể duy trì liên tục nếu một thành viên trong nhóm rời đi.
* Tái cấu trúc và cải tiến liên tục được khuyến khích. Các thành viên nhóm làm việc chung để cung cấp kết quả chất lượng cao và khắc phục sự cố, bất kể những cá nhân ban đầu đã tạo ra thiết kế hoặc chương trình.

## 4.19 Những lợi thế của một nhóm cố kết

Các tiêu chuẩn chất lượng nhóm có thể được phát triển bởi các thành viên nhóm.

Các thành viên trong nhóm học hỏi lẫn nhau và làm quen với công việc của nhau; Sự ức chế gây ra bởi sự thiếu hiểu biết được giảm xuống.

Kiến thức được chia sẻ. Có thể duy trì liên tục nếu một thành viên trong nhóm rời đi.

Tái cấu trúc và cải tiến liên tục được khuyến khích. Các thành viên nhóm làm việc chung để cung cấp kết quả chất lượng cao và khắc phục sự cố, bất kể những cá nhân ban đầu đã tạo ra thiết kế hoặc chương trình.

## 4.20 Hiệu quả của một đội

**Những người trong nhóm**

*Bạn cần một sự kết hợp của những người trong một nhóm dự án vì phát triển phần mềm liên quan đến các hoạt động đa dạng như đàm phán với khách hàng, lập trình, thử nghiệm và tài liệu.*

**Tổ chức nhóm**

*Một nhóm nên được tổ chức để các cá nhân có thể đóng góp vào khả năng và nhiệm vụ tốt nhất của họ có thể được hoàn thành như mong đợi.*

**Truyền thông kỹ thuật và quản lý**

*Giao tiếp tốt giữa các thành viên nhóm, và giữa nhóm kỹ sư phần mềm và các bên liên quan dự án khác, là điều cần thiết.*

## 4.21 Chọn thành viên nhóm

Người quản lý hoặc công việc của trưởng nhóm là tạo ra một nhóm gắn kết và tổ chức nhóm của họ để họ có thể làm việc cùng nhau hiệu quả.

Điều này liên quan đến việc tạo ra một nhóm với sự cân bằng quyền kỹ năng và tính cách kỹ thuật, và tổ chức nhóm đó để các thành viên làm việc cùng nhau hiệu quả.

## 4.22 Lắp ráp một đội

Có thể không thể chỉ định những người lý tưởng để làm việc trên một dự án

* Ngân sách dự án có thể không cho phép sử dụng nhân viên được trả lương cao;
* Nhân viên có kinh nghiệm thích hợp có thể không có sẵn;
* Một tổ chức có thể muốn phát triển các kỹ năng của nhân viên trong một dự án phần mềm.

Các nhà quản lý phải làm việc trong những hạn chế này đặc biệt là khi có sự thiếu hụt của các nhân viên được đào tạo.

## 4.23 Thành phần nhóm

Nhóm gồm các thành viên chia sẻ cùng một động lực có thể có vấn đề

* Định hướng công việc - mọi người đều muốn làm việc riêng của họ;
* Tự định hướng - mọi người đều muốn trở thành ông chủ;
* Tương tác theo định hướng - trò chuyện quá nhiều, không đủ công việc.

Một nhóm hiệu quả có một sự cân bằng của tất cả các loại.

Điều này có thể khó khăn để đạt được các kỹ sư phần mềm thường là nhiệm vụ theo định hướng.

Những người có khuynh hướng tương tác rất quan trọng vì họ có thể phát hiện và giải quyết những căng thẳng phát sinh.

## 4.24 Tổ chức nhóm

Cách thức tổ chức một nhóm ảnh hưởng đến các quyết định được thực hiện bởi nhóm đó, cách thức trao đổi thông tin và các tương tác giữa nhóm phát triển và các bên liên quan dự án bên ngoài.

Các câu hỏi chính bao gồm:

* *Người quản lí dự án có nên là người lãnh đạo kỹ thuật của nhóm không?*
* *Ai sẽ tham gia vào việc đưa ra các quyết định kỹ thuật quan trọng và chúng sẽ được thực hiện như thế nào?*
* *Tương tác với các bên liên quan bên ngoài và quản lý công ty cấp cao sẽ được xử lý như thế nào?*
* *Các nhóm có thể tích hợp những người không đồng vị trí như thế nào?*
* *Kiến thức có thể được chia sẻ như thế nào trong nhóm?*

Các nhóm kỹ nghệ phần mềm nhỏ thường được tổ chức một cách không chính thức mà không có cấu trúc cứng nhắc.

Đối với các dự án lớn, có thể có một cấu trúc phân cấp, trong đó các nhóm khác nhau chịu trách nhiệm cho các tiểu dự án khác nhau.

Phát triển nhanh nhẹn luôn dựa trên một nhóm không chính thức trên nguyên tắc cấu trúc chính thức ngăn cản trao đổi thông tin

## 4.25 Nhóm không chính thức

Nhóm hoạt động như một toàn thể và đi đến một sự đồng thuận về các quyết định ảnh hưởng đến hệ thống.

Trưởng nhóm hoạt động như giao diện bên ngoài của nhóm nhưng không phân bổ các mục công việc cụ thể.

Thay vào đó, công việc được thảo luận bởi cả nhóm và nhiệm vụ được phân bổ theo khả năng và kinh nghiệm.

Cách tiếp cận này thành công cho các nhóm mà tất cả các thành viên đều có kinh nghiệm và có năng lực.

## 4.26 Nhóm truyền thông

Giao tiếp tốt là điều cần thiết để nhóm làm việc hiệu quả.

Thông tin phải được trao đổi về tình trạng công việc, quyết định thiết kế và thay đổi các quyết định trước đó.

Giao tiếp tốt cũng tăng cường sự gắn kết nhóm vì nó thúc đẩy sự hiểu biết.

**Kích thước nhóm**

Nhóm càng lớn thì càng khó để mọi người giao tiếp với các thành viên khác trong nhóm.

**Cấu trúc nhóm**

Giao tiếp tốt hơn ở các nhóm có cấu trúc không chính thức so với các nhóm có cấu trúc phân cấp.

**Thành phần nhóm**

Giao tiếp tốt hơn khi có các loại cá tính khác nhau trong một nhóm và khi các nhóm được trộn lẫn chứ không phải là quan hệ tình dục đơn lẻ.

**Môi trường làm việc vật lý**

Tổ chức nơi làm việc tốt có thể giúp khuyến khích truyền thông.

**Key points**

Mọi người được thúc đẩy bởi sự tương tác với những người khác, sự công nhận của quản lý và đồng nghiệp của họ, và bằng cách được trao cơ hội phát triển cá nhân.

Các nhóm phát triển phần mềm nên khá nhỏ và gắn kết. Các yếu tố chính ảnh hưởng đến hiệu quả của một nhóm là những người trong nhóm đó, cách thức tổ chức và giao tiếp giữa các thành viên nhóm.

Truyền thông trong một nhóm bị ảnh hưởng bởi các yếu tố như trạng thái của các thành viên nhóm, kích thước của nhóm, thành phần giới tính của nhóm, tính cách và các kênh truyền thông có sẵn.

# Chapter 5: Requirement Engineering

## 5.1 Topics covered

* Các yêu cầu chức năng và phi chức năng
* Tài liệu yêu cầu phần mềm
* Yêu cầu kỹ thuật
* Yêu cầu quy trình kỹ thuật
* Yêu cầu gợi ý và phân tích
* Xác nhận yêu cầu
* Quản lý yêu cầu

## 5.2 Requirements Engineering

* Quá trình thiết lập các dịch vụ mà khách hàng yêu cầu từ một hệ thống và các ràng buộc mà theo đó nó hoạt động và được phát triển.
* Bản thân các yêu cầu là mô tả về các dịch vụ hệ thống và các ràng buộc được tạo ra trong quá trình kỹ thuật yêu cầu.

## 5.3 What Is a Requirement?

* Nó có thể nằm trong một tuyên bố trừu tượng mức cao của một dịch vụ hoặc một ràng buộc hệ thống đối với một đặc tả chức năng toán học chi tiết.
* Điều này là không thể tránh khỏi vì các yêu cầu có thể phục vụ một chức năng kép
  + Có thể là cơ sở cho một giá thầu cho một hợp đồng - do đó phải được mở để giải thích;
  + Có thể là cơ sở cho bản thân hợp đồng - do đó phải được xác định chi tiết;
  + Cả hai tuyên bố này có thể được gọi là yêu cầu.

## 5.4 Yêu cầu trừu tượng (Davis)

“Nếu một công ty muốn để cho một hợp đồng cho một dự án phát triển phần mềm lớn, nó phải xác định nhu cầu của mình một cách đầy đủ trừu tượng rằng một giải pháp không được xác định trước. Các yêu cầu phải được viết để một số nhà thầu có thể đấu thầu cho hợp đồng, cung cấp, có lẽ, các cách khác nhau để đáp ứng nhu cầu của tổ chức khách hàng. Khi hợp đồng đã được trao, nhà thầu phải viết một định nghĩa hệ thống cho khách hàng một cách chi tiết hơn để khách hàng hiểu và có thể xác nhận những gì phần mềm sẽ làm. Cả hai tài liệu này có thể được gọi là tài liệu yêu cầu cho hệ thống. ”

## 5.5 Các loại yêu cầu

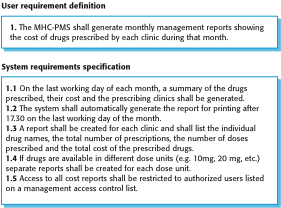
Yêu cầu người sử dụng

* Các phát biểu bằng ngôn ngữ tự nhiên cộng với sơ đồ của các dịch vụ mà hệ thống cung cấp và các ràng buộc hoạt động của nó. Viết cho khách hàng.

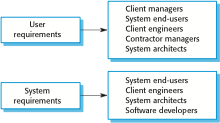
Yêu cầu hệ thống

* Một tài liệu có cấu trúc trình bày các mô tả chi tiết về các chức năng, dịch vụ và các ràng buộc hoạt động của hệ thống. Xác định những gì nên được thực hiện như vậy có thể là một phần của hợp đồng giữa khách hàng và nhà thầu.

## 5.6 Yêu cầu của người dùng và hệ thống

****

## 5.7 Độc giả của các loại yêu cầu khác nhau

****

## 5.8 Yêu cầu chức năng và phi chức năng

* Yêu cầu chức năng
  + Báo cáo về các dịch vụ mà hệ thống sẽ cung cấp, hệ thống sẽ phản ứng như thế nào với các đầu vào cụ thể và cách hệ thống hoạt động trong các tình huống cụ thể.
  + Có thể tuyên bố những gì hệ thống không nên làm.
* Những yêu cầu phi lý
  + Hạn chế về các dịch vụ hoặc chức năng được cung cấp bởi hệ thống như ràng buộc thời gian, ràng buộc về quy trình phát triển, tiêu chuẩn, v.v.
  + Thường áp dụng cho toàn bộ hệ thống chứ không phải là các tính năng hoặc dịch vụ riêng lẻ.
* Yêu cầu về miền
  + Các ràng buộc trên hệ thống từ miền hoạt động

## 5.9 Yêu cầu chức năng

* Mô tả chức năng hoặc dịch vụ hệ thống.
* Phụ thuộc vào loại phần mềm, người dùng được mong đợi và loại hệ thống mà phần mềm được sử dụng.
* Các yêu cầu của người dùng chức năng có thể là các báo cáo mức cao về những gì hệ thống nên làm.
* Yêu cầu hệ thống chức năng nên mô tả chi tiết các dịch vụ hệ thống.

## 5.10 Yêu cầu chức năng cho MHC-PMS

* Người dùng có thể tìm kiếm danh sách cuộc hẹn cho tất cả các phòng khám.
* Hệ thống sẽ tạo ra mỗi ngày, cho mỗi phòng khám, một danh sách các bệnh nhân được dự kiến ​​sẽ tham dự các cuộc hẹn ngày hôm đó.
* Mỗi nhân viên sử dụng hệ thống sẽ được xác định duy nhất bằng số nhân viên gồm 8 chữ số của mình.

## 5.11 Yêu cầu Imprecision(NULL)

* Các vấn đề phát sinh khi các yêu cầu không được nêu rõ.
* Các yêu cầu mơ hồ có thể được diễn giải theo nhiều cách khác nhau bởi các nhà phát triển và người dùng.
* Xem xét thuật ngữ 'tìm kiếm' theo yêu cầu 1
  + Mục đích của người dùng - tìm kiếm tên bệnh nhân trên tất cả các cuộc hẹn trong tất cả các phòng khám;
  + Diễn giải của nhà phát triển - tìm kiếm tên bệnh nhân trong một phòng khám riêng lẻ. Người dùng chọn phòng khám rồi tìm kiếm.

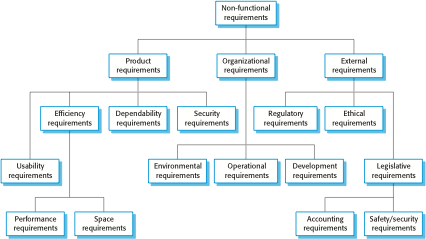
## 5.12 Yêu cầu đầy đủ và nhất quán

* Về nguyên tắc, các yêu cầu phải vừa hoàn chỉnh vừa nhất quán.
* Hoàn thành
  + Họ nên bao gồm mô tả của tất cả các cơ sở cần thiết.
* Thích hợp
  + Không nên có mâu thuẫn hoặc mâu thuẫn trong phần mô tả các cơ sở hệ thống.
* Trong thực tế, không thể tạo ra một tài liệu yêu cầu đầy đủ và nhất quán.

## 5.13 Những yêu cầu phi lý

* Xác định các thuộc tính hệ thống và các ràng buộc, ví dụ: độ tin cậy, thời gian đáp ứng và yêu cầu lưu trữ. Ràng buộc là khả năng thiết bị I / O, biểu diễn hệ thống, v.v.
* Yêu cầu quy trình cũng có thể được chỉ định bắt buộc một IDE, ngôn ngữ lập trình hoặc phương pháp phát triển cụ thể.
* Các yêu cầu phi chức năng có thể quan trọng hơn các yêu cầu chức năng. Nếu những điều này không được đáp ứng, hệ thống có thể vô ích.

## 5.16 Các loại yêu cầu phi chức năng

****

## 5.17 Thực hiện các yêu cầu phi chức năng

Các yêu cầu phi chức năng có thể ảnh hưởng đến kiến ​​trúc tổng thể của một hệ thống hơn là các thành phần riêng lẻ.

* Ví dụ, để đảm bảo rằng các yêu cầu về hiệu suất được đáp ứng, bạn có thể phải tổ chức hệ thống để giảm thiểu thông tin liên lạc giữa các thành phần.

Một yêu cầu không có chức năng, chẳng hạn như yêu cầu bảo mật, có thể tạo ra một số yêu cầu chức năng liên quan để xác định các dịch vụ hệ thống được yêu cầu.

* Nó cũng có thể tạo ra các yêu cầu hạn chế các yêu cầu hiện tại.

## 5.18 Phân loại phi chức năng

Yêu cầu sản phẩm

* Các yêu cầu xác định rằng sản phẩm được phân phối phải hoạt động theo một cách cụ thể, ví dụ: tốc độ thực thi, độ tin cậy, v.v.

Yêu cầu tổ chức

* Yêu cầu là hậu quả của chính sách và thủ tục của tổ chức, ví dụ: các tiêu chuẩn quy trình được sử dụng, các yêu cầu thực hiện, v.v.

Yêu cầu bên ngoài

* Các yêu cầu phát sinh từ các yếu tố bên ngoài hệ thống và quá trình phát triển của nó, ví dụ: các yêu cầu về khả năng tương tác, các yêu cầu lập pháp, v.v.

## 5.19 Ví dụ về các yêu cầu phi chức năng trong MHC-PMS

Yêu cầu sản phẩm

MHC-PMS sẽ có sẵn cho tất cả các phòng khám trong giờ làm việc bình thường (Thứ Hai – Thứ Sáu, 08:30–17,30). Thời gian ngừng hoạt động trong giờ làm việc bình thường không được vượt quá năm giây trong bất kỳ ngày nào.

Yêu cầu về tổ chức Người sử dụng hệ thống MHC-PMS sẽ tự xác thực bằng cách sử dụng thẻ căn cước y tế của họ.

Yêu cầu bên ngoài Hệ thống sẽ thực hiện các quy định về quyền riêng tư của bệnh nhân như được quy định trong HStan-03-2006-priv.

## 5.20 Mục tiêu và yêu cầu

* Các yêu cầu phi chức năng có thể rất khó để nêu rõ các yêu cầu chính xác và không chính xác có thể khó xác minh.
* Mục tiêu
  + Một ý định chung của người dùng như dễ sử dụng.
* Yêu cầu phi chức năng có thể xác minh
  + Một tuyên bố sử dụng một số biện pháp có thể được kiểm tra khách quan.
* Mục tiêu là hữu ích cho các nhà phát triển khi họ truyền đạt ý định của người dùng hệ thống.

## 5.21 Yêu cầu về tính khả dụng

* Hệ thống nên dễ sử dụng bởi nhân viên y tế và nên được tổ chức theo cách mà các lỗi người dùng được giảm thiểu. (Mục tiêu)
* Nhân viên y tế có thể sử dụng tất cả các chức năng của hệ thống sau bốn giờ huấn luyện. Sau khóa đào tạo này, số lỗi trung bình do người dùng có kinh nghiệm thực hiện không được vượt quá hai giờ mỗi giờ sử dụng hệ thống. (Yêu cầu phi chức năng có thể kiểm tra)

## 5.22 Số liệu để chỉ định các yêu cầu phi chức năng

* Speed: Giao dịch đã xử lý / giây. Thời gian phản hồi của người dùng / sự kiện. Thời gian làm mới màn hình
* Size: Mbyte. Số lượng chip ROM
* Easy of use: Thời gian huấn luyện. Số lượng khung trợ giúp
* Reliability ( độ tin cậy): Thời gian trung bình để thất bại. Xác suất không có sẵn. Tỷ lệ xảy ra lỗi. Khả dụng
* Robustness ( độ mạnh): Thời gian khởi động lại sau thất bại. Tỷ lệ phần trăm các sự kiện gây ra lỗi. Xác suất tham nhũng dữ liệu trên thất bại
* Portability ( tính di động): Tỷ lệ phần trăm của các báo cáo phụ thuộc mục tiêu. Số lượng hệ thống đích

## 5.23 Yêu cầu về miền

* Miền hoạt động của hệ thống áp đặt các yêu cầu trên hệ thống.
  + Ví dụ, một hệ thống điều khiển xe lửa phải tính đến đặc điểm phanh trong các điều kiện thời tiết khác nhau.
* Yêu cầu miền là các yêu cầu chức năng mới, các ràng buộc đối với các yêu cầu hiện có hoặc xác định các tính toán cụ thể.
* Nếu yêu cầu miền không hài lòng, hệ thống có thể không hoạt động được.

## 5.24 Hệ thống bảo vệ tàu hỏa

* Đây là yêu cầu miền cho hệ thống bảo vệ tàu:
* Sự giảm tốc của tàu sẽ được tính như sau:
  + Dtrain = Dcontrol + Dgradient
  + trong đó Dgradient là 9,81ms2 \* bù gradient / alpha và các giá trị 9,81ms2 / alpha được biết đến với các loại tàu khác nhau.
* Rất khó cho một người không phải là chuyên gia để hiểu những tác động của điều này và cách nó tương tác với các yêu cầu khác.

## 5.25 Các vấn đề về yêu cầu miền

Hiểu được

* Yêu cầu được thể hiện bằng ngôn ngữ của miền ứng dụng;
* Điều này thường không được hiểu bởi các kỹ sư phần mềm phát triển hệ thống.

Ngụ ý

* Các chuyên gia tên miền hiểu rõ khu vực này đến mức họ không nghĩ đến việc yêu cầu miền rõ ràng.

## 5.26 Key points (chốt bài)

* Yêu cầu đối với một hệ thống phần mềm đặt ra những gì hệ thống nên làm và xác định các ràng buộc về hoạt động và thực hiện của nó.
* Các yêu cầu chức năng là các câu lệnh của các dịch vụ mà hệ thống phải cung cấp hoặc là các mô tả về cách thực hiện một số tính toán.
* Các yêu cầu phi chức năng thường hạn chế hệ thống đang được phát triển và quá trình phát triển đang được sử dụng.
* Chúng thường liên quan đến các đặc tính nổi bật của hệ thống và do đó áp dụng cho toàn bộ hệ thống.

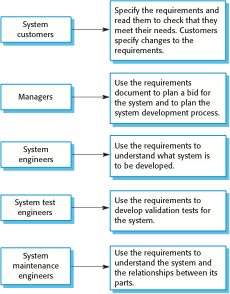
**Chapter 5 (tt): Software Requirements Document**

* Tài liệu yêu cầu phần mềm là tuyên bố chính thức về những gì được yêu cầu của các nhà phát triển hệ thống.
* Nên bao gồm cả định nghĩa các yêu cầu của người dùng và đặc điểm kỹ thuật của các yêu cầu hệ thống.
* Nó KHÔNG phải là một tài liệu thiết kế. Càng nhiều càng tốt, nó nên thiết lập những gì hệ thống nên làm thay vì CÁCH nó nên làm điều đó.

## 5.27 Phương thức và yêu cầu nhanh

* Nhiều phương pháp nhanh nhẹn cho rằng việc tạo ra một tài liệu yêu cầu là một sự lãng phí thời gian khi các yêu cầu thay đổi nhanh chóng.
* Do đó, tài liệu luôn lỗi thời.
* Các phương pháp như XP sử dụng kỹ thuật yêu cầu gia tăng và các yêu cầu rõ ràng là ‘câu chuyện của người dùng’
* Điều này là thiết thực cho các hệ thống kinh doanh nhưng có vấn đề đối với các hệ thống yêu cầu nhiều phân tích trước khi phân phối (ví dụ: các hệ thống quan trọng) hoặc các hệ thống do nhiều nhóm phát triển.

## 5.28 Người dùng tài liệu yêu cầu

****

## 5.29 Yêu cầu thay đổi tài liệu

* Thông tin trong tài liệu yêu cầu phụ thuộc vào loại hệ thống và cách tiếp cận để phát triển được sử dụng.
* Các hệ thống phát triển dần dần, thông thường, có ít chi tiết hơn trong tài liệu yêu cầu.
* Yêu cầu các tiêu chuẩn tài liệu đã được thiết kế, ví dụ: Tiêu chuẩn IEEE. Đây là chủ yếu áp dụng cho các yêu cầu cho các dự án kỹ thuật hệ thống lớn.

## 5.30 Cấu trúc của tài liệu yêu cầu

* Preface (lời nói đầu): Điều này sẽ xác định độc giả dự kiến ​​của tài liệu và mô tả lịch sử phiên bản của nó, bao gồm lý do để tạo phiên bản mới và tóm tắt các thay đổi được thực hiện trong mỗi phiên bản.
* Introduction (giới thiệu): Điều này sẽ mô tả sự cần thiết của hệ thống. Nó sẽ mô tả ngắn gọn các chức năng của hệ thống và giải thích cách nó hoạt động với các hệ thống khác. Nó cũng nên mô tả cách hệ thống phù hợp với mục tiêu kinh doanh tổng thể hoặc mục tiêu chiến lược của tổ chức vận hành phần mềm.
* Glossary ( bảng chú giải): Điều này nên xác định các thuật ngữ kỹ thuật được sử dụng trong tài liệu. Bạn không nên đưa ra các giả định về kinh nghiệm hay chuyên môn của người đọc.
* User requirements definition ( định nghĩa yêu cầu người dùng): Ở đây, bạn mô tả các dịch vụ được cung cấp cho người dùng. Các yêu cầu hệ thống phi chức năng cũng nên được mô tả trong phần này. Mô tả này có thể sử dụng ngôn ngữ tự nhiên, biểu đồ hoặc các ký hiệu khác dễ hiểu đối với khách hàng. Các tiêu chuẩn sản phẩm và quy trình phải tuân thủ phải được chỉ định.
* System architecture ( kiến trúc hệ thống): Chương này sẽ trình bày tổng quan cấp cao về kiến ​​trúc hệ thống dự kiến, cho thấy sự phân bố các chức năng trên các mô-đun hệ thống. Các thành phần kiến ​​trúc được tái sử dụng nên được làm nổi bật.

## 5.31 Cấu trúc của tài liệu yêu cầu

* System requirements specification ( đặc tả): Điều này sẽ mô tả các yêu cầu chức năng và phi chức năng chi tiết hơn. Nếu cần, chi tiết hơn nữa cũng có thể được thêm vào các yêu cầu phi chức năng. Giao diện với các hệ thống khác có thể được xác định.
* System models ( mô hình hệ thống): Điều này có thể bao gồm các mô hình hệ thống đồ họa cho thấy mối quan hệ giữa các thành phần hệ thống và hệ thống và môi trường của nó. Ví dụ về các mô hình có thể là mô hình đối tượng, mô hình luồng dữ liệu hoặc mô hình dữ liệu ngữ nghĩa.
* System evolution ( tiến hóa hệ thống): Điều này sẽ mô tả các giả định cơ bản mà hệ thống dựa vào và mọi thay đổi được dự đoán do sự phát triển phần cứng, thay đổi nhu cầu của người dùng, v.v. Phần này rất hữu ích cho các nhà thiết kế hệ thống vì nó có thể giúp họ tránh các quyết định thiết kế có thể hạn chế các thay đổi trong tương lai đối với hệ thống.
* Appendices ( phụ lục): Những thông tin này sẽ cung cấp thông tin chi tiết, cụ thể có liên quan đến ứng dụng đang được phát triển; ví dụ, phần cứng và mô tả cơ sở dữ liệu. Yêu cầu phần cứng xác định cấu hình tối thiểu và tối ưu cho hệ thống. Yêu cầu cơ sở dữ liệu xác định tổ chức hợp lý của dữ liệu được hệ thống sử dụng và mối quan hệ giữa dữ liệu.
* Index ( mục lục): Một số chỉ mục cho tài liệu có thể được bao gồm. Cũng như chỉ mục chữ cái bình thường, có thể có chỉ mục sơ đồ, chỉ mục hàm, v.v.

## 5.32 Yêu cầu kỹ thuật

* Quá trình viết về yêu cầu của người dùng và hệ thống trong tài liệu yêu cầu.
* Các yêu cầu của người dùng phải dễ hiểu bởi người dùng cuối và khách hàng không có nền tảng kỹ thuật.
* Yêu cầu hệ thống là các yêu cầu chi tiết hơn và có thể bao gồm nhiều thông tin kỹ thuật hơn.
* Các yêu cầu có thể là một phần của hợp đồng phát triển hệ thống
  + Do đó, điều quan trọng là chúng càng hoàn chỉnh càng tốt.

## 5.33 Cách viết đặc tả yêu cầu hệ thống

* Natural language ( ngôn ngữ tự nhiên): Các yêu cầu được viết bằng các câu được đánh số bằng ngôn ngữ tự nhiên. Mỗi câu phải thể hiện một yêu cầu.
* Structured natural language ( ngôn ngữ tự nhiên có cấu trúc ): Các yêu cầu được viết bằng ngôn ngữ tự nhiên trên một mẫu hoặc mẫu chuẩn. Mỗi trường cung cấp thông tin về một khía cạnh của yêu cầu.
* Design description languages ( ngôn ngữ mô tả thiết kế ): Cách tiếp cận này sử dụng một ngôn ngữ như một ngôn ngữ lập trình, nhưng với nhiều tính năng trừu tượng hơn để xác định các yêu cầu bằng cách xác định một mô hình hoạt động của hệ thống. Cách tiếp cận này hiện nay hiếm khi được sử dụng mặc dù nó có thể hữu ích cho các đặc tả giao diện.
* Graphical notations ( ký hiệu đồ họa): Các mô hình đồ họa, được bổ sung bằng chú thích văn bản, được sử dụng để xác định các yêu cầu chức năng cho hệ thống; Trường hợp sử dụng UML và sơ đồ trình tự thường được sử dụng.
* Mathematical specifications ( thông số toán học ): Các ký hiệu này dựa trên các khái niệm toán học như các máy hoặc tập hợp hữu hạn. Mặc dù các thông số kỹ thuật rõ ràng này có thể làm giảm sự mơ hồ trong tài liệu yêu cầu, hầu hết khách hàng không hiểu một đặc điểm chính thức. Họ không thể kiểm tra xem nó đại diện cho những gì họ muốn và không muốn chấp nhận nó như một hợp đồng hệ thống.

## 5.34 Yêu cầu và thiết kế

* Về nguyên tắc, các yêu cầu nên nêu rõ hệ thống nên làm gì và thiết kế nên mô tả nó hoạt động như thế nào.
* Trong thực tế, yêu cầu và thiết kế là không thể tách rời
  + Một kiến ​​trúc hệ thống có thể được thiết kế để cấu trúc các yêu cầu;
  + Hệ thống có thể liên kết với các hệ thống khác tạo ra các yêu cầu thiết kế;
  + Việc sử dụng một kiến ​​trúc cụ thể để đáp ứng các yêu cầu phi chức năng có thể là một yêu cầu miền.
  + Điều này có thể là kết quả của một yêu cầu pháp lý.

## 5.35 Đặc điểm ngôn ngữ tự nhiên

* Yêu cầu được viết thành câu tự nhiên được bổ sung bởi biểu đồ và bảng.
* Được sử dụng để viết yêu cầu bởi vì nó là biểu cảm, trực quan và phổ quát. Điều này có nghĩa là các yêu cầu có thể được hiểu bởi người dùng và khách hàng.

## 5.36 Hướng dẫn viết yêu cầu

* Phát minh một định dạng chuẩn và sử dụng nó cho tất cả các yêu cầu.
* Sử dụng ngôn ngữ một cách nhất quán. Sử dụng cho các yêu cầu bắt buộc, nên cho các yêu cầu mong muốn.
* Sử dụng đánh dấu văn bản để xác định các phần chính của yêu cầu.
* Tránh sử dụng thuật ngữ máy tính.
* Bao gồm một lời giải thích (lý do) về lý do tại sao một yêu cầu là cần thiết.

## 5.37 Vấn đề với ngôn ngữ tự nhiên

Thiếu sự rõ ràng

* Độ chính xác rất khó mà không làm cho tài liệu khó đọc.

Yêu cầu nhầm lẫn

* Các yêu cầu chức năng và phi chức năng có xu hướng được trộn lẫn.

Yêu cầu hợp nhất

* Một số yêu cầu khác nhau có thể được thể hiện cùng nhau.

**Ví dụ về yêu cầu mẫu cho hệ thống phần mềm bơm insulin**

3.2 Hệ thống sẽ đo lượng đường trong máu và cung cấp insulin, nếu cần, cứ 10 phút một lần. (Những thay đổi về lượng đường trong máu tương đối chậm nên đo thường xuyên hơn là không cần thiết, ít đo lường thường xuyên hơn có thể dẫn đến mức đường không cần thiết cao.)

3.6 Hệ thống sẽ chạy một thói quen tự kiểm tra mỗi phút với các điều kiện cần kiểm tra và các hành động liên quan được định nghĩa trong Bảng 1. (Một thói quen tự kiểm tra có thể phát hiện ra các vấn đề về phần cứng và phần mềm và thông báo cho người dùng biết không thể.)

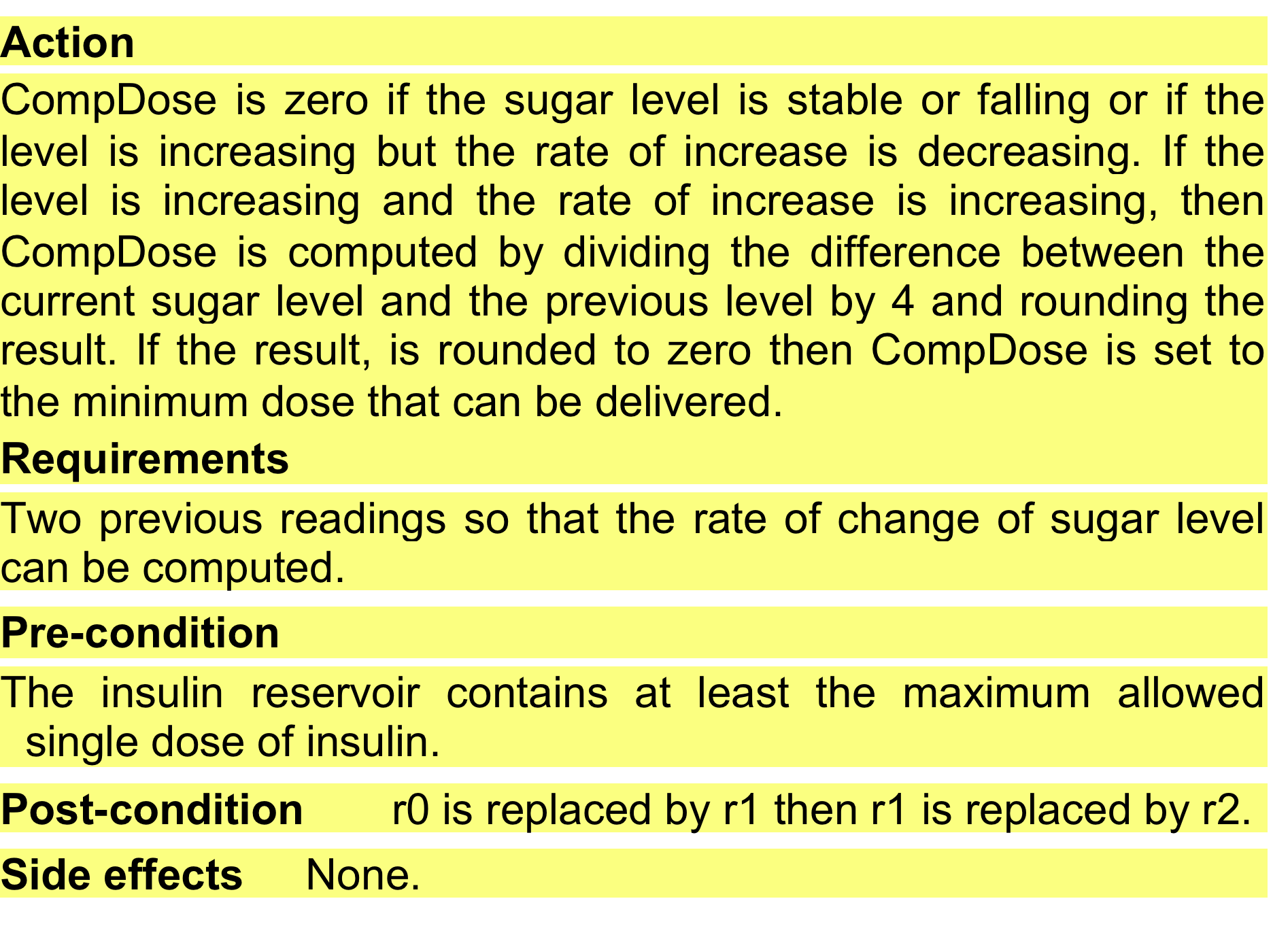
## 5.38 Thông số kỹ thuật có cấu trúc

* Một cách tiếp cận để viết các yêu cầu mà sự tự do của người viết yêu cầu bị hạn chế và các yêu cầu được viết theo cách tiêu chuẩn.
* Điều này phù hợp với một số loại yêu cầu, ví dụ: các yêu cầu cho hệ thống điều khiển nhúng nhưng đôi khi quá cứng nhắc để viết các yêu cầu hệ thống kinh doanh.

## 5.39 Thông số dựa trên biểu mẫu

* Định nghĩa hàm hoặc thực thể.
* Mô tả đầu vào và vị trí của chúng.
* Mô tả các kết quả đầu ra và nơi chúng đến.
* Thông tin về thông tin cần thiết cho tính toán và các thực thể khác được sử dụng.
* Mô tả hành động cần thực hiện.
* Điều kiện trước và sau (nếu thích hợp).
* Các tác dụng phụ (nếu có) của chức năng.

**Một đặc tả cấu trúc của một yêu cầu cho một máy bơm insulin**



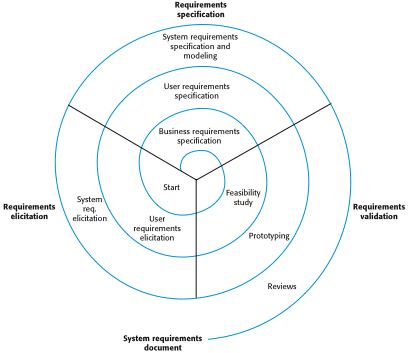
**Đặc điểm bảng**

* Được sử dụng để bổ sung ngôn ngữ tự nhiên.
* Đặc biệt hữu ích khi bạn phải xác định một số các khóa học hành động thay thế có thể có.
* Ví dụ, các hệ thống bơm insulin căn cứ tính toán của nó về tốc độ thay đổi lượng đường trong máu và đặc điểm kỹ thuật dạng bảng giải thích cách tính yêu cầu insulin cho các tình huống khác nhau.

## 5.40 Yêu cầu quy trình kỹ thuật

* Các quy trình được sử dụng cho RE thay đổi rất nhiều tùy thuộc vào miền ứng dụng, những người liên quan và tổ chức phát triển các yêu cầu.
* Tuy nhiên, có một số hoạt động chung chung cho tất cả các quy trình
  + Yêu cầu gợi;
  + Phân tích yêu cầu;
  + Yêu cầu xác nhận;
  + Quản lý yêu cầu.
* Trong thực tế, RE là một hoạt động lặp lại trong đó các quá trình này được xen kẽ.

## 5.41 Một cái nhìn xoắn ốc của quá trình kỹ thuật yêu cầu

****

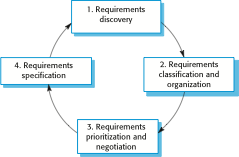
## 5.42 Yêu cầu gợi ý và phân tích

* Đôi khi được gọi là yêu cầu gợi ý hoặc khám phá yêu cầu.
* Liên quan đến nhân viên kỹ thuật làm việc với khách hàng để tìm hiểu về miền ứng dụng, các dịch vụ mà hệ thống sẽ cung cấp và các hạn chế hoạt động của hệ thống.
* Có thể liên quan đến người dùng cuối, người quản lý, kỹ sư tham gia bảo trì, chuyên gia tên miền, công đoàn, v.v. Đây được gọi là bên liên quan

## 5.43 Vấn đề phân tích yêu cầu

* Các bên liên quan không biết họ thực sự muốn gì.
* Các bên liên quan thể hiện các yêu cầu theo cách riêng của họ.
* Các bên liên quan khác nhau có thể có các yêu cầu xung đột.
* Các yếu tố tổ chức và chính trị có thể ảnh hưởng đến các yêu cầu hệ thống.
* Các yêu cầu thay đổi trong quá trình phân tích. Các bên liên quan mới có thể xuất hiện và môi trường kinh doanh có thể thay đổi.
* Kỹ sư phần mềm làm việc với một loạt các bên liên quan hệ thống để tìm hiểu về miền ứng dụng, các dịch vụ mà hệ thống sẽ cung cấp, hiệu suất hệ thống được yêu cầu, các ràng buộc phần cứng, các hệ thống khác, v.v.
* Các giai đoạn bao gồm:
  + Yêu cầu khám phá,
  + Yêu cầu phân loại và tổ chức,
  + Yêu cầu ưu tiên và thương lượng,
  + Yêu cầu kỹ thuật.

## 5.44 Quy trình phân tích và yêu cầu yêu cầu

****

## 5.45 Quy trình hoạt động

* Yêu cầu khám phá
  + Tương tác với các bên liên quan để khám phá các yêu cầu của họ. Yêu cầu về miền cũng được phát hiện ở giai đoạn này.
* Phân loại và tổ chức yêu cầu
  + Các yêu cầu liên quan đến nhóm và tổ chức chúng thành các cụm kết hợp.
* Ưu tiên và thương lượng
  + Ưu tiên các yêu cầu và giải quyết xung đột các yêu cầu.
* Yêu cầu kỹ thuật
  + Yêu cầu được ghi lại và nhập vào vòng tiếp theo của hình xoắn ốc.

## 5.46 Các vấn đề về yêu cầu

* Các bên liên quan không biết họ thực sự muốn gì.
* Các bên liên quan thể hiện các yêu cầu theo cách riêng của họ.
* Các bên liên quan khác nhau có thể có các yêu cầu xung đột.
* Các yếu tố tổ chức và chính trị có thể ảnh hưởng đến các yêu cầu hệ thống.
* Các yêu cầu thay đổi trong quá trình phân tích. Các bên liên quan mới có thể nổi lên và thay đổi môi trường kinh doanh.

**Key points ( chốt cmn bài )**

* Tài liệu yêu cầu phần mềm là một tuyên bố đồng thuận về các yêu cầu hệ thống. Nó nên được tổ chức để cả khách hàng hệ thống và các nhà phát triển phần mềm có thể sử dụng nó.
* Quy trình kỹ thuật yêu cầu là một quá trình lặp đi lặp lại bao gồm yêu cầu gợi ý, đặc điểm kỹ thuật và xác thực.
* Yêu cầu gợi ý và phân tích là một quá trình lặp lại có thể được biểu diễn như một vòng xoáy của các hoạt động - phát hiện yêu cầu, phân loại yêu cầu và tổ chức, thương lượng yêu cầu và tài liệu yêu cầu.

# Chapter 6: Requirement Engineer Process

Mục tiêu:

* Để mô tả các yêu cầu chính về các hoạt động kỹ thuật và mối quan hệ của họ.
* Giới thiệu các kỹ thuật về yêu cầu gợi ý và phân tích.
* Để mô tả yêu cầu có tính xác thực và vai trò của các đánh giá yêu cầu.
* Để thảo luận về vai trò của quản lý yêu cầu để hỗ trợ các quy trình yêu cầu kỹ thuật khác.

Các chủ đề được đề cập:

* Nghiên cứu khả thi.
* Yêu cầu gợi ý và phân tích.
* Xác nhận yêu cầu.
* Quản lý yêu cầu.

## 6.1 Yêu cầu quy trình kỹ thuật:

* Các quy trình được sử dụng cho RE thay đổi rất nhiều tùy thuộc vào miền ứng dụng, những người liên quan và tổ chức phát triển các yêu cầu.
* Tuy nhiên, có một số hoạt động chung chung cho tất cả các quy trình.
* Yêu cầu gợi.
* Phân tích yêu cầu.
* Yêu cầu xác nhận.
* Quản lý yêu cầu.

7.1 RE-process.eps                                             001057BBMacintosh HD                   B8AA5F2E:

Yêu cầu kỹ thuật:

7.2.eps                                                        001BF29EMacintosh HD                   B8AA5F2E:

### 6.1.1 Nghiên cứu khả thi

* Một nghiên cứu khả thi quyết định liệu hệ thống đề xuất có đáng giá hay không.
* Một nghiên cứu ngắn để kiểm tra:
* Nếu hệ thống đóng góp vào mục tiêu tổ chức;
* Nếu hệ thống có thể được thiết kế bằng công nghệ hiện tại và trong phạm vi ngân sách;
* Nếu hệ thống có thể được tích hợp với các hệ thống khác được sử dụng.

Triển khai nghiên cứu khả thi:

* Dựa trên đánh giá thông tin (những gì được yêu cầu), thu thập thông tin và viết báo cáo.
* Câu hỏi dành cho những người trong tổ chức
* Điều gì sẽ xảy ra nếu hệ thống không được triển khai?
* Các vấn đề về quy trình hiện tại là gì?
* Hệ thống đề xuất sẽ giúp như thế nào?
* Vấn đề hội nhập sẽ là gì?
* Công nghệ mới có cần thiết không? Kỹ năng gì?
* Các hệ thống được đề xuất phải hỗ trợ những cơ sở nào?

Khai thác và phân tích:

* Đôi khi được gọi là yêu cầu gợi ý hoặc tìm hiểu yêu cầu.
* Liên quan đến nhân viên kỹ thuật làm việc với khách hàng để tìm hiểu về miền ứng dụng, các dịch vụ mà hệ thống sẽ cung cấp và các hạn chế hoạt động của hệ thống.
* Có thể liên quan đến người dùng cuối, người quản lý, kỹ sư tham gia vào việc bảo trì, chuyên gia về các domain, công đoàn, v.v. Đây được gọi là các bên liên quan.

Các vấn đề về phân tích yêu cầu:

* Các bên liên quan không biết họ thực sự muốn gì.
* Các bên liên quan thể hiện các yêu cầu theo cách riêng của họ.
* Các bên liên quan khác nhau có thể có các yêu cầu xung đột.
* Các yếu tố tổ chức và chính trị có thể ảnh hưởng đến các yêu cầu hệ thống.
* Các yêu cầu thay đổi trong quá trình phân tích. Các bên liên quan mới có thể phát triền tiếng tăm và thay đổi môi trường kinh doanh.

Yêu cầu xoắn ốc:

7.3*.eps                                                       001BEA14Macintosh HD                   B8AA5F2E:

Quy trình hoạt động:

* Yêu cầu về tìm hiểu
  + Tương tác với các bên liên quan để khám phá các yêu cầu của họ. Yêu cầu về miền cũng được phát hiện ở giai đoạn này.
* Phân loại và tổ chức yêu cầu
  + Các yêu cầu liên quan đến nhóm và tổ chức chúng thành các cụm kết hợp.
* Ưu tiên và thương lượng
  + Ưu tiên các yêu cầu và giải quyết xung đột các yêu cầu.
* Tài liệu yêu cầu
  + Yêu cầu được ghi lại và nhập vào vòng tiếp theo của hình xoắn ốc.

Yêu cầu về tìm hiểu

* Quá trình thu thập thông tin về các hệ thống được đề xuất và hiện có và phân biệt các yêu cầu của người dùng và hệ thống từ thông tin này.
* Nguồn thông tin bao gồm tài liệu, các bên liên quan hệ thống và các thông số kỹ thuật của các hệ thống tương tự.\

Các bên liên quan của ATM

* Khách hàng của ngân hàng
* Đại diện của các ngân hàng khác
* Người quản lý ngân hàng
* Nhân viên truy cập
* Quản trị cơ sở dữ liệu
* Quản lý an ninh
* Bộ phận tiếp thị
* Kỹ sư bảo trì phần cứng và phần mềm
* Ngân hàng quản lý

## 6.2 Quan điểm:

* Quan điểm là một cách để xây dựng cấu trúc của các yêu cầu để thể hiện quan điểm của các bên liên quan khác nhau. Các bên liên quan có thể được phân loại theo các quan điểm khác nhau.
* Phân tích đa góc độ này rất quan trọng vì không có cách nào đơn lẻ để phân tích các yêu cầu hệ thống.

Các loại quan điểm

* Quan điểm của người tuong tác
  + Con người hoặc các hệ thống khác tương tác trực tiếp với hệ thống. Trong máy ATM, cơ sở dữ liệu của khách hàng và tài khoản là các VP tương tác.
* Quan điểm gián tiếp
  + Các bên liên quan không tự sử dụng hệ thống nhưng lại ảnh hưởng đến các yêu cầu. Trong máy ATM, nhân viên quản lý và an ninh là quan điểm gián tiếp.
* Quan điểm tên miền
  + Các đặc điểm và hạn chế về miền ảnh hưởng đến các yêu cầu. Trong một máy ATM, một ví dụ sẽ là tiêu chuẩn cho việc liên lạc giữa các ngân hàng.

Nhận dạng quan điểm

* Xác định các quan điểm bằng cách sử dụng
  + Nhà cung cấp và người nhận dịch vụ hệ thống;
  + Các hệ thống tương tác trực tiếp với hệ thống đang được chỉ định;
  + Các quy định và tiêu chuẩn;
  + Nguồn yêu cầu kinh doanh và phi chức năng.
  + Các kỹ sư phải phát triển và duy trì hệ thống;
  + Tiếp thị và quan điểm kinh doanh khác.

## 6.3 LIBSYS viewpoint hierarchy

7.4.eps                                                        001BF29EMacintosh HD                   B8AA5F2E:

### 6.3.1 Họp bàn ý kiến

* Trong cuộc họp chính thức hoặc không chính thức, nhóm RE đặt câu hỏi cho các bên liên quan về hệ thống mà họ sử dụng và hệ thống được phát triển.
* Có hai loại:
  + Các cuộc họp kín, câu hỏi đã biết trước câu trả lời.
  + Mở các cuộc họp với các câu hỏi không được biết trước.

Các cuộc họp bàn ý kiến thực tế:

* Thông thường, kết hợp phỏng vấn khép kín và mở.
* Các cuộc họp rất tốt để có được sự hiểu biết tổng thể về những gì các bên liên quan làm và cách họ có thể tương tác với hệ thống.
* Các cuộc phỏng vấn không tốt cho việc hiểu các yêu cầu về miền
* Cuộc họp khonong tốt cho việc yêu cầu tên miền cụ thể;
* Một số kiến thức về miền quá quen thuộc đến nỗi mọi người thấy khó để nói rõ hoặc nghĩ rằng nó không đáng nói.

Kịch bản(Scenarios):

* Các kịch bản là những ví dụ thực tế về cách một hệ thống có thể được sử dụng.
* Bao gồm
  + Mô tả tình huống bắt đầu;
  + Một mô tả về trường hợp chuẩn.
  + Một mô tả về các trường hợp sai có thể xảy ra.
  + Thông tin về các hoạt động đồng thời khác.
  + Mô tả trạng thái khi kịch bản kết thúc.

LIBSYS scenario :

* Giả định ban đầu: Người dùng đã đăng nhập vào hệ thống LIBSYS và đã đặt journal chứa bản sao của bài viết.
* Bình thường: Người dùng chọn bài viết cần sao chép. Sau đó, người đó được hệ thống nhắc nhở cung cấp thông tin người đăng ký cho journal hoặc để cho biết họ sẽ thanh toán như thế nào. Phương thức thanh toán thay thế bằng thẻ tín dụng hoặc bằng cách trích dẫn số tài khoản tổ chức.
* Sau đó, người dùng được yêu cầu điền vào một biểu mẫu bản quyền để duy trì chi tiết về giao dịch và sau đó họ gửi nó tới hệ thống LIBSYS.
* Mẫu bản quyền được chọn và, nếu OK, phiên bản PDF của bài viết được tải xuống khu vực làm việc LIBSYS trên máy tính của người dùng và người dùng được thông báo rằng nó có sẵn. Người dùng được yêu cầu chọn một máy in và một bản sao của bài báo được in. Nếu bài viết đã bị gắn cờ là 'chỉ in', bài viết đó sẽ bị xóa khỏi hệ thống của người dùng khi người dùng đã xác nhận rằng quá trình in hoàn tất.
* Điều gì có thể xảy ra: Người dùng có thể không điền đúng mẫu. Trong trường hợp này, biểu mẫu sẽ được trình bày lại cho người dùng để sửa chữa. Nếu biểu mẫu gửi lại vẫn không chính xác thì yêu cầu của người dùng bị từ chối.
* Thanh toán có thể bị hệ thống từ chối. Yêu cầu của người dùng bị từ chối.
* Tải xuống bài viết có thể không thành công. Thử lại cho đến khi thành công hoặc người dùng chấm dứt việc tải.
* Không thể in bài viết. Nếu bài viết không được gắn cờ là 'chỉ in' thì bài viết đó được giữ trong không gian làm việc LIBSYS. Nếu không, bài viết sẽ bị xóa và tài khoản của người dùng được ghi có với chi phí của bài viết.
* Các hoạt động khác: Tải xuống đồng thời các bài viết khác.
* Trạng thái hệ thống khi hoàn thành: Người dùng đã đăng nhập. Bài viết đã tải xuống đã bị xóa khỏi không gian làm việc LIBSYS nếu nó đã được gắn cờ là chỉ in.

Use case:

* Các use-case là một kỹ thuật dựa trên scenario trong UML để xác định các tác nhân trong một tương tác và mô tả tương tác đó.
* Một tập hợp các use-case nên mô tả tất cả các tương tác có thể có với hệ thống.
* Biểu đồ trình tự(SD) có thể được sử dụng để thêm chi tiết vào các ca sử dụng bằng cách hiển thị chuỗi xử lý sự kiện trong hệ thống.

## 6.4 Article printing use-case:

7.6.eps                                                        001BF29EMacintosh HD                   B8AA5F2E:

## 6.5 LIBSYS use cases

7.7 LIBSYSUseCases(6.12)**.eps                                 001BEA14Macintosh HD                   B8AA5F2E:

## 6.6 Article Printing

7.8*.eps                                                       001BEA14Macintosh HD                   B8AA5F2E:

## 6.7 Print article sequences:


7.8*IS.eps                                                     001BEA14Macintosh HD                   B8AA5F2E:

## 6.8 Các yếu tố xã hội và tổ chức

* Các hệ thống phần mềm được sử dụng trong bối cảnh xã hội và tổ chức. Điều này có thể ảnh hưởng hoặc thậm chí thống trị các yêu cầu hệ thống.
* Các yếu tố xã hội và tổ chức không phải là một quan điểm duy nhất mà là những ảnh hưởng trên tất cả các quan điểm.
* Các nhà phân tích giỏi phải nhạy cảm với những yếu tố này nhưng hiện tại không có cách nào có hệ thống để giải quyết phân tích của họ.

Dân tộc học(Etnography):

* Một nhà khoa học xã hội dành một thời gian đáng kể quan sát và phân tích cách mọi người thực sự làm việc.
* Mọi người không phải giải thích hoặc nêu rõ công việc của họ.
* Các yếu tố xã hội và tổ chức có tầm quan trọng có thể được quan sát.
* Các nghiên cứu dân tộc học đã chỉ ra rằng công việc thường phong phú hơn và phức tạp hơn so với các mô hình hệ thống đơn giản.

Dân tộc học tập trung(Etnography focused)

* Phát triển trong một dự án nghiên cứu quá trình kiểm soát không lưu
* Kết hợp dân tộc học với prototyping
* Kết quả phát triển thử nghiệm trong các câu hỏi chưa được trả lời tập trung vào phân tích dân tộc học.
* Vấn đề với dân tộc học là nó nghiên cứu các thực hành hiện tại có thể có một số cơ sở lịch sử không còn phù hợp nữa.

## 6.9 Ethnography and Prototyping

7.9.eps                                                        001BF29EMacintosh HD                   B8AA5F2E:

Phạm vi dân tộc học:

* Yêu cầu bắt nguồn từ cách mọi người làm việc nhiều hơn là cách tôi đề nghị họ phải làm.
* Yêu cầu bắt nguồn từ sự hợp tác và nhận thức về các hoạt động của người khác.

## 6.10 Xác nhận yêu cầu

* Quan tâm đến việc chứng minh rằng các yêu cầu xác định hệ thống mà khách hàng thực sự muốn.
* Yêu cầu chi phí lỗi cao nên việc xác nhận là rất quan trọng
* Việc sửa lỗi yêu cầu sau khi gửi có thể tốn tới 100 lần chi phí sửa lỗi ban đầu.

Yêu cầu kiểm tra

* Hiệu lực. Hệ thống có cung cấp các chức năng hỗ trợ tốt nhất nhu cầu của khách hàng không?
* Tính nhất quán. Có bất kỳ xung đột yêu cầu nào không?
* Đầy đủ. Có bao gồm tất cả các chức năng theo yêu cầu của khách hàng không?
* Chủ nghĩa hiện thực. Các yêu cầu có thể được thực hiện dựa trên ngân sách và công nghệ có sẵn không
* Tính xác thực. Các yêu cầu có thể được kiểm tra không?

Yêu cầu kỹ thuật xác nhận

* Yêu cầu đánh giá
* Phân tích thủ công có hệ thống các yêu cầu.
* Tạo mẫu
* Sử dụng một mô hình thực thi của hệ thống để kiểm tra các yêu cầu.
* Tạo trường hợp thử nghiệm
* Phát triển các bài kiểm tra cho các yêu cầu để kiểm tra khả năng.

Yêu cầu đánh giá

* Các đánh giá thông thường phải được tổ chức trong khi định nghĩa về yêu cầu đang được xây dựng.
* Cả nhân viên khách hàng và nhà thầu nên tham gia vào các đánh giá.
* Bài đánh giá có thể chính thức (với tài liệu hoàn chỉnh) hoặc không chính thức. Giao tiếp tốt giữa các nhà phát triển, khách hàng và người dùng có thể giải quyết vấn đề ở giai đoạn đầu.

Kiểm tra đánh giá

* Tính xác thực. Yêu cầu có thực nghiệm được không?
* Tính toàn diện. Yêu cầu có được hiểu đúng không?
* Truy xuất nguồn gốc. Nguồn gốc của yêu cầu có được nêu rõ không?
* Khả năng thích ứng. Yêu cầu có thể thay đổi mà không ảnh hưởng lớn đến các yêu cầu khác không?

Quản lý yêu cầu

* Quản lý yêu cầu là quá trình quản lý các yêu cầu thay đổi trong quá trình yêu cầu kỹ thuật và phát triển hệ thống.
* Yêu cầu chắc chắn không đầy đủ và không nhất quán
* Các yêu cầu mới xuất hiện trong quá trình khi nhu cầu kinh doanh thay đổi và sự hiểu biết tốt hơn về hệ thống được phát triển;
* Các quan điểm khác nhau có các yêu cầu khác nhau và những điều này thường mâu thuẫn.

Yêu cầu thay đổi

* Ưu tiên các yêu cầu từ các quan điểm khác nhau thay đổi trong quá trình phát triển.
* Khách hàng hệ thống có thể chỉ định các yêu cầu từ quan điểm kinh doanh xung đột với các yêu cầu của người dùng cuối.
* Môi trường kinh doanh và kỹ thuật của hệ thống thay đổi trong quá trình phát triển của nó.

Yêu cầu tiến hóa(requirements evolution)

7.10.eps                                                       001BF29EMacintosh HD                   B8AA5F2E:

Yêu cầu lâu dài và tạm thời:

* Yêu cầu về lâu dài. Yêu cầu ổn định bắt nguồn từ hoạt động cốt lõi của tổ chức khách hàng. Ví dụ. một bệnh viện sẽ luôn luôn có bác sĩ, y tá, vv Có thể bắt nguồn từ các domain model
* Yêu cầu tạm thời. Yêu cầu thay đổi trong quá trình phát triển hoặc khi hệ thống đang được sử dụng. Trong bệnh viện, các yêu cầu bắt nguồn từ chính sách chăm sóc sức khỏe

## 6.11 Phân loại yêu cầu

* Loại yêu cầu
* Các yêu cầu có thể thay: đổi Các yêu cầu thay đổi do các thay đổi đối với môi trường mà tổ chức đang hoạt động. Ví dụ, trong các hệ thống bệnh viện, kinh phí chăm sóc bệnh nhân có thể thay đổi và do đó yêu cầu thu thập thông tin điều trị khác nhau.
* Yêu cầu khẩn cấp: Các yêu cầu xuất hiện khi sự hiểu biết của khách hàng về hệ thống phát triển trong quá trình phát triển hệ thống. Quy trình thiết kế có thể tiết lộ các yêu cầu mới nổi lên.
* Yêu cầu hậu quả: Các yêu cầu bắt nguồn từ hệ thống máy tính. Giới thiệu hệ thống máy tính có thể thay đổi quy trình của tổ chức và mở ra các cách làm việc mới, tạo ra các yêu cầu hệ thống mới
* Yêu cầu tương thích: Các yêu cầu phụ thuộc vào các hệ thống hoặc quy trình nghiệp vụ cụ thể trong một tổ chức. Khi những thay đổi này, các yêu cầu về tính tương thích trên hệ thống được giao hoặc được giao cũng có thể phải phát triển.

## 6.12 Yêu cầu lập kế hoạch quản lý

* Trong quá trình yêu cầu kỹ thuật, bạn phải lập kế hoạch:
  + Yêu cầu nhận dạng
  + Các yêu cầu được xác định riêng như thế nào;
  + Quy trình quản lý thay đổi
  + Quá trình tiếp theo khi phân tích yêu cầu thay đổi;
  + Chính sách truy xuất nguồn gốc
  + Lượng thông tin về các mối quan hệ yêu cầu được duy trì;
  + Hỗ trợ công cụ CASE
  + Công cụ hỗ trợ cần thiết để giúp quản lý các yêu cầu thay đổi;

## 6.13 Truy xuất nguồn gốc

* Truy xuất nguồn gốc có liên quan với các mối quan hệ giữa các yêu cầu, nguồn của chúng và thiết kế hệ thống
* Truy xuất nguồn gốc
* Liên kết từ các yêu cầu đến các bên liên quan đã đề xuất các yêu cầu này;
* Yêu cầu truy xuất nguồn gốc
* Liên kết giữa các yêu cầu phụ thuộc;
* Thiết kế truy xuất nguồn gốc
* Liên kết từ các yêu cầu đến thiết kế;

## 6.14 Ma trận truy xuất nguồn gốc(A traceability matrix)

1



## 6.15 CASE Tool Support

* Yêu cầu lưu trữ
* Yêu cầu phải được quản lý trong kho lưu trữ dữ liệu được quản lý, an toàn.
* Thay đổi cách quản lý
* Quá trình quản lý thay đổi là một quy trình quy trình làm việc có các giai đoạn có thể được xác định và luồng thông tin giữa các giai đoạn này được tự động hóa một phần.
* Quản lý truy xuất nguồn gốc
* Tự động thu hồi các liên kết giữa các yêu cầu.

Yêu cầu quản lý thay đổi

* Nên áp dụng cho tất cả các thay đổi được đề xuất cho các yêu cầu.
* Các giai đoạn chính
* Phân tích vấn đề. Thảo luận về vấn đề yêu cầu và đề xuất thay đổi;
* Thay đổi phân tích và chi phí. Đánh giá tác động của thay đổi đối với các yêu cầu khác;
* Thay đổi triển khai. Sửa đổi tài liệu yêu cầu và các tài liệu khác để phản ánh thay đổi.

Thay đổi cách quản lý

7.13.eps                                                       001BF29EMacintosh HD                   B8AA5F2E:

Những điểm chính

* Quy trình kỹ thuật yêu cầu bao gồm nghiên cứu khả thi, yêu cầu và phân tích yêu cầu, yêu cầu đặc điểm kỹ thuật và quản lý yêu cầu.
* Yêu cầu gợi ý và phân tích là lặp đi lặp lại liên quan đến hiểu biết về miền, thu thập yêu cầu, phân loại, cấu trúc, ưu tiên và xác nhận.
* Hệ thống có nhiều bên liên quan với các yêu cầu khác nhau.
* Các yếu tố xã hội và tổ chức ảnh hưởng đến các yêu cầu của hệ thống.
* Yêu cầu xác nhận là có liên quan với kiểm tra tính hợp lệ, nhất quán, đầy đủ, hiện thực và kiểm chứng.
* Thay đổi kinh doanh chắc chắn dẫn đến yêu cầu thay đổi.
* Quản lý yêu cầu bao gồm quản lý quy hoạch và thay đổi

# Chapter 7 : Design & Implement

## 7.1 Thiết kế và thực hiện

Thiết kế và thực hiện phần mềm là giai đoạn trong quy trình kỹ thuật phần mềm mà tại đó một hệ thống phần mềm thực thi được phát triển.

Các hoạt động thiết kế và triển khai phần mềm luôn được liên tục.

* Thiết kế phần mềm là một hoạt động sáng tạo trong đó bạn xác định các thành phần phần mềm và các mối quan hệ của chúng, dựa trên các yêu cầu của khách hàng.
* Thực hiện là quá trình thực hiện thiết kế như một chương trình

## 7.2 Xây dựng hoặc mua

Trong một loạt các lĩnh vực, bây giờ có thể mua các hệ thống off-the-shelf (COTS) có thể được điều chỉnh và phù hợp với yêu cầu của người dùng.

* Ví dụ, nếu bạn muốn thực hiện một hệ thống hồ sơ y tế, bạn có thể mua một gói đã được sử dụng trong các bệnh viện. Nó có thể rẻ hơn và nhanh hơn để sử dụng phương pháp này hơn là phát triển một hệ thống bằng ngôn ngữ lập trình thông thường.

Khi bạn phát triển một ứng dụng theo cách này, quá trình thiết kế trở nên quan tâm đến cách sử dụng các tính năng cấu hình của hệ thống đó để cung cấp các yêu cầu hệ thống.

## 7.3 Hướng đối tượng

Các quá trình thiết kế hướng đối tượng có cấu trúc liên quan đến việc phát triển một số mô hình hệ thống khác nhau.

Họ đòi hỏi rất nhiều nỗ lực để phát triển và bảo trì các mô hình này, và đối với các hệ thống nhỏ, điều này có thể không hiệu quả về chi phí.

Tuy nhiên, đối với các hệ thống lớn được phát triển bởi các mô hình thiết kế nhóm khác nhau là một cơ chế truyền thông quan trọng.

## 7.4 Quy trình giai đoạn

Có nhiều quy trình thiết kế hướng đối tượng khác nhau phụ thuộc vào tổ chức sử dụng quy trình.

Các hoạt động phổ biến trong các quy trình này bao gồm:

* Xác định bối cảnh và phương thức sử dụng hệ thống;
* Thiết kế kiến trúc hệ thống;
* Xác định các đối tượng hệ thống chính;
* Phát triển các mô hình thiết kế;
* Chỉ định giao diện đối tượng.

Quy trình được minh họa ở đây sử dụng thiết kế cho trạm thời tiết hoang dã.

## 7.5 Ngữ cảnh hệ thống và tương tác

Hiểu được mối quan hệ giữa phần mềm đang được thiết kế và môi trường bên ngoài của nó là điều cần thiết để quyết định cách cung cấp chức năng hệ thống được yêu cầu và cách cấu trúc hệ thống để giao tiếp với môi trường của nó.

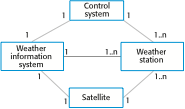
Hiểu biết về bối cảnh cũng cho phép bạn thiết lập các ranh giới của hệ thống. Đặt ranh giới hệ thống giúp bạn quyết định những tính năng nào được triển khai trong hệ thống đang được thiết kế và những tính năng nào nằm trong các hệ thống liên quan khác.

**Mô hình ngữ cảnh và tương tác**

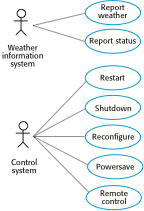
Mô hình ngữ cảnh hệ thống là một mô hình kết cấu thể hiện các hệ thống khác trong môi trường của hệ thống đang được phát triển.

Mô hình tương tác là một mô hình động cho thấy cách hệ thống tương tác với môi trường của nó khi nó được sử dụng.

## 7.6 System Context For The Weather Station



## 7.7 Weather station use cases



## 7.8 Use Case Description—Report Weather

**Use case:** Báo cáo thời tiết

**Actors:** Hệ thống thông tin thời tiết, trạm thời tiết

**Description:** Trạm thời tiết gửi bản tóm tắt dữ liệu thời tiết đã được thu thập từ các công cụ trong giai đoạn thu thập tới hệ thống thông tin thời tiết. Dữ liệu được gửi là nhiệt độ mặt đất và không khí tối đa, tối thiểu và trung bình; áp lực không khí tối đa, tối thiểu và trung bình; tốc độ gió tối đa, tối thiểu và trung bình; tổng lượng mưa; và hướng gió được lấy mẫu ở khoảng thời gian năm phút.

**Stimulus ( kích thích kinh tế )**: Hệ thống thông tin thời tiết thiết lập một liên kết truyền thông vệ tinh với trạm thời tiết và yêu cầu truyền dữ liệu.

**Response:** Dữ liệu tóm tắt được gửi đến hệ thống thông tin thời tiết.

**Comments:** Các trạm thời tiết thường được yêu cầu báo cáo một lần mỗi giờ nhưng tần suất này có thể khác nhau từ trạm này sang trạm khác và có thể được sửa đổi trong tương lai.

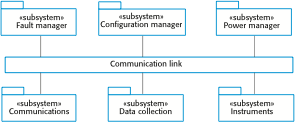
## 7.9 Thiết kế kiến trúc

Khi các tương tác giữa hệ thống và môi trường của hệ thống đã được hiểu, bạn sử dụng thông tin này để thiết kế kiến trúc hệ thống.

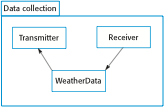
Bạn xác định các thành phần chính tạo nên hệ thống và tương tác của chúng, và sau đó có thể tổ chức các thành phần bằng cách sử dụng một mẫu kiến trúc như mô hình lớp hoặc máy khách-máy chủ.

Trạm thời tiết bao gồm các hệ thống con độc lập giao tiếp bằng cách truyền phát thông điệp trên cơ sở hạ tầng chung

## 7.10 Kiến trúc cấp cao của Trạm thời tiết



## 7.11 Kiến trúc của hệ thống thu thập dữ liệu



**Nhận dạng lớp đối tượng**

Việc xác định các lớp đối tượng thường là một phần khó khăn của thiết kế hướng đối tượng.

Không có 'công thức ma thuật' để nhận diện đối tượng. Nó dựa trên kỹ năng, kinh nghiệm và kiến thức về miền của các nhà thiết kế hệ thống.

Nhận dạng đối tượng là một quá trình lặp lại. Bạn có thể không nhận được nó ngay lần đầu tiên.

## 7.12 Phương pháp tiếp cận để xác định

Sử dụng phương pháp ngữ pháp dựa trên mô tả ngôn ngữ tự nhiên của hệ thống (được sử dụng trong phương pháp Hood OOD).

Căn cứ việc xác định những thứ hữu hình trong miền ứng dụng.

Sử dụng cách tiếp cận hành vi và xác định các đối tượng dựa trên những gì tham gia vào hành vi nào.

Sử dụng phân tích dựa trên kịch bản. Các đối tượng, thuộc tính và phương pháp trong từng kịch bản(scenario) được xác định.

**Trạm thời tiết**

Trạm thời tiết là một gói các công cụ kiểm soát phần mềm thu thập dữ liệu, thực hiện một số xử lý dữ liệu và truyền dữ liệu này để xử lý tiếp. Các dụng cụ bao gồm nhiệt kế không khí và mặt đất, máy đo gió, cánh gió, thước đo và thước đo mưa. Dữ liệu được thu thập định kỳ.

Khi một lệnh được đưa ra để truyền dữ liệu thời tiết, trạm thời tiết sẽ xử lý và tóm tắt dữ liệu đã thu thập. Dữ liệu tóm tắt được truyền tới máy tính ánh xạ khi nhận được yêu cầu.

## 7.13 Các lớp đối tượng trạm thời tiết

Việc xác định lớp đối tượng trong hệ thống trạm thời tiết có thể dựa trên phần cứng và dữ liệu hữu hình trong hệ thống:

Nhiệt kế đất, Máy đo gió, Barometer

* Các đối tượng miền ứng dụng là các đối tượng 'phần cứng' có liên quan đến các công cụ trong hệ thống.

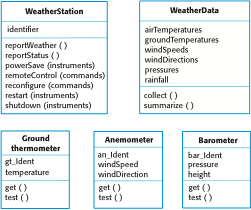
Trạm thời tiết

* Giao diện cơ bản của trạm thời tiết với môi trường của nó. Do đó, nó phản ánh các tương tác được xác định trong mô hình ca sử dụng.

Dữ liệu thời tiết

* Đóng gói dữ liệu tóm tắt từ các công cụ.

## 7.14 Các lớp đối tượng trạm thời tiết



## 7.15 Mô hình thiết kế

Các mô hình thiết kế hiển thị các đối tượng và các lớp đối tượng và mối quan hệ giữa các thực thể này.

Các mô hình tĩnh mô tả cấu trúc tĩnh của hệ thống về các lớp đối tượng và các mối quan hệ.

Mô hình động mô tả tương tác động giữa các đối tượng.

**Ví dụ về các mẫu thiết kế**

Các mô hình hệ thống con hiển thị các nhóm đối tượng logic vào các hệ thống phụ mạch lạc.

Các mô hình trình tự hiển thị chuỗi các tương tác đối tượng.

Các mô hình máy trạng thái hiển thị cách các đối tượng riêng lẻ thay đổi trạng thái của chúng để đáp ứng với các sự kiện.

Các mô hình khác bao gồm các mô hình ca sử dụng, mô hình tổng hợp, mô hình khái quát, v.v.

## 7.16 Mô hình hệ thống con

Cho thấy cách thiết kế được tổ chức thành các nhóm đối tượng có liên quan logic.

Trong UML, chúng được hiển thị bằng cách sử dụng các gói - một cấu trúc đóng gói. Đây là một mô hình logic. Việc tổ chức thực tế các đối tượng trong hệ thống có thể khác nhau.

**Mô hình hệ thống con(Subsystem Model)**

Cho thấy cách thiết kế được tổ chức thành các nhóm đối tượng có liên quan logic.

Trong UML, chúng được hiển thị bằng cách sử dụng các gói - một cấu trúc đóng gói. Đây là một mô hình logic. Việc tổ chức thực tế các đối tượng trong hệ thống có thể khác nhau.

## 7.17 Mô hình trình tự(Sequence Model)

Các mô hình trình tự hiển thị chuỗi các tương tác đối tượng diễn ra

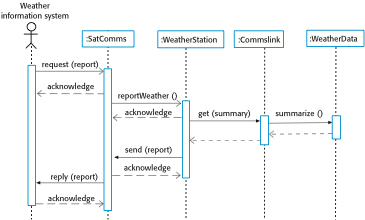
Các đối tượng được sắp xếp theo chiều ngang trên đầu;

Thời gian được biểu diễn theo chiều dọc để các mô hình được đọc từ trên xuống dưới;

Tương tác được biểu diễn bằng các mũi tên có nhãn, Các kiểu mũi tên khác nhau thể hiện các loại tương tác khác nhau;

Một hình chữ nhật trong một đường dây đối tượng biểu thị thời gian khi đối tượng là đối tượng điều khiển trong hệ thống.

## 7.18 Biểu đồ trình tự mô tả thu thập dữ liệu(Sequence Diagram Describing Data Collection)



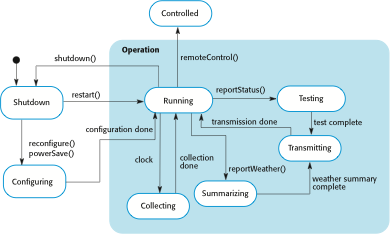
## 7.19 Sơ đồ nhà nước, Sơ đồ trạng thái(State Diagrams)(vcc sơ đồ nhà nước ☺)

Sơ đồ trạng thái được sử dụng để hiển thị cách đối tượng phản ứng với các yêu cầu dịch vụ khác nhau và chuyển tiếp trạng thái được kích hoạt bởi các yêu cầu này.

Sơ đồ trạng thái là các mô hình cấp cao hữu ích của một hệ thống hoặc hành vi thời gian chạy của đối tượng.

Bạn thường không cần một sơ đồ trạng thái cho tất cả các đối tượng trong hệ thống. Nhiều đối tượng trong một hệ thống tương đối đơn giản và một mô hình nhà nước thêm các chi tiết không cần thiết vào thiết kế.

**Sơ đồ trạng thái trạm thời tiết(Weather Station State Diagram)**



## 7.20 Đặc điểm giao diện(Interface Specification)

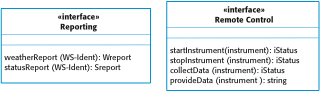
Các giao diện đối tượng phải được xác định để các đối tượng và các thành phần khác có thể được thiết kế song song.

Các nhà thiết kế nên tránh thiết kế biểu diễn giao diện nhưng nên ẩn nó trong chính đối tượng đó.

Các đối tượng có thể có một số giao diện là các quan điểm trên các phương thức được cung cấp.

UML sử dụng các sơ đồ lớp cho đặc tả giao diện nhưng Java cũng có thể được sử dụng.

**Giao diện trạm thời tiết(Weather station interfaces)**



**Key points**

Thiết kế và thực hiện phần mềm là các hoạt động liên ngành. Mức độ chi tiết trong thiết kế phụ thuộc vào loại hệ thống và liệu bạn có đang sử dụng phương pháp tiếp cận theo kế hoạch hoặc nhanh hay không.

Quá trình thiết kế hướng đối tượng bao gồm các hoạt động để thiết kế kiến trúc hệ thống, xác định các đối tượng trong hệ thống, mô tả thiết kế sử dụng các mô hình đối tượng khác nhau và ghi lại các giao diện thành phần.

Một loạt các mô hình khác nhau có thể được tạo ra trong quá trình thiết kế hướng đối tượng. Chúng bao gồm các mô hình tĩnh (các mô hình lớp, mô hình khái quát hóa, mô hình liên kết) và các mô hình động (các mô hình trình tự, các mô hình máy trạng thái).

Các giao diện thành phần phải được định nghĩa chính xác để các đối tượng khác có thể sử dụng chúng. Một khuôn mẫu giao diện UML có thể được sử dụng để định nghĩa các giao diện.

## 7.21 Mẫu thiết kế(Design Pattern)

* Một mẫu thiết kế là một cách để tái sử dụng kiến thức trừu tượng về một vấn đề và giải pháp của nó.
* Một mô hình là một mô tả về vấn đề và bản chất của giải pháp của nó.
* Nó phải đủ trừu tượng để được tái sử dụng trong các thiết lập khác nhau.
* Mô tả mẫu thường sử dụng các đặc tính hướng đối tượng như kế thừa và đa hình.

**Yếu tố mẫu(Pattern Elements)**

* Tên
* Một định danh mẫu có ý nghĩa.
* Mô tả vấn đề.
* Mô tả giải pháp.
  + Không phải là một thiết kế cụ thể mà là một mẫu cho một giải pháp thiết kế có thể được khởi tạo theo nhiều cách khác nhau.
* Hậu quả
  + Các kết quả và sự cân bằng của việc áp dụng mô hình.

## 7.22 Mẫu Observer

* Tên
* Quan sát viên.
* Sự miêu tả
* Tách hiển thị trạng thái đối tượng khỏi chính đối tượng đó.
* Mô tả vấn đề
* Được sử dụng khi cần hiển thị nhiều trạng thái.
* Mô tả giải pháp
* Xem trang trình bày với mô tả UML.
* Hậu quả
* Tối ưu hóa để nâng cao hiệu suất hiển thị là không thực tế.

**Description:** Phân tách hiển thị trạng thái của một đối tượng khỏi chính đối tượng đó và cho phép hiển thị các màn hình thay thế. Khi trạng thái đối tượng thay đổi, tất cả các màn hình sẽ tự động được thông báo và cập nhật để phản ánh thay đổi.

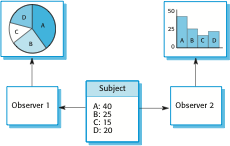
**Problem description:** Trong nhiều tình huống, bạn phải cung cấp nhiều hiển thị thông tin trạng thái, chẳng hạn như màn hình đồ họa và hiển thị bảng. Không phải tất cả những điều này có thể được biết khi thông tin được chỉ định. Tất cả các bản trình bày thay thế phải hỗ trợ tương tác và khi trạng thái được thay đổi, tất cả các màn hình phải được cập nhật.

Mẫu này có thể được sử dụng trong tất cả các tình huống có nhiều hơn một định dạng hiển thị cho thông tin trạng thái là bắt buộc và không cần thiết cho đối tượng duy trì thông tin trạng thái để biết về các định dạng hiển thị cụ thể được sử dụng.

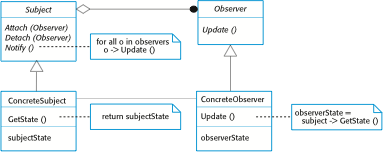
**Solution description:** Điều này liên quan đến hai đối tượng trừu tượng, Subject và Observer, và hai đối tượng cụ thể, ConcreteSubject và ConcreteObject, kế thừa các thuộc tính của các đối tượng trừu tượng có liên quan. Các đối tượng trừu tượng bao gồm các hoạt động chung được áp dụng trong mọi tình huống. Trạng thái được hiển thị được duy trì trong ConcreteSubject, kế thừa các hoạt động từ Subject cho phép nó thêm và loại bỏ các Trình theo dõi (mỗi người quan sát tương ứng với một màn hình) và đưa ra thông báo khi trạng thái đã thay đổi.

ConcreteObserver duy trì một bản sao trạng thái của ConcreteSubject và thực hiện giao diện Update () của Observer cho phép các bản sao này được lưu giữ trong bước. ConcreteObserver tự động hiển thị trạng thái và phản ánh các thay đổi bất cứ khi nào trạng thái được cập nhật.

## 7.23 Nhiều màn hình sử dụng mẫu Observer(Multiple Displays Using the Observer Pattern)



## 7.24 Mô hình UML của mẫu Observer



## 7.25 Vấn đề thiết kế

* Để sử dụng các mẫu trong thiết kế của bạn, bạn cần phải nhận ra rằng bất kỳ vấn đề thiết kế nào mà bạn đang gặp phải đều có thể có mẫu được liên kết có thể được áp dụng.
* Cho một số đối tượng biết rằng trạng thái của một số đối tượng khác đã thay đổi (mẫu Observer).
* Làm gọn các giao diện với một số đối tượng liên quan thường được phát triển dần dần (mẫu mặt tiền).
* Cung cấp cách truy cập chuẩn các phần tử trong một bộ sưu tập, bất kể cách bộ sưu tập đó được thực hiện như thế nào (mẫu Iterator).
* Cho phép khả năng mở rộng chức năng của một lớp hiện có tại thời gian chạy (mẫu Decorator).

## 7.26 Vấn đề thực hiện

Tập trung ở đây không phải là lập trình, mặc dù điều này rõ ràng là quan trọng, nhưng trên các vấn đề triển khai khác thường không được đề cập trong các văn bản lập trình:

* Tái sử dụng Hầu hết các phần mềm hiện đại được xây dựng bằng cách sử dụng lại các thành phần hoặc hệ thống hiện có. Khi bạn đang phát triển phần mềm, bạn nên sử dụng càng nhiều càng tốt mã hiện có.
* Quản lý cấu hình Trong quá trình phát triển, bạn phải theo dõi nhiều phiên bản khác nhau của mỗi thành phần phần mềm trong một hệ thống quản lý cấu hình.
* Phát triển máy chủ mục tiêu Phần mềm sản xuất thường không thực thi trên cùng một máy tính như môi trường phát triển phần mềm. Thay vào đó, bạn phát triển nó trên một máy tính (hệ thống máy chủ) và thực hiện nó trên một máy tính riêng biệt (hệ thống đích).

**Tái sử dụng**

* Từ những năm 1960 đến thập niên 1990, hầu hết các phần mềm mới được phát triển từ đầu, bằng cách viết tất cả các mã bằng một ngôn ngữ lập trình bậc cao.
* Việc tái sử dụng hoặc phần mềm quan trọng duy nhất là việc tái sử dụng các hàm và đối tượng trong các thư viện ngôn ngữ lập trình.
* Chi phí và áp lực lịch biểu có nghĩa là phương pháp này ngày càng trở nên không thay đổi, đặc biệt là cho các hệ thống thương mại và dựa trên Internet.
* Một cách tiếp cận để phát triển dựa trên việc tái sử dụng phần mềm hiện có nổi lên và hiện nay thường được sử dụng cho phần mềm kinh doanh và khoa học.

**Mức tái sử dụng**

Mức trừu tượng

* Ở cấp độ này, bạn không sử dụng lại phần mềm trực tiếp nhưng sử dụng kiến thức về các tóm tắt thành công trong việc thiết kế phần mềm của bạn.

Cấp độ đối tượng

* Ở cấp độ này, bạn trực tiếp sử dụng lại các đối tượng từ thư viện thay vì tự viết mã.

Cấp thành phần

* Các thành phần là các bộ sưu tập các đối tượng và các lớp đối tượng mà bạn sử dụng lại trong các hệ thống ứng dụng.

Mức hệ thống

* Ở cấp độ này, bạn sử dụng lại toàn bộ hệ thống ứng dụng.

**Chi phí tái sử dụng**

* Các chi phí của thời gian dành cho việc tìm kiếm phần mềm để tái sử dụng và đánh giá xem nó có đáp ứng nhu cầu của bạn hay không.
* Khi áp dụng, chi phí mua phần mềm tái sử dụng. Đối với các hệ thống lớn hơn, các chi phí này có thể rất cao.
* Chi phí thích ứng và cấu hình các thành phần hoặc hệ thống phần mềm có thể tái sử dụng để phản ánh các yêu cầu của hệ thống mà bạn đang phát triển.
* Chi phí tích hợp các phần tử phần mềm có thể tái sử dụng với nhau (nếu bạn đang sử dụng phần mềm từ các nguồn khác nhau) và với mã mới mà bạn đã phát triển.

## 7.27 Quản lý cấu hình

* Quản lý cấu hình là tên được đặt cho quá trình quản lý chung một hệ thống phần mềm thay đổi.
* Mục đích của quản lý cấu hình là hỗ trợ quá trình tích hợp hệ thống để tất cả các nhà phát triển có thể truy cập vào mã dự án và tài liệu theo cách được kiểm soát, tìm ra những thay đổi đã được thực hiện và biên dịch và liên kết các thành phần để tạo ra một hệ thống.

**Hoạt động quản lý cấu hình**

* Quản lý phiên bản, nơi hỗ trợ được cung cấp để theo dõi các phiên bản khác nhau của các thành phần phần mềm. Các hệ thống quản lý phiên bản bao gồm các cơ sở để phối hợp phát triển bởi một số lập trình viên.
* Tích hợp hệ thống, nơi hỗ trợ được cung cấp để giúp các nhà phát triển xác định phiên bản của các thành phần được sử dụng để tạo ra mỗi phiên bản của một hệ thống. Mô tả này sau đó được sử dụng để xây dựng một hệ thống tự động bằng cách biên dịch và liên kết các thành phần cần thiết.
* Theo dõi vấn đề, nơi hỗ trợ được cung cấp để cho phép người dùng báo cáo lỗi và các vấn đề khác và cho phép tất cả các nhà phát triển xem ai đang làm việc về những vấn đề này và khi chúng được khắc phục.

## 7.28 Host-Target Development

* Hầu hết các phần mềm được phát triển trên một máy tính (máy chủ), nhưng chạy trên một máy tính riêng biệt (mục tiêu).
  + Nói chung, chúng ta có thể nói về một nền tảng phát triển và một nền tảng thực thi.
* Một nền tảng không chỉ là phần cứng.
  + Nó bao gồm hệ điều hành được cài đặt cùng với các phần mềm hỗ trợ khác như hệ thống quản lý cơ sở dữ liệu hoặc cho các nền tảng phát triển, một môi trường phát triển tương tác.
  + Nền tảng phát triển thường có phần mềm cài đặt khác với nền tảng thực thi; các nền tảng này có thể có các kiến trúc khác nhau

**Công cụ nền tảng phát triển**

* Một trình biên dịch tích hợp và hệ thống chỉnh sửa cú pháp hướng cho phép bạn tạo, chỉnh sửa và biên dịch mã.
* Một hệ thống gỡ lỗi ngôn ngữ.
* Các công cụ chỉnh sửa đồ họa, chẳng hạn như các công cụ để chỉnh sửa các mô hình UML.
* Các công cụ kiểm tra, chẳng hạn như JUnit có thể tự động chạy một bộ kiểm thử trên một phiên bản mới của một chương trình.
* Các công cụ hỗ trợ dự án giúp bạn tổ chức mã cho các dự án phát triển khác nhau.

## 7.29 Môi trường phát triển tích hợp (IDE)

* Các công cụ phát triển phần mềm thường được nhóm lại để tạo ra một môi trường phát triển tích hợp (IDE).
* IDE là một bộ công cụ phần mềm hỗ trợ các khía cạnh khác nhau của phát triển phần mềm, trong một số khung công tác chung và giao diện người dùng.
* Các IDE được tạo ra để hỗ trợ phát triển trong một ngôn ngữ lập trình cụ thể như Java. IDE ngôn ngữ có thể được phát triển đặc biệt, hoặc có thể là sự khởi tạo của một IDE đa năng, với các công cụ hỗ trợ ngôn ngữ cụ thể.

## 7.30 Thành phần / Hệ thống Các yếu tố triển khai

* Nếu một thành phần được thiết kế cho một kiến trúc phần cứng cụ thể, hoặc dựa vào một số hệ thống phần mềm khác, thì rõ ràng nó phải được triển khai trên một nền tảng cung cấp sự hỗ trợ phần cứng và phần mềm cần thiết.
* Các hệ thống sẵn sàng cao có thể yêu cầu các thành phần được triển khai trên nhiều nền tảng. Điều này có nghĩa là, trong trường hợp xảy ra lỗi nền tảng, việc triển khai thay thế thành phần có sẵn.
* Nếu có lưu lượng truy cập thông tin liên lạc ở mức cao giữa các thành phần, thì việc triển khai chúng trên cùng một nền tảng hoặc trên các nền tảng thực tế gần với nhau là rất hợp lý. Điều này làm giảm sự chậm trễ giữa thời gian một tin nhắn được gửi bởi một thành phần và nhận được bởi một thành phần khác.

## 7.31 Phát triển nguồn mở

* Phát triển nguồn mở là một cách tiếp cận để phát triển phần mềm trong đó mã nguồn của một hệ thống phần mềm được xuất bản và các tình nguyện viên được mời tham gia vào quá trình phát triển
* Nguồn gốc của nó nằm trong Quỹ Phần mềm Tự do (www.fsf.org), mà chủ trương rằng mã nguồn không nên là độc quyền mà là luôn luôn có sẵn để người dùng kiểm tra và sửa đổi theo ý muốn.
* Phần mềm nguồn mở đã mở rộng ý tưởng này bằng cách sử dụng Internet để tuyển dụng một số lượng lớn các nhà phát triển tình nguyện viên. Nhiều người trong số họ cũng là người dùng mã.

**Hệ thống nguồn mở**

* Tất nhiên, sản phẩm nguồn mở nổi tiếng nhất là hệ điều hành Linux được sử dụng rộng rãi như một hệ thống máy chủ và ngày càng trở thành một môi trường máy tính để bàn.
* Các sản phẩm nguồn mở quan trọng khác là Java, máy chủ web Apache và hệ thống quản lý cơ sở dữ liệu mySQL.

**Các vấn đề nguồn mở**

* Sản phẩm đang được phát triển có sử dụng các thành phần nguồn mở không?
* Liệu một cách tiếp cận nguồn mở có được sử dụng cho sự phát triển của phần mềm không?

**Kinh doanh nguồn mở**

* Ngày càng có nhiều công ty sản xuất đang sử dụng một cách tiếp cận nguồn mở để phát triển.
* Mô hình kinh doanh của họ không phụ thuộc vào việc bán sản phẩm phần mềm mà là bán hỗ trợ cho sản phẩm đó.
* Họ tin rằng liên quan đến cộng đồng nguồn mở sẽ cho phép phần mềm được phát triển rẻ hơn, nhanh hơn và sẽ tạo ra một cộng đồng người dùng cho phần mềm.

**Cấp phép nguồn mở**

* Một nguyên tắc cơ bản của phát triển nguồn mở là mã nguồn nên được tự do sẵn có, điều này không có nghĩa là bất kỳ ai cũng có thể làm như họ mong muốn với mã đó.
* Về mặt pháp lý, nhà phát triển mã (hoặc công ty hoặc cá nhân) vẫn sở hữu mã. Họ có thể đặt những hạn chế về cách nó được sử dụng bằng cách bao gồm các điều kiện ràng buộc về mặt pháp lý trong giấy phép phần mềm nguồn mở.
* Một số nhà phát triển nguồn mở tin rằng nếu một thành phần nguồn mở được sử dụng để phát triển một hệ thống mới, thì hệ thống đó cũng phải là nguồn mở.
* Những người khác sẵn sàng cho phép mã của họ được sử dụng mà không có hạn chế này. Các hệ thống được phát triển có thể độc quyền và được bán như các hệ thống nguồn đóng.

## 7.32 Mô hình giấy phép(License Model)

* Giấy phép Công cộng GNU (GPL). Đây là giấy phép 'đối ứng' có nghĩa là nếu bạn sử dụng phần mềm nguồn mở được cấp phép theo giấy phép GPL, thì bạn phải làm cho phần mềm nguồn mở đó.
* Giấy phép Công cộng Ít hơn GNU (LGPL) là một biến thể của giấy phép GPL, nơi bạn có thể viết các thành phần liên kết tới mã nguồn mở mà không phải xuất bản nguồn của các thành phần này.
* Giấy phép phân phối tiêu chuẩn Berkley (BSD). Đây là một giấy phép không đối ứng, có nghĩa là bạn không có nghĩa vụ tái xuất bản bất kỳ thay đổi hoặc sửa đổi nào được thực hiện cho mã nguồn mở. Bạn có thể bao gồm mã trong các hệ thống độc quyền được bán.

**Quản lý Giấy phép**

* Thiết lập một hệ thống để duy trì thông tin về các thành phần nguồn mở được tải xuống và sử dụng.
* Hãy nhận biết các loại giấy phép khác nhau và hiểu cách một thành phần được cấp phép trước khi sử dụng.
* Hãy nhận biết các con đường tiến hóa cho các thành phần.
* Giáo dục mọi người về nguồn mở.
* Có hệ thống kiểm toán tại chỗ.
* Tham gia vào cộng đồng nguồn mở.

**Key points**

Khi phát triển phần mềm, bạn nên luôn xem xét khả năng sử dụng lại phần mềm hiện có, hoặc là thành phần, dịch vụ hoặc hệ thống hoàn chỉnh.

Quản lý cấu hình là quá trình quản lý các thay đổi đối với một hệ thống phần mềm phát triển. Điều quan trọng là khi một nhóm người đang hợp tác để phát triển phần mềm.

Hầu hết phát triển phần mềm là phát triển mục tiêu lưu trữ. Bạn sử dụng một IDE trên một máy chủ để phát triển phần mềm, được chuyển đến một máy mục tiêu để thực thi.

Phát triển mã nguồn mở liên quan đến việc làm cho mã nguồn của một hệ thống có sẵn công khai. Điều này có nghĩa là nhiều người có thể đề xuất những thay đổi và cải tiến cho phần mềm.

# Chương 8: Mô hình hệ thống.

Mô hình hóa hệ thống là quá trình phát triển các mô hình trừu tượng của một hệ thống, với mỗi mô hình trình bày một chức năng khác nhau của hệ thống đó.

Mô hình hóa hệ thống giờ đây có ý nghĩa đại diện cho một hệ thống sử dụng một loại ký pháp đồ họa nào đó, mà bây giờ hầu như luôn dựa trên các ký hiệu trong Ngôn ngữ mô hình hóa thống nhất (Unified Modeling Language - UML).

Mô hình hóa hệ thống giúp các nhà phân tích hiểu được chức năng của hệ thống và các mô hình được sử dụng để giao tiếp với khách hàng.

## 8.1 Kế hoạch cho mô hình hệ thống.

Các mô hình của hệ thống hiện có được sử dụng trong quá trình yêu cầu kỹ thuật. Nó giúp làm rõ những gì hệ thống hiện có làm và có thể được sử dụng làm cơ sở để thảo luận về điểm mạnh và điểm yếu của nó. Những điều này sau đó dẫn đến các yêu cầu cho hệ thống mới.

Các mô hình của hệ thống mới được sử dụng trong quá trình yêu cầu kỹ thuật để giúp giải thích các yêu cầu được đề xuất cho các bên liên quan khác của hệ thống. Các kỹ sư sử dụng các mô hình này để thảo luận các đề xuất thiết kế và lập tài liệu cho hệ thống để thực hiện.

Trong quy trình kỹ thuật theo mô hình, có thể tạo ra một hệ thống thực hiện toàn bộ hoặc một phần từ mô hình hệ thống.

## 8.2 Quan điểm hệ thống

An external perspective: nơi bạn mô hình ngữ cảnh hoặc môi trường của hệ thống.

An interaction perspective,: nơi bạn mô hình hóa các tương tác giữa một hệ thống và môi trường của nó, hoặc giữa các thành phần của một hệ thống.

A structural perspective, nơi bạn mô hình hóa tổ chức của một hệ thống hoặc cấu trúc của dữ liệu được xử lý bởi hệ thống.

A behavioral perspective: nơi bạn mô hình hóa hành vi động của hệ thống và cách nó phản ứng với các sự kiện.

## 8.3 Các sơ đồ uml.

### Biểu đồ Use case (Use Case Diagram):

Một biểu đồ Use case chỉ ra một số lượng các tác nhân ngoại cảnh và mối liên kết của chúng đối với Use case mà hệ thống cung cấp (nhìn hình 3.2). Một Use case là một lời miêu tả của một chức năng mà hệ thống cung cấp. Lời miêu tả Use case thường là một văn bản tài liệu, nhưng kèm theo đó cũng có thể là một biểu đồ hoạt động. Các Use case được miêu tả duy nhất theo hướng nhìn từ ngoài vào của các tác nhân (hành vi của hệ thống theo như sự mong đợi của người sử dụng), không miêu tả chức năng được cung cấp sẽ hoạt động nội bộ bên trong hệ thống ra sao. Các Use case định nghĩa các yêu cầu về mặt chức năng đối với hệ thống. Các biểu đồ Use case sẽ được miêu tả chi tiết hơn trong chương 4 (Use case).

### Biểu đồ lớp (Class Diagram):

Một biểu đồ lớp chỉ ra cấu trúc tĩnh của các lớp trong hệ thống (nhìn hình 3.3). Các lớp là đại diện cho các “vật” được xử lý trong hệ thống. Các lớp có thể quan hệ với nhau trong nhiều dạng thức: liên kết (associated - được nối kết với nhau), phụ thuộc (dependent - một lớp này phụ thuộc vào lớp khác), chuyên biệt hóa (specialized - một lớp này là một kết quả chuyên biệt hóa của lớp khác), hay đóng gói ( packaged - hợp với nhau thành một đơn vị). Tất cả các mối quan hệ đó đều được thể hiện trong biểu đồ lớp, đi kèm với cấu trúc bên trong của các lớp theo khái niệm thuộc tính (attribute) và thủ tục (operation). Biểu đồ được coi là biểu đồ tĩnh theo phương diện cấu trúc được miêu tả ở đây có hiệu lực tại bất kỳ thời điểm nào trong toàn bộ vòng đời hệ thống.

Một hệ thống thường sẽ có một loạt các biểu đồ lớp – chẳng phải bao giờ tất cả các biểu đồ lớp này cũng được nhập vào một biểu đồ lớp tổng thể duy nhất – và một lớp có thể tham gia vào nhiều biểu đồ lớp. Biểu đồ lớp được miêu tả chi tiết trong chương sau.

### Biểu đồ trạng thái (State Diagram):

Một biểu đồ trạng thái thường là một sự bổ sung cho lời miêu tả một lớp. Nó chỉ ra tất cả các trạng thái mà đối tượng của lớp này có thể có, và những sự kiện (event) nào sẽ gây ra sự thay đổi trạng thái (hình 3.5). Một sự kiện có thể xảy ra khi một đối tượng tự gửi thông điệp đến cho nó - ví dụ như để thông báo rằng một khoảng thời gian được xác định đã qua đi – hay là một số điều kiện nào đó đã được thỏa mãn. Một sự thay đổi trạng thái được gọi là một sự chuyển đổi trạng thái (State Transition). Một chuyển đổi trạng thái cũng có thể có một hành động liên quan, xác định điều gì phải được thực hiện khi sự chuyển đổi trạng thái này diễn ra.

Biểu đồ trạng thái không được vẽ cho tất cả các lớp, mà chỉ riêng cho những lớp có một số lượng các trạng thái được định nghĩa rõ ràng và hành vi của lớp bị ảnh hưởng và thay đổi qua các trạng thái khác nhau. Biểu đồ trạng thái cũng có thể được vẽ cho hệ thống tổng thể. Biểu đồ trạng thái được miêu tả chi tiết hơn trong chương sau (Mô hình động).

### Biểu đồ trình tự (Sequence Diagram):

Một biểu đồ trình tự chỉ ra một cộng tác động giữa một loạt các đối tượng (xem hình 3.6). Khía cạnh quan trọng của biểu đồ này là chỉ ra trình tự các thông điệp (message) được gửi giữa các đối tượng. Nó cũng chỉ ra trình tự tương tác giữa các đối tượng, điều sẽ xảy ra tại một thời điểm cụ thể nào đó trong trình tự thực thi của hệ thống. Các biểu đồ trình tự chứa một loạt các đối tượng được biểu diễn bằng các đường thẳng đứng. Trục thời gian có hướng từ trên xuống dưới trong biểu đồ, và biểu đồ chỉ ra sự trao đổi thông điệp giữa các đối tượng khi thời gian trôi qua. Các thông điệp được biểu diễn bằng các đường gạch ngang gắn liền với mũi tên (biểu thị thông điệp) nối liền giữa những đường thẳng đứng thể hiện đối tượng. Trục thời gian cùng những lời nhận xét khác thường sẽ được đưa vào phần lề của biểu đồ.

Biểu đồ Activity Diagrams:

Biểu diễn hoạt dộng của một hệ thống từ lúc bắt đầu đến khi kết thúc.

## 8.4 Context Models

Context Models được sử dụng để minh họa bối cảnh hoạt động của một hệ thống - chúng hiển thị những gì nằm bên ngoài ranh giới hệ thống.

Cách tổ chức của mỗi công ty có thể ảnh hưởng đến quyết định vị trí đặt ranh giới hệ thống.

Các mô hình kiến ​​trúc cho thấy hệ thống và mối quan hệ của nó với các hệ thống khác.

## 8.5 Ranh giới hệ thống.

System Boundaries được thiết lập để xác định những gì bên trong và những gì nằm ngoài hệ thống.

Chúng hiển thị các hệ thống khác được sử dụng hoặc phụ thuộc vào hệ thống đang được phát triển.

Vị trí của system boundary có ảnh hưởng sâu sắc đến các yêu cầu hệ thống.

Có thể có những áp lực để phát triển các system boundary làm tăng / giảm ảnh hưởng hoặc khối lượng công việc của các phần khác nhau của một tổ chức.

## 8.6 Phối cảnh quy trình.

Các mô hình ngữ cảnh chỉ đơn giản hiển thị các hệ thống khác trong môi trường.

Các mô hình quy trình cho thấy cách hệ thống đang được phát triển được sử dụng trong các quy trình nghiệp vụ rộng hơn.

Biểu đồ hoạt động UML có thể được sử dụng để xác định các mô hình quy trình nghiệp vụ.

## 8.7Mô hình tương tác.

Việc mô hình hóa tương tác người dùng rất quan trọng vì nó giúp xác định các yêu cầu của người dùng.

Mô hình hóa tương tác hệ thống với hệ thống làm nổi bật các vấn đề giao tiếp có thể phát sinh.

Việc tạo mô hình tương tác thành phần giúp chúng ta hiểu được liệu cấu trúc hệ thống được đề xuất có khả năng mang lại hiệu suất và độ tin cậy của hệ thống được yêu cầu hay không.

Sử dụng case diagrams and sequence diagrams có thể được sử dụng cho mô hình tương tác.

## Use Case Modeling.

Use Case Modeling đã được phát triển ban đầu để hỗ trợ yêu cầu gợi ý và bây giờ được tích hợp vào UML.

Mỗi use case đại diện cho một chức năng có liên quan đến sự tương tác bên ngoài với một hệ thống khác.

Các Actor trong trường hợp sử dụng có thể là người hoặc các hệ thống khác.

Đại diện theo sơ đồ để cung cấp tổng quan về trường hợp sử dụng và dưới dạng văn bản chi tiết hơn.

## 8.8 Các điểm chính.

Mô hình là chế độ xem trừu tượng của hệ thống bỏ qua chi tiết hệ thống. Các mô hình hệ thống bổ sung có thể được phát triển để hiển thị bối cảnh, tương tác, cấu trúc và hành vi của hệ thống.

Các mô hình ngữ cảnh cho thấy cách một hệ thống đang được mô hình hóa được định vị trong môi trường với các hệ thống và quy trình khác.

Use case diagrams and sequence diagrams được sử dụng để mô tả tương tác giữa người dùng và hệ thống trong hệ thống đang được thiết kế. Các use case mô tả tương tác giữa một hệ thống và các tác nhân bên ngoài; sơ đồ tuần tự thêm nhiều thông tin hơn vào các biểu đồ này bằng cách hiển thị tương tác giữa các đối tượng hệ thống.

Structural models cho thấy tổ chức và kiến ​​trúc của một hệ thống. Các sơ đồ lớp được sử dụng để định nghĩa cấu trúc tĩnh của các lớp trong một hệ thống và các liên kết của chúng.

## 8.9 Tổng quát.

Generalization là một kỹ thuật hàng ngày mà chúng tôi sử dụng để quản lý sự phức tạp.

Thay vì tìm hiểu các đặc điểm chi tiết của mọi thực thể mà chúng ta trải nghiệm, chúng ta đặt các thực thể này trong các lớp học tổng quát hơn (động vật, xe hơi, nhà cửa, vv) và tìm hiểu các đặc tính của các lớp này.

Điều này cho phép chúng ta phỏng đoán rằng các lớp khác nhau của các lớp này có một số đặc điểm chung, ví dụ: sóc và chuột là loài gặm nhấm.

Trong các hệ thống mô hình hóa, thường rất dễ dàng để kiểm tra các lớp trong một hệ thống để xem có phạm vi tổng quát hay không. Nếu thay đổi được đề xuất, thì bạn không phải xem tất cả các lớp trong hệ thống để xem liệu chúng có bị ảnh hưởng bởi thay đổi hay không.

Trong các ngôn ngữ hướng đối tượng, chẳng hạn như Java, việc tổng quát hóa được thực hiện bằng cách sử dụng các cơ chế thừa kế lớp được xây dựng trong ngôn ngữ.

Nói chung, các thuộc tính và các hoạt động liên kết với các lớp cấp cao hơn cũng được liên kết với các lớp cấp thấp hơn.

Các lớp cấp thấp hơn là các lớp con kế thừa các thuộc tính và các hoạt động từ các lớp bậc trên của chúng. Các lớp cấp thấp hơn này sau đó thêm các thuộc tính và hoạt động cụ thể hơn.

## 8.10 Mô hình hành vi.

Các mô hình hành vi là các mô hình của hành vi động của một hệ thống khi nó đang thực thi. Chúng cho thấy những gì xảy ra hoặc những gì được cho là xảy ra khi một hệ thống phản ứng với một kích thích từ môi trường của nó.

Nó được chia thành 2 loại:

+Data: số dữ liệu đến mà phải được xử lý bởi hệ thống.

+Event: Một số sự kiện xảy ra sẽ kích hoạt xử lý hệ thống. Sự kiện có thể có dữ liệu được liên kết, mặc dù điều này không phải luôn luôn như vậy.

## 8.11 Mô hình hướng sự kiện

Các hệ thống thời gian thực thường được định hướng theo sự kiện, với xử lý dữ liệu tối thiểu. Ví dụ, một hệ thống chuyển mạch điện thoại cố định đáp ứng các sự kiện như 'móc lấy móc' bằng cách tạo ra âm quay số.

Mô hình theo hướng sự kiện cho thấy cách hệ thống phản hồi các sự kiện bên ngoài và bên trong.

Nó dựa trên giả định rằng một hệ thống có một số hữu hạn các trạng thái và các sự kiện (kích thích) có thể gây ra sự chuyển đổi từ trạng thái này sang trạng thái khác.

## 8.12 Kỹ thuật theo mô hình.

Kỹ thuật theo mô hình (MDE) là một cách tiếp cận để phát triển phần mềm, nơi các mô hình thay vì các chương trình là các đầu ra chính của quá trình phát triển.

Các chương trình thực hiện trên nền tảng phần cứng / phần mềm sau đó được tạo tự động từ các mô hình.

Những người ủng hộ MDE lập luận rằng điều này làm tăng mức độ trừu tượng trong kĩ nghệ phần mềm sao cho các kỹ sư không còn phải quan tâm đến chi tiết ngôn ngữ lập trình hay chi tiết cụ thể của nền tảng thực thi.

## 8.13 Sử dụng kỹ thuật theo mô hình

**Model-Driven Engineering** vẫn đang ở giai đoạn phát triển ban đầu, và không rõ liệu nó có ảnh hưởng đáng kể đến thực hành kỹ nghệ phần mềm hay không.

Ưu điểm

+Cho phép hệ thống được xem xét ở mức trừu tượng cao hơn

+Tạo mã tự động có nghĩa là nó rẻ hơn để thích ứng với hệ thống với nền tảng mới.

Nhược điểm

+Mô hình trừu tượng và không nhất thiết phải thực hiện.

+Tiết kiệm từ việc tạo mã có thể bị ảnh hưởng bởi chi phí phát triển dịch cho nền tảng mới.

## 8.14 Các loại mô hình.

* **Mô hình độc lập tính toán (CIM)** 
  + Những mô hình trừu tượng miền quan trọng được sử dụng trong một hệ thống. CIM đôi khi được gọi là mô hình miền.
* **Mô hình độc lập nền tảng (PIM)**
* Những mô hình hoạt động của hệ thống mà không tham chiếu đến việc thực hiện nó. PIM thường được mô tả bằng cách sử dụng các mô hình UML hiển thị cấu trúc hệ thống tĩnh và cách nó phản ứng với các sự kiện bên ngoài và bên trong.
* **Mô hình nền tảng cụ thể*(PSM)***
  + Đây là những biến đổi của mô hình nền tảng độc lập với PSM riêng biệt cho từng nền tảng ứng dụng. Về nguyên tắc, có thể có các lớp PSM, với mỗi lớp thêm một số chi tiết cụ thể cho nền tảng.

# Chương 9

## 9.1 Kỹ thuật phần mềm dựa trên thành phần.

Phát triển dựa trên thành phần.

* + Kỹ nghệ phần mềm dựa trên thành phần (CBSE) là một cách tiếp cận để phát triển phần mềm dựa trên việc tái sử dụng các thực thể được gọi là 'các thành phần phần mềm'.
  + Nó xuất hiện từ sự thất bại của phát triển hướng đối tượng để hỗ trợ tái sử dụng hiệu quả. Các lớp đối tượng đơn quá chi tiết và cụ thể.
  + Các thành phần trừu tượng hơn các lớp đối tượng và có thể được coi là các nhà cung cấp dịch vụ độc lập. Chúng có thể tồn tại như một thực thể độc lập.

## 9.2 CBSE Essentials

* Các thành phần độc lập được chỉ định bởi giao diện của chúng.
* Các tiêu chuẩn thành phần để tạo thuận lợi cho việc tích hợp thành phần.
* Middleware cung cấp hỗ trợ cho khả năng tương tác thành phần.
* Một quá trình phát triển được thiết kế để tái sử dụng.

**CBSE và nguyên tắc thiết kế**

Ngoài những lợi ích của việc tái sử dụng, CBSE dựa trên các nguyên tắc thiết kế:

+Các thành phần độc lập nên không can thiệp lẫn nhau;

+Triển khai thành phần bị ẩn;

+Giao tiếp là thông qua các giao diện được xác định rõ;

+Các nền tảng thành phần được chia sẻ và giảm chi phí phát triển.

**Tiêu chuẩn thành phần.**

Tiêu chuẩn cần được thiết lập để các thành phần có thể giao tiếp với nhau và liên kết.

Thật không may, một số tiêu chuẩn thành phần cạnh tranh đã được thiết lập:

+Java của doanh nghiệp Sun

+COM và .NET của Microsoft

+Corba của CCM

Trong thực tế, các tiêu chuẩn này đã cản trở sự hấp thu của CBSE. Các thành phần không thể phát triển bằng cách sử dụng các cách tiếp cận khác nhau để làm việc cùng nhau.

**Các vấn đề của CBSE.**

+Tính đáng tin cậy của thành phần - làm cách nào một thành phần không có mã nguồn có sẵn được tin cậy?

+Chứng nhận thành phần - ai sẽ chứng nhận chất lượng của các thành phần?

+Dự đoán tài sản khẩn cấp - làm thế nào để các thuộc tính nổi bật của các thành phần cấu thành được dự đoán?

+Yêu cầu thương mại-off - làm thế nào để chúng tôi làm phân tích thương mại-off giữa các tính năng của một thành phần và khác?

## 9.3 Các thành phần.

Các thành phần cung cấp một dịch vụ mà không quan tâm đến nơi thành phần đang thực thi hoặc ngôn ngữ lập trình của nó

Một thành phần là một thực thể độc lập có thể được tạo thành từ một hoặc nhiều đối tượng thực thi;

Giao diện thành phần được xuất bản và tất cả các tương tác đều thông qua giao diện được xuất bản;

## 9.4 Định nghĩa thành phần

Councill và Heinmann:

Thành phần phần mềm là phần tử phần mềm phù hợp với mô hình thành phần và có thể được triển khai và soạn thảo độc lập mà không sửa đổi theo tiêu chuẩn thành phần.

Szyperski:

Một thành phần phần mềm là một đơn vị thành phần với các giao diện được chỉ định theo hợp đồng và chỉ phụ thuộc ngữ cảnh rõ ràng. Một thành phần phần mềm có thể được triển khai độc lập và phải chịu sự sáng tác của các bên thứ ba.

## 9.5 Thành phần đặc điểm.

* **Standardized (tiêu chuẩn hóa):** Chuẩn hóa thành phần có nghĩa là một thành phần được sử dụng trong quy trình CBSE phải tuân thủ một mô hình thành phần tiêu chuẩn. Mô hình này có thể xác định các giao diện thành phần, siêu dữ liệu thành phần, tài liệu, thành phần và triển khai.
* **Independent ( độc lập):** Một thành phần phải độc lập - có thể soạn và triển khai nó mà không cần phải sử dụng các thành phần cụ thể khác. Trong các trường hợp thành phần cần các dịch vụ được cung cấp bên ngoài, các thành phần này phải được đặt rõ ràng trong thông số giao diện 'yêu cầu'.
* **Composaple (Google đ dịch đc):** Để một thành phần có thể được tổng hợp, tất cả các tương tác bên ngoài phải diễn ra thông qua các giao diện được xác định công khai. Ngoài ra, nó phải cung cấp quyền truy cập bên ngoài vào thông tin về chính nó, chẳng hạn như các phương thức và thuộc tính của nó.
* **Deployable ( có thể triển khai):** Để có thể triển khai, một thành phần phải được khép kín. Nó phải có khả năng hoạt động như một thực thể độc lập trên một nền tảng thành phần cung cấp việc thực hiện mô hình thành phần. Điều này thường có nghĩa là thành phần là nhị phân và không phải được biên dịch trước khi nó được triển khai. Nếu một thành phần được thực hiện như một dịch vụ, thì nó không phải được người dùng của một thành phần triển khai. Thay vào đó, nó được triển khai bởi nhà cung cấp dịch vụ.
* **Documented ( tài liệu ):** Các thành phần phải được ghi chép đầy đủ để người dùng tiềm năng có thể quyết định xem các thành phần có đáp ứng nhu cầu của họ hay không. Cú pháp và, lý tưởng, ngữ nghĩa của tất cả các giao diện thành phần cần được xác định.

## 9.6 Thành phần với tư cách là nhà cung cấp dịch vụ.

Thành phần là một thực thể độc lập, thực thi. Nó không phải được biên dịch trước khi nó được sử dụng với các thành phần khác.

Các dịch vụ được cung cấp bởi một thành phần được cung cấp thông qua một giao diện và tất cả các tương tác thành phần diễn ra thông qua giao diện đó.

Giao diện thành phần được thể hiện dưới dạng các phép toán tham số và trạng thái bên trong của nó không bao giờ được tiếp xúc.

**Giao diện thành phần**

Cung cấp giao diện

Xác định các dịch vụ được cung cấp bởi các thành phần cho các thành phần khác.

Giao diện này, về cơ bản, là API thành phần. Nó định nghĩa các phương thức có thể được gọi bởi người dùng của thành phần.

Yêu cầu giao diện

Xác định các dịch vụ chỉ định các dịch vụ nào phải được cung cấp cho thành phần để thực hiện như được chỉ định.

Điều này không ảnh hưởng đến tính độc lập hoặc khả năng triển khai của một thành phần vì giao diện ‘yêu cầu’ không xác định cách thức cung cấp các dịch vụ này.

## 9.7 Mô hình thành phần.

Một mô hình thành phần là một định nghĩa về các tiêu chuẩn để thực hiện thành phần, tài liệu và triển khai.

Ví dụ về các mô hình thành phần

Mô hình EJB (Enterprise Java Beans)

Mô hình COM + (mô hình .NET)

Mô hình thành phần Corba

Mô hình thành phần xác định cách các giao diện nên được định nghĩa và các phần tử cần được bao gồm trong định nghĩa giao diện.

### 9.7.1 Các thành phần của mô hình thành phần

Giao diện

Các thành phần được định nghĩa bằng cách xác định giao diện của chúng. Mô hình thành phần xác định cách các giao diện cần được xác định và các phần tử, chẳng hạn như các tên hoạt động, các tham số và các ngoại lệ, nên được bao gồm trong định nghĩa giao diện.

Sử dụng

Để các thành phần được phân phối và truy cập từ xa, chúng cần phải có một tên duy nhất hoặc xử lý được liên kết với chúng. Điều này phải là duy nhất trên toàn cầu.

Triển khai

Mô hình thành phần bao gồm một đặc điểm kỹ thuật về cách các thành phần nên được đóng gói để triển khai như các thực thể độc lập, thực thi.

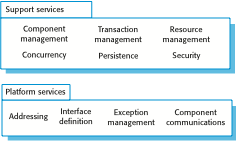
## 9.8 Middleware Hỗ trợ.

Các mô hình thành phần là cơ sở cho phần mềm trung gian cung cấp hỗ trợ cho việc thực hiện các thành phần.

Triển khai mô hình thành phần cung cấp:

* Dịch vụ nền tảng cho phép các thành phần được viết theo mô hình để giao tiếp;
* Các dịch vụ hỗ trợ là các dịch vụ độc lập ứng dụng được sử dụng bởi các thành phần khác nhau.
* Để sử dụng các dịch vụ được cung cấp bởi một mô hình, các thành phần được triển khai trong một thùng chứa. Đây là một bộ các giao diện được sử dụng để truy cập các triển khai dịch vụ.

### 9.8.1 Dịch vụ Middleware được định nghĩa trong một mô hình thành phần



## 9.9 Qúa Trinh CBSE.

Quy trình CBSE là các quá trình phần mềm hỗ trợ kỹ thuật phần mềm dựa trên thành phần.

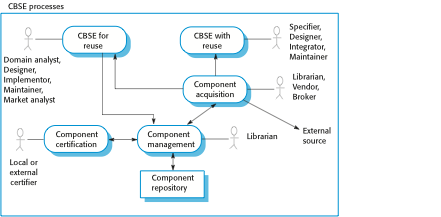
+Họ đưa vào tài khoản các khả năng tái sử dụng và các hoạt động quy trình khác nhau liên quan đến việc phát triển và sử dụng các thành phần tái sử dụng.

Phát triển tái sử dụng

+Quá trình này liên quan đến việc phát triển các thành phần hoặc dịch vụ sẽ được tái sử dụng trong các ứng dụng khác. Nó thường liên quan đến việc tổng hợp các thành phần hiện có.

Phát triển với tái sử dụng

+Quá trình này là quá trình phát triển các ứng dụng mới sử dụng các thành phần và dịch vụ hiện có.



## 9.10 Quy trình hỗ trợ.

Mua lại hợp phần là quá trình mua lại các thành phần để tái sử dụng hoặc phát triển thành một thành phần có thể sử dụng lại được.

+Nó có thể liên quan đến việc truy cập các thành phần hoặc dịch vụ do địa phương phát triển hoặc tìm các thành phần này từ một nguồn bên ngoài.

Quản lý thành phần liên quan đến việc quản lý các thành phần có thể sử dụng lại của công ty, đảm bảo rằng chúng được phân loại chính xác, được lưu trữ và cung cấp để sử dụng lại.

Chứng nhận thành phần là quá trình kiểm tra một thành phần và xác nhận rằng nó đáp ứng đặc điểm kỹ thuật của nó.

## 9.11 CBSE để tái sử dụng.

CBSE để tái sử dụng là quá trình phát triển các thành phần tái sử dụng và làm cho chúng có sẵn để tái sử dụng thông qua một hệ thống quản lý thành phần.

CBSE để tái sử dụng tập trung vào phát triển thành phần.

Các thành phần được phát triển cho một ứng dụng cụ thể thường phải được khái quát hóa để làm cho chúng có thể tái sử dụng được.

Một thành phần có nhiều khả năng được tái sử dụng nếu nó được kết hợp với một sự trừu tượng miền ổn định (đối tượng nghiệp vụ).

Ví dụ, trong một trừu tượng miền ổn định của bệnh viện có liên quan với mục đích cơ bản - y tá, bệnh nhân, phương pháp điều trị, v.v.

**Phát triển thành phần để tái sử dụng.**

Các thành phần để tái sử dụng có thể được xây dựng đặc biệt bằng cách khái quát các thành phần hiện có.

Tái sử dụng thành phần

+Nên phản ánh trừu tượng miền ổn định;

+Nên ẩn đại diện nhà nước;

+Nên càng độc lập càng tốt;

+Nên xuất bản ngoại lệ thông qua giao diện thành phần.

## 9.12 Nên phản ánh trừu tượng miền ổn định.

Các yếu tố cơ bản của miền ứng dụng có thay đổi chậm không

Ví dụ

* Trong hệ thống ngân hàng, trừu tượng tên miền có thể bao gồm tài khoản, chủ tài khoản và báo cáo.
* Trong một hệ thống quản lý bệnh viện, trừu tượng miền có thể bao gồm bệnh nhân, điều trị và y tá.

Tóm tắt tên miền này đôi khi được gọi là ‘đối tượng kinh doanh’.

Nếu thành phần là một triển khai thực hiện một cách trừu tượng tên miền được sử dụng phổ biến hoặc nhóm các đối tượng kinh doanh liên quan, nó có thể có thể được tái sử dụng.

## 9.13 Phát triển thành phần để tái sử dụng.

Có sự cân bằng giữa khả năng sử dụng lại và khả năng sử dụng

+Giao diện tổng quát hơn, khả năng sử dụng lại càng lớn nhưng sau đó nó phức tạp hơn và do đó ít khả dụng hơn.

**Thay đổi về khả năng sử dụng lại.**

Loại bỏ các phương pháp cụ thể cho ứng dụng.

Thay đổi tên để làm cho chúng chung chung.

Thêm phương pháp để mở rộng phạm vi phủ sóng.

Hãy xử lý ngoại lệ nhất quán.

Thêm một giao diện cấu hình để thích ứng thành phần.

Tích hợp các thành phần cần thiết để giảm phụ thuộc.

## 9.14 Xử lý ngoại lệ.

Các thành phần không nên tự xử lý các ngoại lệ, vì mỗi ứng dụng sẽ có các yêu cầu riêng của nó để xử lý ngoại lệ.

Thay vào đó, thành phần nên xác định những ngoại lệ nào có thể phát sinh và nên xuất bản chúng như một phần của giao diện.

Trong thực tế, tuy nhiên, có hai vấn đề với điều này:

Xuất bản tất cả các ngoại lệ dẫn đến các giao diện cồng kềnh khó hiểu hơn. Điều này có thể khiến người dùng tiềm năng của thành phần.

Các hoạt động của các thành phần có thể phụ thuộc vào xử lý ngoại lệ địa phương, và thay đổi điều này có thể có ý nghĩa nghiêm trọng cho các chức năng của các thành phần.

## 9.15 Các thành phần hệ thống cũ.

Các hệ thống cũ hiện có đáp ứng chức năng kinh doanh hữu ích có thể được đóng gói lại thành các thành phần để tái sử dụng.

Điều này liên quan đến việc viết một thành phần bao bọc thực hiện cung cấp và yêu cầu các giao diện sau đó truy cập vào hệ thống kế thừa.

Mặc dù tốn kém, điều này có thể rẻ hơn nhiều so với việc viết lại hệ thống cũ.

**Các thành phần có thể tái sử dụng.**

Chi phí phát triển của các thành phần tái sử dụng có thể cao hơn chi phí tương đương cụ thể. Chi phí tăng cường khả năng sử dụng lại bổ sung này phải là một tổ chức chứ không phải là chi phí dự án.

Các thành phần chung có thể ít hiệu quả về mặt không gian hơn và có thể có thời gian thực thi lâu hơn so với các phần tử tương đương cụ thể của chúng.

**Quản lý thành phần.**

Quản lý thành phần bao gồm việc quyết định cách phân loại thành phần để có thể phát hiện thành phần đó, làm cho thành phần có sẵn trong kho hoặc dưới dạng dịch vụ, duy trì thông tin về việc sử dụng thành phần và theo dõi các phiên bản thành phần khác nhau.

Một công ty có chương trình tái sử dụng có thể thực hiện một số hình thức chứng nhận thành phần trước khi thành phần được cung cấp để tái sử dụng.

Chứng nhận có nghĩa là một người ngoài nhà phát triển kiểm tra chất lượng của thành phần.

## 9.16 CBSE với tái sử dụng.

CBSE với quá trình tái sử dụng phải tìm và tích hợp các thành phần tái sử dụng.

Khi sử dụng lại các thành phần, điều cần thiết là tạo ra sự cân bằng giữa các yêu cầu lý tưởng và các dịch vụ thực sự được cung cấp bởi các thành phần có sẵn.

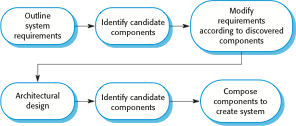
Điều này bao gồm:

+Phát triển các yêu cầu phác thảo;

+Tìm kiếm các thành phần sau đó sửa đổi các yêu cầu theo chức năng có sẵn.

+Tìm kiếm lại để tìm xem có các thành phần nào tốt hơn đáp ứng các yêu cầu đã sửa đổi hay không.

+Soạn các thành phần để tạo hệ thống.



## 9.17 Quy trình nhận dạng thành phần.

**17.8 IdentifyComps.eps**

## 9.18 Các vấn đề về nhận dạng thành phần.

Lòng tin: Bạn cần có khả năng tin tưởng nhà cung cấp của một thành phần. Tốt nhất, một thành phần không đáng tin cậy có thể không hoạt động như quảng cáo; tồi tệ nhất, nó có thể vi phạm an ninh của bạn.

Yêu cầu: Các nhóm thành phần khác nhau sẽ đáp ứng các yêu cầu khác nhau.

Xác thực: Đặc tả thành phần có thể không đủ chi tiết để cho phép phát triển các thử nghiệm toàn diện.

+Các thành phần có thể có chức năng không mong muốn. Làm thế nào bạn có thể kiểm tra điều này sẽ không can thiệp vào ứng dụng của bạn?

## 9.19 Xác thực thành phần.

Xác nhận hợp phần bao gồm việc phát triển một bộ các trường hợp thử nghiệm cho một thành phần (hoặc, có thể, mở rộng các trường hợp thử nghiệm được cung cấp với thành phần đó) và phát triển một khai thác thử nghiệm để chạy các kiểm tra thành phần.

Vấn đề chính với việc xác thực hợp phần là đặc tả thành phần có thể không đầy đủ chi tiết để cho phép bạn phát triển một bộ kiểm tra thành phần hoàn chỉnh.

Cũng như thử nghiệm rằng một thành phần để tái sử dụng thực hiện những gì bạn yêu cầu, bạn cũng có thể phải kiểm tra xem thành phần đó có bao gồm bất kỳ mã hoặc chức năng độc hại nào mà bạn không cần.

**Lỗi phóng Ariane - lỗi xác thực?**

Vào năm 1996, chuyến bay thử nghiệm đầu tiên của tên lửa Ariane 5 đã kết thúc trong thảm họa khi phóng đã mất kiểm soát 37 giây sau khi cất cánh.

Vấn đề là do một thành phần được sử dụng lại từ phiên bản trước của trình khởi chạy (Hệ thống điều hướng quán tính) không thành công vì các giả định được thực hiện khi thành phần đó được phát triển không giữ cho Ariane 5.

Chức năng không thành công trong thành phần này không được yêu cầu trong Ariane 5.

**Tổng hợp.**

Trong quá trình CBSE, các quy trình kỹ thuật yêu cầu và thiết kế hệ thống được xen kẽ.

Thành phần cấu phần là quá trình của các thành phần "nối dây" với nhau để tạo ra một hệ thống.

Khi soạn các thành phần có thể tái sử dụng, bạn thường phải viết các bộ điều hợp để đối chiếu các giao diện thành phần khác nhau.

Khi chọn bố cục, bạn phải xem xét các chức năng cần thiết, các yêu cầu phi chức năng và tiến hóa hệ thống.

# Chapter 10: Thiết kế kiến ​​trúc

**Chủ đề được đề cập**

* Quyết định thiết kế kiến ​​trúc
* Quan điểm kiến ​​trúc
* Mẫu kiến ​​trúc
* Kiến trúc ứng dụng

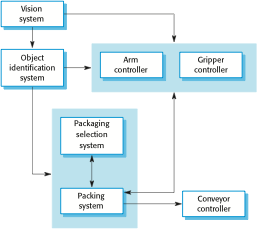
## 10.1 Kiến trúc phần mềm

* Quá trình thiết kế để xác định các hệ thống con tạo nên một hệ thống và khuôn khổ để kiểm soát và truyền thông hệ thống phụ là Thiết kế Kiến trúc.
* Đầu ra của quá trình thiết kế này là một mô tả về kiến ​​trúc phần mềm.

**Thiết kế kiến ​​trúc**

* Thiết kế kiến ​​trúc có liên quan với sự hiểu biết làm thế nào một hệ thống nên được tổ chức và thiết kế cấu trúc tổng thể của hệ thống đó
* Giai đoạn đầu của quá trình thiết kế hệ thống.
* Đại diện cho liên kết giữa quy trình thiết kế và đặc điểm kỹ thuật.
* Thường được thực hiện song song với một số hoạt động đặc điểm kỹ thuật.
* Nó liên quan đến việc xác định các thành phần hệ thống chính và thông tin liên lạc của chúng.

“Đầu ra của quá trình thiết kế kiến ​​trúc là một mô hình kiến ​​trúc mô tả cách hệ thống được tổ chức như một tập hợp các thành phần giao tiếp”



**Tóm tắt kiến ​​trúc**

Bạn có thể thiết kế kiến ​​trúc phần mềm ở 2 mức trừu tượng:

* Kiến trúc trong nhỏ có liên quan đến kiến ​​trúc của các chương trình riêng lẻ. Ở cấp độ này, chúng tôi quan tâm đến cách thức mà một chương trình riêng lẻ được phân tách thành các thành phần.
* Kiến trúc lớn có liên quan đến kiến ​​trúc của các hệ thống doanh nghiệp phức tạp bao gồm các hệ thống, chương trình và thành phần chương trình khác. Các hệ thống doanh nghiệp này được phân phối trên các máy tính khác nhau, có thể được sở hữu và quản lý bởi các công ty khác nhau.

Kiến trúc phần mềm là quan trọng ... Bởi vì nó ảnh hưởng đến hiệu suất, độ mạnh, khả năng phân phối và khả năng bảo trì của một hệ thống

## 10.2 Ưu điểm của kiến ​​trúc rõ ràng

Thông tin liên lạc của bên liên quan

* Kiến trúc là một bản trình bày cao cấp của hệ thống có thể được sử dụng làm trọng tâm để thảo luận bởi một loạt các bên liên quan khác nhau

Phân tích hệ thống

* Quyết định thiết kế kiến ​​trúc có ảnh hưởng sâu sắc đến việc hệ thống có thể đáp ứng các yêu cầu quan trọng như hiệu suất, độ tin cậy và khả năng bảo trì hay không.

Tái sử dụng quy mô lớn

* Mô hình kiến ​​trúc hệ thống là một mô tả nhỏ gọn, dễ quản lý về cách thức tổ chức hệ thống và cách các thành phần tương thích.
* Kiến trúc hệ thống thường giống nhau đối với các hệ thống có yêu cầu tương tự và do đó có thể hỗ trợ tái sử dụng phần mềm quy mô lớn

## 10.3 Đại diện kiến ​​trúc

* Sơ đồ khối đơn giản, không chính thức cho thấy các thực thể và các mối quan hệ là phương pháp thường được sử dụng nhất để ghi lại các kiến ​​trúc phần mềm.
* Nhưng những điều này đã bị chỉ trích vì chúng thiếu ngữ nghĩa, không hiển thị các loại mối quan hệ giữa các thực thể cũng như các đặc tính nhìn thấy được của các thực thể trong kiến ​​trúc.
* Phụ thuộc vào việc sử dụng các mô hình kiến ​​trúc. Các yêu cầu cho ngữ nghĩa mô hình phụ thuộc vào cách các mô hình được sử dụng.

## 10.4 Biểu đồ hộp & đường

* Rất trừu tượng - chúng không thể hiện bản chất của các mối quan hệ thành phần cũng như các thuộc tính hiển thị bên ngoài của các hệ thống phụ.
* Tuy nhiên, hữu ích cho giao tiếp với các bên liên quan và lập kế hoạch dự án.

## 10.5 Sử dụng mô hình kiến ​​trúc

Như một cách để tạo điều kiện thảo luận về thiết kế hệ thống

* Một quan điểm kiến ​​trúc cấp cao của một hệ thống là hữu ích cho việc giao tiếp với các bên liên quan hệ thống và lập kế hoạch dự án bởi vì nó không lộn xộn với các chi tiết. Các bên liên quan có thể liên quan đến nó và hiểu một cái nhìn trừu tượng của hệ thống. Sau đó, họ có thể thảo luận về toàn bộ hệ thống mà không bị nhầm lẫn chi tiết.

Như một cách để viết tư liệu một kiến ​​trúc đã được thiết kế

* Mục tiêu ở đây là tạo ra một mô hình hệ thống hoàn chỉnh cho thấy các thành phần khác nhau trong một hệ thống, các giao diện và các kết nối của chúng.

## 10.6 Quyết định thiết kế kiến ​​trúc

* Thiết kế kiến ​​trúc là một quá trình sáng tạo nơi bạn thiết kế một tổ chức hệ thống sẽ đáp ứng các yêu cầu chức năng và phi chức năng của một hệ thống
* Bởi vì nó là một quá trình sáng tạo, các hoạt động trong quá trình này phụ thuộc vào loại hệ thống đang được phát triển, nền tảng và kinh nghiệm của kiến ​​trúc sư hệ thống và các yêu cầu cụ thể cho hệ thống.
* Do đó, hữu ích khi nghĩ đến thiết kế kiến ​​trúc là một loạt các quyết định được đưa ra chứ không phải là một chuỗi các hoạt động
* Tuy nhiên, một số quyết định phổ biến kéo dài tất cả các quy trình thiết kế và các quyết định này ảnh hưởng đến các đặc điểm phi chức năng của hệ thống.
* Trong quá trình thiết kế kiến ​​trúc, các kiến ​​trúc sư hệ thống phải thực hiện một số quyết định về cấu trúc ảnh hưởng sâu sắc đến hệ thống và quá trình phát triển của nó.

***Dựa trên kiến ​​thức và kinh nghiệm của họ, họ phải xem xét những câu hỏi cơ bản sau đây về hệ thống….***

* Có kiến ​​trúc ứng dụng chung nào có thể được sử dụng không?
* Hệ thống sẽ được phân phối như thế nào?
* Kiểu kiến ​​trúc nào phù hợp?
* Cách tiếp cận nào sẽ được sử dụng để cấu trúc hệ thống?
* Làm thế nào hệ thống sẽ được phân tích thành các mô-đun?
* Chiến lược kiểm soát nào nên được sử dụng?
* Thiết kế kiến ​​trúc sẽ được đánh giá như thế nào?
* Kiến trúc nên được ghi chép như thế nào?

## 10.7 Kiến trúc tái sử dụng

* Các hệ thống trong cùng một miền thường có kiến ​​trúc tương tự phản ánh khái niệm miền.
* Các dòng sản phẩm ứng dụng được xây dựng xung quanh một kiến ​​trúc cốt lõi với các biến thể đáp ứng các yêu cầu cụ thể của khách hàng.
* Kiến trúc của một hệ thống có thể được thiết kế xung quanh một trong nhiều kiểu kiến ​​trúc hoặc "kiểu" hơn.
  + Chúng nắm bắt được bản chất của một kiến ​​trúc và có thể được khởi tạo theo nhiều cách khác nhau.

## 10.8 Kiến trúc & đặc điểm hệ thống

Do mối quan hệ chặt chẽ giữa các yêu cầu phi chức năng và kiến ​​trúc phần mềm, kiểu kiến ​​trúc và cấu trúc cụ thể mà bạn chọn cho một hệ thống phải phụ thuộc vào các yêu cầu hệ thống phi chức năng

Hiệu suất

* Bản địa hóa các hoạt động quan trọng và giảm thiểu thông tin liên lạc. Sử dụng các thành phần lớn hơn là hạt mịn.

Bảo vệ

* Sử dụng kiến ​​trúc lớp với các tài sản quan trọng trong các lớp bên trong.

An toàn

* Bản địa hóa các tính năng quan trọng về an toàn trong một số ít các hệ thống phụ.

khả dụng

* Bao gồm các thành phần dự phòng và cơ chế cho khả năng chịu lỗi.

Duy trì

* Sử dụng hạt mịn, thành phần thay thế.

**1. Hiệu suất**

* Nếu hiệu suất là một yêu cầu quan trọng, kiến ​​trúc phải được thiết kế để bản địa hóa các hoạt động quan trọng trong một số lượng nhỏ các thành phần, với các thành phần này được triển khai trên cùng một máy tính thay vì được phân phối trên mạng.
* Điều này có thể có nghĩa là sử dụng một vài thành phần tương đối lớn hơn là các thành phần nhỏ, hạt mịn, làm giảm số lượng các thành phần truyền thông. Bạn cũng có thể xem xét các tổ chức hệ thống thời gian chạy cho phép hệ thống được sao chép và thực hiện trên các bộ vi xử lý khác nhau

**2. Bảo mật**

Nếu bảo mật là một yêu cầu quan trọng, nên sử dụng cấu trúc phân lớp cho kiến ​​trúc, với các tài sản quan trọng nhất được bảo vệ trong các lớp trong cùng, với mức độ xác thực bảo mật cao được áp dụng cho các lớp này

**3. An toàn**

* Nếu an toàn là một yêu cầu quan trọng, kiến ​​trúc phải được thiết kế sao cho tất cả các hoạt động liên quan đến an toàn đều nằm trong một thành phần đơn lẻ hoặc trong một số lượng nhỏ các thành phần.
* Điều này làm giảm chi phí và các vấn đề về xác nhận an toàn và làm cho nó có thể cung cấp các hệ thống bảo vệ liên quan có thể tắt hệ thống một cách an toàn trong trường hợp thất bại.

## 10.9 Quan điểm kiến ​​trúc

* Quan điểm hoặc quan điểm nào hữu ích khi thiết kế và ghi lại kiến ​​trúc của hệ thống?
* Ký hiệu nào nên được sử dụng để mô tả các mô hình kiến ​​trúc?
* Mỗi mô hình kiến ​​trúc chỉ hiển thị một khung nhìn hoặc phối cảnh của hệ thống.
  + Nó có thể cho thấy cách một hệ thống bị phân tách thành các mô-đun, cách các quy trình thời gian chạy tương tác hoặc các cách khác nhau trong đó các thành phần hệ thống được phân phối trên một mạng. Đối với cả thiết kế và tài liệu, bạn thường cần trình bày nhiều quan điểm của kiến ​​trúc phần mềm.

## 10.10 4 + 1 mô hình xem kiến ​​trúc phần mềm

* Một khung nhìn lô-gic, cho thấy các trừu tượng chính trong hệ thống như các đối tượng hoặc các lớp đối tượng.
* Một khung nhìn quy trình, trong đó cho thấy, tại thời gian chạy, hệ thống bao gồm các quá trình tương tác.
* Một khung nhìn phát triển, cho thấy cách phần mềm bị phân hủy để phát triển.
* Một khung nhìn vật lý, hiển thị phần cứng hệ thống và cách các thành phần phần mềm được phân phối trên các bộ xử lý trong hệ thống.
* Liên quan đến các trường hợp sử dụng hoặc kịch bản (+1)

## 10.11 Mẫu kiến ​​trúc

* Các mẫu là một phương tiện đại diện, chia sẻ và sử dụng lại kiến ​​thức.
* Một mô hình kiến ​​trúc là một mô tả cách điệu về thực hành thiết kế tốt, đã được thử nghiệm và thử nghiệm trong các môi trường khác nhau.
* Các mẫu nên bao gồm thông tin về thời điểm chúng và khi nào chúng không hữu ích.
* Các mẫu có thể được biểu diễn bằng cách sử dụng các mô tả dạng bảng và đồ họa.

## 10.12 Mẫu Model-View-Controller (MVC)

Sự miêu tả

* Phân tách bản trình bày và tương tác từ dữ liệu hệ thống. Hệ thống được cấu trúc thành ba thành phần logic tương tác với nhau. Thành phần Model quản lý dữ liệu hệ thống và các hoạt động liên quan trên dữ liệu đó. Thành phần Chế độ xem xác định và quản lý cách dữ liệu được hiển thị cho người dùng. Thành phần Bộ điều khiển quản lý tương tác của người dùng (ví dụ: các lần nhấn phím, các lần nhấp chuột, v.v.) và chuyển các tương tác này đến Chế độ xem và Mô hình.

Thí dụ

* Hình 6.4 cho thấy kiến ​​trúc của một hệ thống ứng dụng dựa trên web được tổ chức bằng cách sử dụng mẫu MVC.

Khi đã sử dụng

* Được sử dụng khi có nhiều cách để xem và tương tác với dữ liệu. Cũng được sử dụng khi các yêu cầu tương lai cho tương tác và trình bày dữ liệu chưa được biết.

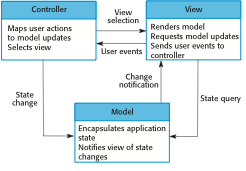
Ưu điểm

* Cho phép dữ liệu thay đổi độc lập với biểu diễn của nó và ngược lại. Hỗ trợ trình bày cùng một dữ liệu theo các cách khác nhau với các thay đổi được thực hiện trong một biểu diễn được hiển thị trong tất cả chúng.

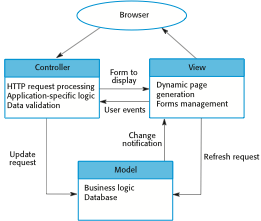
Nhược điểm

* Có thể liên quan đến mã phức tạp và mã bổ sung khi mô hình dữ liệu và tương tác rất đơn giản.

## 10.13 Tổ chức Model-View-Controller



## 10.14 Kiến trúc ứng dụng web sử dụng mẫu MVC



## 10.15 Kiến trúc lớp

* Được sử dụng để mô hình hóa giao diện của các hệ thống phụ.
* Tổ chức hệ thống thành một tập hợp các lớp (hoặc các máy trừu tượng), mỗi hệ thống cung cấp một tập hợp các dịch vụ.
* Hỗ trợ sự phát triển gia tăng của các hệ thống con trong các lớp khác nhau. Khi một giao diện lớp thay đổi, chỉ có lớp liền kề bị ảnh hưởng.
* Tuy nhiên, thường là nhân tạo để cấu trúc hệ thống theo cách này.

## 10.16 Mô hình kiến ​​trúc Layered

Sự miêu tả

* Tổ chức hệ thống thành các lớp với chức năng liên quan được liên kết với mỗi lớp. Một lớp cung cấp dịch vụ cho lớp ở trên nó để các lớp mức thấp nhất đại diện cho các dịch vụ cốt lõi có khả năng được sử dụng trong toàn bộ hệ thống.

Thí dụ

* Một mô hình phân lớp của một hệ thống để chia sẻ các tài liệu bản quyền được lưu giữ trong các thư viện khác nhau, như trong Hình 6.7.

Khi đã sử dụng

* Được sử dụng khi xây dựng các cơ sở mới trên các hệ thống hiện có; khi sự phát triển được lan truyền qua nhiều nhóm với trách nhiệm của mỗi nhóm đối với một lớp chức năng; khi có yêu cầu về bảo mật đa cấp.

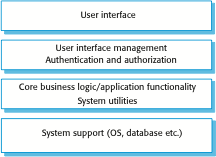
Ưu điểm

* Cho phép thay thế toàn bộ các lớp, miễn là giao diện được duy trì. Các cơ sở dự phòng (ví dụ: xác thực) có thể được cung cấp trong mỗi lớp để tăng tính tin cậy của hệ thống.

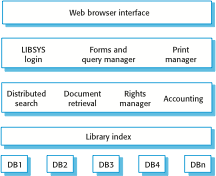
Nhược điểm

* Trong thực tế, việc cung cấp sự tách biệt giữa các lớp thường khó khăn và một lớp cao cấp có thể phải tương tác trực tiếp với các lớp cấp thấp hơn là thông qua lớp ngay bên dưới nó. Hiệu suất có thể là một vấn đề vì nhiều mức độ giải thích của một yêu cầu dịch vụ vì nó được xử lý ở mỗi lớp.

## 10.17 Một kiến ​​trúc lớp chung



## 10.18 Kiến trúc của hệ thống LIBSYS



**Những điểm chính**

* Kiến trúc phần mềm là mô tả về cách thức tổ chức hệ thống phần mềm.
* Quyết định thiết kế kiến ​​trúc bao gồm các quyết định về loại ứng dụng, sự phân bố của hệ thống, kiểu kiến ​​trúc được sử dụng.
* Các kiến ​​trúc có thể được ghi lại từ một số quan điểm hoặc quan điểm khác nhau như một khung nhìn khái niệm, một khung nhìn logic, một khung nhìn quy trình và một khung nhìn phát triển.
* Các mẫu kiến ​​trúc là một phương tiện tái sử dụng kiến ​​thức về các kiến ​​trúc hệ thống chung. Chúng mô tả kiến ​​trúc, giải thích khi nào nó có thể được sử dụng và mô tả những ưu điểm và nhược điểm của nó.

## 10.19 Kiến trúc kho lưu trữ

* Các hệ thống phụ phải trao đổi dữ liệu. Điều này có thể được thực hiện theo hai cách:
  + Dữ liệu được chia sẻ được lưu trữ trong một cơ sở dữ liệu hoặc kho lưu trữ trung tâm và có thể được truy cập bởi tất cả các hệ thống con;
  + Mỗi hệ thống con duy trì cơ sở dữ liệu riêng của mình và truyền dữ liệu một cách rõ ràng cho các hệ thống con khác.
* Khi lượng lớn dữ liệu được chia sẻ, mô hình chia sẻ lưu trữ thường được sử dụng nhất, đây là cơ chế chia sẻ dữ liệu hiệu quả.

## 10.20 Mẫu Repository

Sự miêu tả

* Tất cả dữ liệu trong một hệ thống được quản lý trong một kho lưu trữ trung tâm có thể truy cập được tới tất cả các thành phần hệ thống. Các thành phần không tương tác trực tiếp, chỉ thông qua kho lưu trữ.

Thí dụ

* Hình 6.9 là một ví dụ về một IDE nơi các thành phần sử dụng một kho lưu trữ thông tin thiết kế hệ thống. Mỗi công cụ phần mềm tạo ra thông tin mà sau đó có sẵn để sử dụng bởi các công cụ khác.

Khi đã sử dụng

* Bạn nên sử dụng mẫu này khi bạn có một hệ thống trong đó khối lượng lớn thông tin được tạo ra phải được lưu trữ trong một thời gian dài. Bạn cũng có thể sử dụng nó trong các hệ thống theo hướng dữ liệu trong đó việc đưa dữ liệu vào kho lưu trữ kích hoạt một hành động hoặc công cụ.

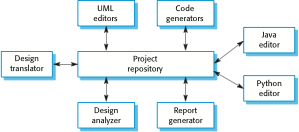
Ưu điểm

* Các thành phần có thể độc lập - chúng không cần biết về sự tồn tại của các thành phần khác. Các thay đổi được thực hiện bởi một thành phần có thể được truyền cho tất cả các thành phần. Tất cả dữ liệu có thể được quản lý một cách nhất quán (ví dụ: các bản sao lưu được thực hiện cùng một lúc) vì tất cả dữ liệu đều ở cùng một nơi.

Nhược điểm

* Kho lưu trữ là một điểm lỗi duy nhất nên các vấn đề trong kho lưu trữ ảnh hưởng đến toàn bộ hệ thống. Có thể không hiệu quả trong việc tổ chức tất cả các giao tiếp thông qua kho lưu trữ. Việc phân phối kho lưu trữ trên nhiều máy tính có thể khó khăn.

## 10.21 Kiến trúc kho lưu trữ cho một IDE



## 10.22 Kiến trúc máy khách-máy chủ

* Mô hình hệ thống phân tán cho thấy cách dữ liệu và xử lý được phân phối trên một loạt các thành phần.
* Có thể được triển khai trên một máy tính.
* Đặt các máy chủ độc lập cung cấp các dịch vụ cụ thể như in, quản lý dữ liệu, v.v.
* Đặt các khách hàng gọi đến các dịch vụ này.
* Mạng cho phép khách hàng truy cập máy chủ.

## 10.23 Mẫu Client-server

Description

* Trong kiến ​​trúc client-server, chức năng của hệ thống được tổ chức thành các dịch vụ, với mỗi dịch vụ được phân phối từ một máy chủ riêng biệt. Khách hàng là người dùng của các dịch vụ này và truy cập máy chủ để sử dụng chúng.

When used

* Được sử dụng khi dữ liệu trong cơ sở dữ liệu được chia sẻ phải được truy cập từ một loạt các vị trí. Bởi vì máy chủ có thể được nhân rộng, cũng có thể được sử dụng khi tải trên hệ thống là biến.

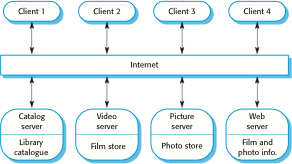
Advantages

* Ưu điểm chính của mô hình này là các máy chủ có thể được phân phối trên mạng. Chức năng chung (ví dụ: dịch vụ in) có thể có sẵn cho tất cả khách hàng và không cần phải được thực hiện bởi tất cả các dịch vụ.

Disadvantages

* Mỗi dịch vụ là một điểm thất bại nên dễ bị tấn công từ chối dịch vụ hoặc lỗi máy chủ. Hiệu suất có thể không thể đoán trước được vì nó phụ thuộc vào mạng cũng như hệ thống. Có thể là vấn đề quản lý nếu máy chủ được sở hữu bởi các tổ chức khác nhau.

## 10.24 Kiến trúc máy khách-máy chủ cho thư viện phim



**Kiến trúc ống & lọc**

* Chuyển đổi chức năng xử lý đầu vào của chúng để tạo ra kết quả đầu ra.
* Có thể được gọi là mô hình đường ống và bộ lọc (như trong vỏ UNIX).
* Các biến thể của phương pháp này rất phổ biến. Khi các biến đổi là tuần tự, đây là một mô hình tuần tự theo lô được sử dụng rộng rãi trong các hệ thống xử lý dữ liệu.
* Không thực sự phù hợp với các hệ thống tương tác.

**Mẫu ống và bộ lọc**

Description

* Việc xử lý dữ liệu trong một hệ thống được tổ chức sao cho mỗi thành phần xử lý (bộ lọc) là rời rạc và thực hiện một loại chuyển đổi dữ liệu. Các luồng dữ liệu (như trong một đường ống) từ một thành phần này đến thành phần khác để xử lý.

When used

* Thường được sử dụng trong các ứng dụng xử lý dữ liệu (cả hàng loạt và dựa trên giao dịch), trong đó các đầu vào được xử lý trong các giai đoạn riêng biệt để tạo ra các kết quả đầu ra liên quan.

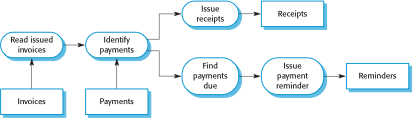
Advantages

* Dễ hiểu và hỗ trợ tái sử dụng chuyển đổi. Kiểu luồng công việc khớp với cấu trúc của nhiều quy trình nghiệp vụ. Sự tiến hóa bằng cách thêm các phép biến đổi là đơn giản. Có thể được thực hiện như một hệ thống tuần tự hoặc đồng thời.

Disadvantages

* Định dạng truyền dữ liệu phải được thỏa thuận giữa các phép chuyển đổi truyền thông. Mỗi phép biến đổi phải phân tích cú pháp đầu vào của nó và tách đầu ra của nó thành dạng đã đồng ý. Điều này làm tăng chi phí của hệ thống và có thể có nghĩa là không thể tái sử dụng các phép biến đổi chức năng sử dụng các cấu trúc dữ liệu không tương thích.

**Ví dụ về cấu trúc đường ống và bộ lọc**



## 10.25 Kiến trúc ứng dụng

* Các hệ thống ứng dụng được thiết kế để đáp ứng nhu cầu của tổ chức.
* Khi các doanh nghiệp có nhiều điểm chung, các hệ thống ứng dụng của họ cũng có xu hướng có một kiến ​​trúc chung phản ánh các yêu cầu ứng dụng.
* Kiến trúc ứng dụng chung là một kiến ​​trúc cho một loại hệ thống phần mềm có thể được cấu hình và điều chỉnh để tạo ra một hệ thống đáp ứng các yêu cầu cụ thể.

**Sử dụng kiến ​​trúc ứng dụng**

* Là điểm khởi đầu cho thiết kế kiến ​​trúc.
* Là một danh sách kiểm tra thiết kế. Như một cách để tổ chức công việc của nhóm phát triển.
* Là một phương tiện đánh giá các thành phần để tái sử dụng.
* Là một từ vựng để nói về các loại ứng dụng.

**Ví dụ về các loại ứng dụng**

Ứng dụng xử lý dữ liệu

* Các ứng dụng điều khiển dữ liệu xử lý dữ liệu theo lô mà không có sự can thiệp của người dùng rõ ràng trong quá trình xử lý.

Ứng dụng xử lý giao dịch

* Các ứng dụng tập trung vào dữ liệu xử lý các yêu cầu của người dùng và cập nhật thông tin trong cơ sở dữ liệu hệ thống.

Hệ thống xử lý sự kiện

* Các ứng dụng mà các hành động hệ thống phụ thuộc vào việc diễn giải sự kiện từ môi trường của hệ thống.

Hệ thống xử lý ngôn ngữ

* Các ứng dụng mà ý định của người dùng được chỉ định bằng ngôn ngữ chính thức được xử lý và diễn giải bởi hệ thống.

**Ví dụ về loại ứng dụng**

* Tập trung ở đây là về xử lý giao dịch và hệ thống xử lý ngôn ngữ.
* Hệ thống xử lý giao dịch
  + Hệ thống thương mại điện tử;
  + Hệ thống đặt chỗ.
* Hệ thống xử lý ngôn ngữ
  + Trình biên dịch;
  + Trình thông dịch lệnh.

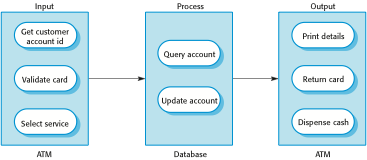
## 10.26 Hệ thống xử lý giao dịch

* Xử lý các yêu cầu của người dùng về thông tin từ cơ sở dữ liệu hoặc yêu cầu cập nhật cơ sở dữ liệu.
* Từ góc độ người dùng, giao dịch là:
  + Bất kỳ chuỗi hoạt động chặt chẽ nào thỏa mãn mục tiêu;
  + Ví dụ: - tìm thời gian chuyến bay từ Luân Đôn đến Paris.
* Người dùng thực hiện các yêu cầu không đồng bộ cho dịch vụ mà sau đó được xử lý bởi người quản lý giao dịch.

## 10.28 Cấu trúc của các ứng dụng xử lý giao dịch

6.14 TransactionProcSys.eps

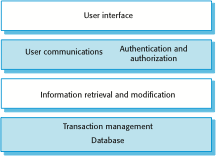
## 10.29 Kiến trúc phần mềm của một hệ thống ATM



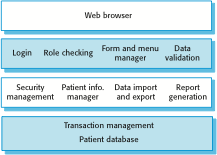
## 10.30 Kiến trúc hệ thống thông tin

* Các hệ thống thông tin có một kiến ​​trúc chung có thể được tổ chức như một kiến ​​trúc phân lớp.
* Đây là những hệ thống dựa trên giao dịch khi tương tác với các hệ thống này thường liên quan đến các giao dịch cơ sở dữ liệu.
* Các lớp bao gồm:
  + Giao diện người dùng
  + Liên lạc của người dùng
  + Thu hồi thông tin
  + Cơ sở dữ liệu hệ thống

## 10.31 Kiến trúc hệ thống thông tin lớp



## 10.32 Kiến trúc của MHC-PMS



## 10.33 Hệ thống thông tin dựa trên web

* Thông tin và các hệ thống quản lý tài nguyên hiện nay thường là các hệ thống dựa trên web, nơi các giao diện người dùng được thực hiện bằng cách sử dụng một trình duyệt web.
* Ví dụ, các hệ thống thương mại điện tử là các hệ thống quản lý tài nguyên dựa trên Internet chấp nhận các đơn đặt hàng điện tử cho hàng hóa hoặc dịch vụ và sau đó sắp xếp việc phân phối các hàng hoá hoặc dịch vụ này cho khách hàng.
* Trong hệ thống thương mại điện tử, lớp ứng dụng cụ thể bao gồm chức năng bổ sung hỗ trợ ‘giỏ hàng’ trong đó người dùng có thể đặt một số mặt hàng trong các giao dịch riêng biệt, sau đó thanh toán cho tất cả chúng trong cùng một giao dịch.

## 10.34 Triển khai máy chủ

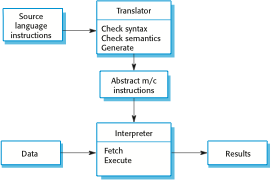
Các hệ thống này thường được triển khai dưới dạng máy chủ / kiến ​​trúc khách hàng nhiều tầng (được thảo luận trong Chương 18)

* Máy chủ web chịu trách nhiệm về tất cả các liên lạc của người dùng, với giao diện người dùng được triển khai bằng trình duyệt web;
* Máy chủ ứng dụng chịu trách nhiệm triển khai logic ứng dụng cụ thể cũng như các yêu cầu lưu trữ và truy xuất thông tin;
* Máy chủ cơ sở dữ liệu di chuyển thông tin đến và từ cơ sở dữ liệu và xử lý quản lý giao dịch.

## 10.35 Hệ thống xử lý ngôn ngữ

* Chấp nhận một ngôn ngữ tự nhiên hoặc nhân tạo làm đầu vào và tạo ra một số đại diện khác của ngôn ngữ đó.
* Có thể bao gồm một thông dịch viên để thực hiện các hướng dẫn bằng ngôn ngữ đang được xử lý.
* Được sử dụng trong các tình huống mà cách dễ nhất để giải quyết vấn đề là mô tả thuật toán hoặc mô tả dữ liệu hệ thống
  + Các công cụ Meta-case mô tả công cụ xử lý, các quy tắc phương thức, vv và tạo ra các công cụ.

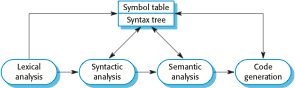
**Kiến trúc của một hệ thống xử lý ngôn ngữ**



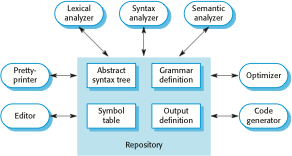
## 10.36 Thành phần trình biên dịch

* Một máy phân tích từ vựng, có các mã thông báo ngôn ngữ đầu vào và chuyển đổi chúng thành một biểu mẫu bên trong.
* Một bảng biểu tượng, chứa thông tin về tên của các thực thể (các biến, tên lớp, tên đối tượng, vv) được sử dụng trong văn bản đang được dịch.
* Trình phân tích cú pháp, kiểm tra cú pháp của ngôn ngữ đang được dịch.
* Một cây cú pháp, là một cấu trúc bên trong biểu diễn chương trình đang được biên dịch.
* Một trình phân tích ngữ nghĩa sử dụng thông tin từ cây cú pháp và bảng biểu tượng để kiểm tra tính chính xác ngữ nghĩa của văn bản ngôn ngữ nhập liệu.
* Một trình tạo mã mà ‘đi’ cây cú pháp và tạo ra mã máy trừu tượng.

## 10.37 Kiến trúc trình biên dịch ống và bộ lọc



## 10.38 Kiến trúc kho lưu trữ cho hệ thống xử lý ngôn ngữ



**Key points**

* Các mô hình kiến ​​trúc hệ thống ứng dụng giúp chúng ta hiểu và so sánh các ứng dụng, xác nhận các thiết kế hệ thống ứng dụng và đánh giá các thành phần có quy mô lớn để tái sử dụng.
* Hệ thống xử lý giao dịch là các hệ thống tương tác cho phép thông tin trong cơ sở dữ liệu được một số người dùng truy cập và sửa đổi từ xa.
* Hệ thống xử lý ngôn ngữ được sử dụng để dịch văn bản từ ngôn ngữ này sang ngôn ngữ khác và thực hiện các hướng dẫn được chỉ định bằng ngôn ngữ nhập liệu. Chúng bao gồm một dịch giả và một máy trừu tượng thực thi ngôn ngữ được tạo ra.

# Chapter 11: Kiểm thử phần mềm

**Chủ đề được đề cập**

* Thử nghiệm phát triển
* Hướng phát triển thử nghiệm
* Phát hành thử nghiệm
* Kiểm tra người dùng

## 11.1 Kiểm tra chương trình

* Thử nghiệm được thiết kế để cho thấy rằng một chương trình thực hiện những gì nó được dự định để làm và để khám phá các khuyết tật của chương trình trước khi nó được đưa vào sử dụng.
* Khi bạn kiểm tra phần mềm, bạn thực thi một chương trình bằng cách sử dụng dữ liệu nhân tạo.
* Bạn kiểm tra kết quả chạy thử để tìm lỗi, bất thường hoặc thông tin về các thuộc tính không hoạt động của chương trình.
* Có thể tiết lộ sự hiện diện của các lỗi KHÔNG vắng mặt của họ.
* Kiểm tra là một phần của quá trình xác minh và xác thực tổng quát hơn, bao gồm các kỹ thuật xác thực tĩnh.

## 11.2 Mục tiêu kiểm tra chương trình

Để minh họa cho nhà phát triển và khách hàng rằng phần mềm đáp ứng các yêu cầu của họ.

* Đối với phần mềm tùy chỉnh, điều này có nghĩa là phải có ít nhất một thử nghiệm cho mọi yêu cầu trong tài liệu yêu cầu. Đối với các sản phẩm phần mềm chung, nó có nghĩa là phải có các thử nghiệm cho tất cả các tính năng của hệ thống, cộng với sự kết hợp các tính năng này, sẽ được đưa vào bản phát hành sản phẩm.

Để khám phá các tình huống trong đó hành vi của phần mềm không chính xác, không mong muốn hoặc không tuân theo đặc điểm kỹ thuật của phần mềm.

* Kiểm tra khiếm khuyết có liên quan đến việc tạo ra các hành vi hệ thống không mong muốn như sự cố hệ thống, các tương tác không mong muốn với các hệ thống khác, tính toán sai và tham nhũng dữ liệu.

## 11.3 Kiểm tra xác thực và lỗi

Mục tiêu đầu tiên dẫn đến kiểm tra xác thực

* Bạn mong đợi hệ thống hoạt động chính xác bằng cách sử dụng một tập hợp các trường hợp thử nghiệm nhất định phản ánh việc sử dụng dự kiến ​​của hệ thống.

Mục tiêu thứ hai dẫn đến kiểm tra lỗi

* Các trường hợp thử nghiệm được thiết kế để lộ các khuyết tật. Các trường hợp thử nghiệm trong kiểm tra lỗi có thể được cố ý che khuất và không cần phải phản ánh cách hệ thống được sử dụng bình thường.

## 11.4 Mục tiêu quy trình thử nghiệm

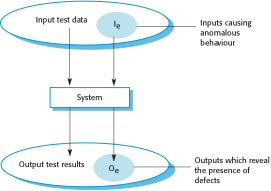
Kiểm tra xác thực

* Để minh họa cho nhà phát triển và khách hàng hệ thống rằng phần mềm đáp ứng các yêu cầu của nó
* Một thử nghiệm thành công cho thấy rằng hệ thống hoạt động như dự định.

Kiểm tra lỗi

* Để phát hiện lỗi hoặc lỗi trong phần mềm mà hành vi của nó không chính xác hoặc không phù hợp với đặc điểm kỹ thuật của nó
* Một thử nghiệm thành công là một thử nghiệm mà làm cho hệ thống hoạt động không chính xác và để lộ ra một khiếm khuyết trong hệ thống.

## 11.5 Một mô hình đầu vào-đầu ra của kiểm thử chương trình



## 11.6 Xác minh và xác thực

Xác minh: "Chúng tôi đang xây dựng sản phẩm đúng".

* Phần mềm phải phù hợp với đặc điểm kỹ thuật của nó.

Xác nhận: "Chúng tôi đang xây dựng đúng sản phẩm".

* Phần mềm phải làm những gì người dùng thực sự yêu cầu.

## 11.7 Tự tin V & V

Mục tiêu của V & V là thiết lập sự tự tin rằng hệ thống là 'phù hợp với mục đích'.

Phụ thuộc vào mục đích của hệ thống, kỳ vọng của người dùng và môi trường tiếp thị

* Mục đích phần mềm
  + Mức độ tin cậy phụ thuộc vào mức độ quan trọng của phần mềm đối với một tổ chức.
* Kỳ vọng của người dùng
  + Người dùng có thể có kỳ vọng thấp của một số loại phần mềm nhất định.
  + Môi trường tiếp thị
* Đưa sản phẩm ra thị trường sớm có thể quan trọng hơn việc tìm ra khuyết tật trong chương trình.

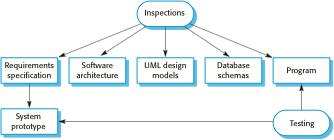
## 11.8 Kiểm tra & Kiểm tra

Kiểm tra phần mềm Quan tâm đến việc phân tích biểu diễn hệ thống tĩnh để phát hiện sự cố (xác minh tĩnh)

* Có thể được bổ sung bằng phân tích mã và tài liệu dựa trên công cụ.

Kiểm tra phần mềm Quan tâm đến việc thực hiện và quan sát hành vi của sản phẩm (xác minh động)

* Hệ thống được thực hiện với dữ liệu thử nghiệm và hành vi hoạt động của nó được quan sát.



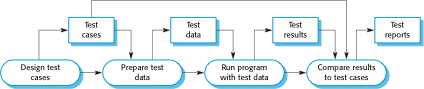
## 11.9 Kiểm tra phần mềm

* Những người này liên quan đến việc kiểm tra sự biểu diễn nguồn với mục đích phát hiện dị thường và khuyết tật.
* Việc kiểm tra không yêu cầu thực thi hệ thống để có thể được sử dụng trước khi triển khai.
* Chúng có thể được áp dụng cho bất kỳ đại diện nào của hệ thống (yêu cầu, thiết kế, dữ liệu cấu hình, dữ liệu thử nghiệm, v.v.).
* Chúng đã được chứng minh là một kỹ thuật hiệu quả để phát hiện lỗi chương trình.

### 11.9.1Ưu điểm của kiểm tra

* Trong quá trình thử nghiệm, lỗi có thể che giấu (ẩn) các lỗi khác. Vì kiểm tra là một quá trình tĩnh nên bạn không phải lo lắng về tương tác giữa các lỗi.
* Các phiên bản không đầy đủ của một hệ thống có thể được kiểm tra mà không phải trả thêm chi phí. Nếu một chương trình không đầy đủ, thì bạn cần phát triển các bộ kiểm tra chuyên dụng để kiểm tra các bộ phận có sẵn.
* Cũng như tìm kiếm các khiếm khuyết của chương trình, việc kiểm tra cũng có thể xem xét các thuộc tính chất lượng rộng hơn của một chương trình, chẳng hạn như tuân thủ các tiêu chuẩn, tính di động và khả năng bảo trì.
* Kiểm tra và thử nghiệm là bổ sung và không phản đối các kỹ thuật xác minh.
* Cả hai nên được sử dụng trong quá trình V & V.
* Kiểm tra có thể kiểm tra sự phù hợp với đặc điểm kỹ thuật nhưng không tuân thủ các yêu cầu thực tế của khách hàng.
* Kiểm tra không thể kiểm tra các đặc điểm phi chức năng như hiệu suất, khả năng sử dụng, v.v.

## 11.10 Một mô hình của quá trình kiểm thử phần mềm



### 11.10.1 Các giai đoạn thử nghiệm

* Thử nghiệm phát triển, nơi hệ thống được kiểm tra trong quá trình phát triển để phát hiện lỗi và lỗi.
* Phát hành thử nghiệm, trong đó một nhóm thử nghiệm riêng biệt kiểm tra một phiên bản hoàn chỉnh của hệ thống trước khi nó được phát hành cho người dùng.
* Kiểm tra người dùng, nơi người dùng hoặc người dùng tiềm năng của hệ thống kiểm tra hệ thống trong môi trường của riêng họ.

## 11.11 Thử nghiệm phát triển

Thử nghiệm phát triển bao gồm tất cả các hoạt động thử nghiệm được thực hiện bởi nhóm phát triển hệ thống.

* Kiểm tra đơn vị, nơi các đơn vị chương trình riêng lẻ hoặc các lớp đối tượng được kiểm tra. Thử nghiệm đơn vị nên tập trung vào việc kiểm tra chức năng của các đối tượng hoặc phương pháp.
* Kiểm tra thành phần, trong đó một số đơn vị riêng lẻ được tích hợp để tạo thành phần hỗn hợp. Kiểm tra thành phần nên tập trung vào việc kiểm tra các giao diện thành phần.
* Kiểm tra hệ thống, trong đó một số hoặc tất cả các thành phần trong một hệ thống được tích hợp và hệ thống được kiểm tra toàn bộ. Kiểm tra hệ thống nên tập trung vào thử nghiệm các tương tác thành phần.

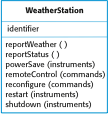
## 11.12 Kiểm tra đơn vị

* Thử nghiệm đơn vị là quá trình thử nghiệm từng thành phần riêng biệt.
* Đó là một quá trình kiểm tra lỗi.
* Các đơn vị có thể là:
  + Các hàm hoặc phương thức riêng lẻ trong một đối tượng
  + Các lớp đối tượng với một số thuộc tính và phương thức
  + Các thành phần tổng hợp với các giao diện được xác định được sử dụng để truy cập chức năng của chúng.

## 11.13 Kiểm tra lớp đối tượng

* Phạm vi kiểm tra hoàn chỉnh của một lớp học liên quan đến
  + Kiểm tra tất cả các hoạt động liên kết với một đối tượng
  + Thiết lập và thẩm vấn tất cả các thuộc tính đối tượng
  + Tập thể dục đối tượng ở tất cả các trạng thái có thể.
* Thừa kế làm cho việc kiểm tra lớp đối tượng khó khăn hơn vì thông tin được kiểm tra không được bản địa hóa.

## 11.14 Giao diện đối tượng trạm thời tiết

****

### 11.14.1 Kiểm tra trạm thời tiết

* Cần xác định các trường hợp kiểm thử cho reportWeather, hiệu chỉnh, kiểm tra, khởi động và tắt máy.
* Sử dụng một mô hình nhà nước, xác định trình tự của các quá trình chuyển đổi trạng thái được kiểm tra và các chuỗi sự kiện gây ra các quá trình chuyển đổi này
* Ví dụ:
  + Shutdown -> Running-> Shutdown
  + Cấu hình-> Chạy-> Thử nghiệm -> Đang truyền -> Đang chạy
  + Chạy-> Thu thập-> Đang chạy-> Tóm tắt -> Đang truyền -> Đang chạy

## 11.15 Thử nghiệm tự động

* Bất cứ khi nào có thể, kiểm tra đơn vị phải được tự động hóa để kiểm tra được chạy và kiểm tra mà không cần sự can thiệp thủ công.
* Trong thử nghiệm đơn vị tự động, bạn sử dụng một khung công tác tự động hóa thử nghiệm (như JUnit) để viết và chạy các thử nghiệm chương trình của bạn.
* Các khung kiểm thử đơn vị cung cấp các lớp thử nghiệm chung mà bạn mở rộng để tạo các trường hợp thử nghiệm cụ thể. Sau đó, họ có thể chạy tất cả các thử nghiệm mà bạn đã triển khai và báo cáo, thường xuyên thông qua một số GUI, về sự thành công của các thử nghiệm khác.

### 11.15.1 Các thành phần kiểm tra tự động

* Một phần thiết lập, nơi bạn khởi tạo hệ thống với trường hợp thử nghiệm, cụ thể là các đầu vào và đầu ra dự kiến.
* Một phần cuộc gọi, nơi bạn gọi đối tượng hoặc phương pháp được kiểm tra.
* Một phần khẳng định nơi bạn so sánh kết quả của cuộc gọi với kết quả mong đợi. Nếu xác nhận đánh giá là đúng, thử nghiệm đã thành công nếu sai, sau đó nó đã thất bại.

## 11.16 Đơn vị kiểm tra hiệu quả

* Các trường hợp thử nghiệm sẽ cho thấy rằng, khi được sử dụng như mong đợi, thành phần mà bạn đang thử nghiệm thực hiện những gì nó được cho là phải làm.
* Nếu có lỗi trong thành phần, chúng sẽ được tiết lộ bởi các trường hợp thử nghiệm.
* Điều này dẫn đến 2 loại trường hợp kiểm thử đơn vị:
  + Việc đầu tiên trong số này nên phản ánh hoạt động bình thường của một chương trình và sẽ cho thấy rằng các thành phần hoạt động như mong đợi.
  + Loại trường hợp thử nghiệm khác phải dựa trên trải nghiệm thử nghiệm về những vấn đề phổ biến phát sinh. Nó nên sử dụng đầu vào bất thường để kiểm tra xem chúng được xử lý đúng cách và không làm hỏng thành phần.

## 11.17 Chiến lược thử nghiệm

Thử nghiệm phân vùng, nơi bạn xác định các nhóm đầu vào có đặc điểm chung và cần được xử lý theo cùng một cách.

* Bạn nên chọn các bài kiểm tra từ trong mỗi nhóm này.

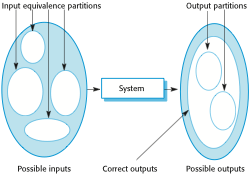
Kiểm tra dựa trên hướng dẫn, nơi bạn sử dụng nguyên tắc kiểm tra để chọn các trường hợp kiểm tra.

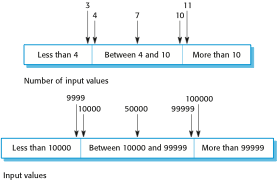
* Những hướng dẫn này phản ánh kinh nghiệm trước đây về các loại lỗi mà các lập trình viên thường thực hiện khi phát triển các thành phần.

## 11.18 Kiểm tra phân vùng

* Dữ liệu đầu vào và kết quả đầu ra thường rơi vào các lớp khác nhau, nơi tất cả các thành viên của một lớp có liên quan.
* Mỗi lớp này là một phân vùng tương đương hoặc miền nơi chương trình hoạt động theo cách tương đương cho mỗi thành viên của lớp.
* Các trường hợp kiểm tra nên được chọn từ mỗi phân vùng.

**Phân vùng tương đương**





## 11.19 Nguyên tắc kiểm tra (chuỗi)

* Kiểm tra phần mềm với các chuỗi chỉ có một giá trị duy nhất.
* Sử dụng chuỗi các kích thước khác nhau trong các thử nghiệm khác nhau.
* Kiểm tra nguồn gốc để các phần tử đầu tiên, trung bình và cuối cùng của chuỗi được truy cập.
* Thử nghiệm với các chuỗi có độ dài bằng không.

### 11.19.1 Nguyên tắc kiểm tra chung

* Chọn đầu vào buộc hệ thống tạo tất cả thông báo lỗi
* Thiết kế đầu vào gây ra bộ đệm đầu vào tràn
* Lặp lại cùng một đầu vào hoặc một loạt các đầu vào nhiều lần
* Buộc tạo ra các đầu ra không hợp lệ
* Lực lượng kết quả tính toán quá lớn hoặc quá nhỏ.

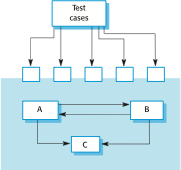
**Key points**

* Thử nghiệm chỉ có thể hiển thị sự hiện diện của các lỗi trong một chương trình. Nó không thể chứng minh rằng không có lỗi còn lại.
* Kiểm thử phát triển là trách nhiệm của nhóm phát triển phần mềm. Một nhóm riêng biệt phải chịu trách nhiệm thử nghiệm một hệ thống trước khi nó được phát hành cho khách hàng.
* Thử nghiệm phát triển bao gồm kiểm tra đơn vị, trong đó bạn kiểm tra từng đối tượng và phương pháp thử nghiệm thành phần trong đó bạn kiểm tra các nhóm đối tượng liên quan và kiểm tra hệ thống, trong đó bạn thử nghiệm một phần hoặc toàn bộ hệ thống.

## 11.20 Kiểm tra thành phần

* Các thành phần phần mềm thường là các thành phần tổng hợp được tạo thành từ một số đối tượng tương tác.
  + Ví dụ, trong hệ thống trạm thời tiết, thành phần cấu hình lại bao gồm các đối tượng xử lý từng khía cạnh của cấu hình lại.
* Bạn truy cập chức năng của các đối tượng này thông qua giao diện thành phần được xác định.
* Do đó, thử nghiệm các thành phần tổng hợp nên tập trung vào việc hiển thị rằng giao diện thành phần hoạt động theo đặc điểm kỹ thuật của nó.
  + Bạn có thể giả định rằng các bài kiểm tra đơn vị trên các đối tượng riêng lẻ trong thành phần đã được hoàn thành.

## 11.21 Kiểm tra giao diện

****

* Mục tiêu là để phát hiện lỗi do lỗi giao diện hoặc giả định không hợp lệ về giao diện.
* Loại giao diện
  + Giao diện tham số Dữ liệu truyền từ một phương thức hoặc thủ tục này sang phương thức khác.
  + Giao diện bộ nhớ chia sẻ Khối bộ nhớ được chia sẻ giữa các thủ tục hoặc chức năng.
  + Các giao diện thủ tục Hệ thống con gói gọn một tập hợp các thủ tục được gọi bởi các hệ thống con khác.
  + Các giao diện truyền tin nhắn Các hệ thống con yêu cầu các dịch vụ từ các hệ thống con khác

## 11.22 Lỗi giao diện

Giao diện sử dụng sai

* Thành phần gọi điện gọi một thành phần khác và gây ra lỗi khi sử dụng giao diện của nó, ví dụ: thông số theo thứ tự sai.

Giao diện hiểu lầm

* Một thành phần gọi điện thoại nhúng các giả định về hành vi của thành phần được gọi là không chính xác.

Lỗi thời gian

* Thành phần gọi và gọi hoạt động ở các tốc độ khác nhau và thông tin lỗi thời được truy cập.

## 11.23 Nguyên tắc kiểm tra giao diện

* Thiết kế kiểm tra để các tham số cho một thủ tục được gọi là ở các đầu cực của phạm vi của chúng.
* Luôn kiểm tra các tham số con trỏ với con trỏ null.
* Kiểm tra thiết kế khiến cho thành phần thất bại.
* Sử dụng thử nghiệm ứng suất trong các hệ thống truyền tin nhắn.
* Trong các hệ thống bộ nhớ dùng chung, thay đổi thứ tự các thành phần được kích hoạt.

## 11.24 Thử nghiệm hệ thống

* Kiểm tra hệ thống trong quá trình phát triển bao gồm tích hợp các thành phần để tạo ra một phiên bản của hệ thống và sau đó kiểm tra hệ thống tích hợp.
* Trọng tâm trong kiểm thử hệ thống là kiểm tra tương tác giữa các thành phần.
* Kiểm tra hệ thống kiểm tra xem các thành phần có tương thích hay không, tương tác một cách chính xác và chuyển đúng dữ liệu vào đúng thời điểm trên các giao diện của chúng.
* Kiểm tra hệ thống kiểm tra hành vi nổi lên của một hệ thống.

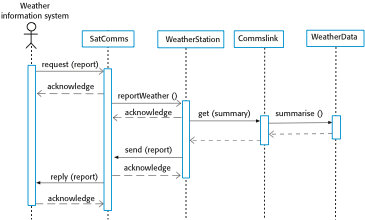
## 11.25 Kiểm tra hệ thống và thành phần

* Trong quá trình kiểm tra hệ thống, các thành phần có thể tái sử dụng đã được phát triển riêng biệt và các hệ thống có thể được tích hợp với các thành phần mới được phát triển. Hệ thống hoàn chỉnh sau đó được kiểm tra.
* Các thành phần được phát triển bởi các thành viên nhóm khác nhau hoặc các nhóm phụ có thể được tích hợp ở giai đoạn này. Kiểm tra hệ thống là một tập thể chứ không phải là một quy trình riêng lẻ.
  + Ở một số công ty, việc kiểm tra hệ thống có thể liên quan đến một nhóm thử nghiệm riêng biệt mà không có sự tham gia của các nhà thiết kế và lập trình viên.

## 11.27 Thử nghiệm Usecase

* Các trường hợp sử dụng được phát triển để xác định các tương tác hệ thống có thể được sử dụng làm cơ sở cho việc kiểm tra hệ thống.
* Mỗi trường hợp sử dụng thường liên quan đến một số thành phần hệ thống để kiểm tra trường hợp sử dụng buộc các tương tác này xảy ra.
* Các biểu đồ trình tự liên kết với ca sử dụng ghi lại các thành phần và các tương tác đang được thử nghiệm.

## 11.28 Thu thập biểu đồ trình tự dữ liệu thời tiết



## 11.29 Chính sách thử nghiệm

Thử nghiệm hệ thống thông gió là không thể vì vậy các chính sách thử nghiệm xác định phạm vi kiểm tra hệ thống yêu cầu có thể được phát triển.

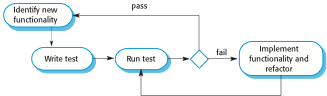
Ví dụ về chính sách thử nghiệm:

* Tất cả các chức năng hệ thống được truy cập thông qua các menu sẽ được kiểm tra.
* Việc kết hợp các hàm (ví dụ: định dạng văn bản) được truy cập thông qua cùng một menu phải được kiểm tra.
* Trong trường hợp đầu vào của người dùng được cung cấp, tất cả các hàm phải được kiểm tra với cả đầu vào chính xác và không chính xác.

## 11.30 Hướng phát triển thử nghiệm

* Phát triển theo hướng thử nghiệm (TDD) là một cách tiếp cận để phát triển chương trình, trong đó bạn liên tục kiểm tra và phát triển mã.
* Các bài kiểm tra được viết trước khi viết mã và ‘qua’ các bài kiểm tra là trình điều khiển phát triển quan trọng.
* Bạn phát triển mã tăng dần, cùng với một thử nghiệm cho số gia tăng đó. Bạn không chuyển sang bước tăng tiếp theo cho đến khi mã mà bạn đã phát triển vượt qua thử nghiệm của nó.
* TDD được giới thiệu như một phần của các phương thức nhanh như Lập trình cực đoan. Tuy nhiên, nó cũng có thể được sử dụng trong các quy trình phát triển theo kế hoạch.

**Hướng phát triển thử nghiệm**



## 11.31 TDD Process Các hoạt động

* Bắt đầu bằng cách xác định sự gia tăng của chức năng được yêu cầu. Điều này thường sẽ nhỏ và có thể triển khai trong một vài dòng mã.
* Viết một bài kiểm tra cho chức năng này và thực hiện điều này như một bài kiểm tra tự động.
* Chạy thử nghiệm, cùng với tất cả các thử nghiệm khác đã được triển khai. Ban đầu, bạn chưa triển khai chức năng để thử nghiệm mới sẽ thất bại.
* Thực hiện chức năng và chạy lại thử nghiệm.
* Khi tất cả các thử nghiệm chạy thành công, bạn chuyển sang triển khai phần tiếp theo của chức năng.

### 11.31.1 Lợi ích của phát triển theo hướng thử nghiệm

Mã số bảo hiểm

* Mỗi phân đoạn mã mà bạn viết có ít nhất một thử nghiệm được liên kết để tất cả các mã được viết có ít nhất một thử nghiệm.

Kiểm tra hồi quy

* Một bộ kiểm thử hồi quy được phát triển dần dần khi một chương trình được phát triển.

Gỡ lỗi đơn giản

* Khi một thử nghiệm thất bại, nó sẽ được rõ ràng, nơi vấn đề nằm. Mã mới được viết cần được kiểm tra và sửa đổi.

Tài liệu hệ thống

* Bản thân các bài kiểm tra là một dạng tài liệu mô tả mã nên làm gì.

## 11.32 Kiểm tra hồi quy

* Kiểm tra hồi quy đang kiểm tra hệ thống để kiểm tra xem các thay đổi có không phải là 'làm hỏng' mã hoạt động trước đó hay không.
* Trong quy trình kiểm tra thủ công, kiểm thử hồi quy là tốn kém nhưng, với thử nghiệm tự động, nó đơn giản và dễ hiểu. Tất cả các bài kiểm tra đều chạy lại mỗi lần thay đổi được thực hiện cho chương trình.
* Các thử nghiệm phải chạy ‘thành công’ trước khi thay đổi được thực hiện.

## 11.33 Kiểm tra phát hành

* Kiểm tra phát hành là quá trình thử nghiệm một bản phát hành cụ thể của một hệ thống được thiết kế để sử dụng bên ngoài nhóm phát triển.
* Mục tiêu chính của quá trình thử nghiệm phát hành là thuyết phục nhà cung cấp của hệ thống rằng nó đủ tốt để sử dụng.
  + Do đó, việc thử nghiệm phát hành phải cho thấy rằng hệ thống cung cấp chức năng, hiệu suất và độ tin cậy được chỉ định của nó và rằng nó không bị lỗi trong quá trình sử dụng bình thường.
* Phát hành thử nghiệm thường là một quá trình thử nghiệm hộp đen, nơi các thử nghiệm chỉ được lấy từ đặc tả hệ thống.

## 11.34 Phát hành thử nghiệm và kiểm tra hệ thống

* Kiểm tra phát hành là một dạng thử nghiệm hệ thống.
* Sự khác biệt quan trọng:
  + Một nhóm riêng biệt không tham gia vào việc phát triển hệ thống, phải chịu trách nhiệm kiểm tra phát hành.
  + Kiểm tra hệ thống của nhóm phát triển nên tập trung vào việc phát hiện lỗi trong hệ thống (kiểm tra lỗi). Mục tiêu của thử nghiệm phát hành là kiểm tra xem hệ thống có đáp ứng các yêu cầu của nó và đủ tốt để sử dụng bên ngoài (kiểm tra xác thực) hay không.

## 11.35 Yêu cầu dựa trên thử nghiệm

* Kiểm tra dựa trên yêu cầu bao gồm kiểm tra từng yêu cầu và phát triển một bài kiểm tra hoặc kiểm tra cho nó.
* Yêu cầu MHC-PMS:
  + Nếu bệnh nhân được biết là bị dị ứng với bất kỳ loại thuốc cụ thể nào, thì việc kê toa thuốc đó sẽ dẫn đến một thông báo cảnh báo được đưa ra cho người sử dụng hệ thống.
  + Nếu người kê đơn chọn bỏ qua cảnh báo dị ứng, họ sẽ cung cấp lý do tại sao điều này bị bỏ qua.

### 11.35.1 Yêu cầu kiểm tra

* Thiết lập một hồ sơ bệnh nhân không có dị ứng được biết đến. Kê đơn thuốc cho dị ứng được biết là tồn tại. Kiểm tra xem thông báo cảnh báo có được hệ thống phát hành không.
* Thiết lập một hồ sơ bệnh nhân với một dị ứng được biết đến. Kê đơn thuốc cho bệnh nhân bị dị ứng, và kiểm tra xem hệ thống có phát ra cảnh báo hay không.
* Thiết lập một hồ sơ bệnh nhân trong đó dị ứng với hai hoặc nhiều loại thuốc được ghi lại. Kê toa cả hai loại thuốc này một cách riêng biệt và kiểm tra xem có cảnh báo chính xác cho từng loại thuốc không.
* Kê toa hai loại thuốc mà bệnh nhân bị dị ứng. Kiểm tra xem hai cảnh báo có được cấp đúng hay không.
* Quy định một loại thuốc phát hành cảnh báo và ghi đè cảnh báo đó. Kiểm tra xem hệ thống có yêu cầu người dùng cung cấp thông tin giải thích tại sao cảnh báo bị loại bỏ hay không.

## 11.36 Tính năng thử nghiệm theo kịch bản

* Xác thực bằng cách đăng nhập vào hệ thống.
* Tải xuống và tải lên hồ sơ bệnh nhân được chỉ định cho máy tính xách tay.
* Trang chủ lịch trình truy cập.
* Mã hóa và giải mã hồ sơ bệnh nhân trên thiết bị di động.
* Ghi lại và sửa đổi bản ghi.
* Liên kết với cơ sở dữ liệu thuốc duy trì thông tin tác dụng phụ.
* Hệ thống nhắc cuộc gọi.

## 11.37 Kịch bản sử dụng cho MHC-PMS

|  |
| --- |
| Kate là một y tá chuyên chăm sóc sức khỏe tâm thần. Một trong những trách nhiệm của cô là đến thăm các bệnh nhân ở nhà để kiểm tra xem việc điều trị của họ có hiệu quả không và họ không bị tác dụng phụ của thuốc.  Vào một ngày thăm nhà, Kate đăng nhập vào MHC-PMS và sử dụng nó để in lịch trình thăm nhà của cô cho ngày hôm đó, cùng với thông tin tóm tắt về bệnh nhân được đến thăm. Cô yêu cầu các hồ sơ cho những bệnh nhân này được tải xuống máy tính xách tay của cô. Cô được nhắc nhở cho cụm từ khóa của mình để mã hóa các hồ sơ trên máy tính xách tay.  Một trong những bệnh nhân mà cô đến thăm là Jim, người đang được điều trị bằng thuốc trị trầm cảm. Jim cảm thấy rằng thuốc đang giúp anh ta nhưng tin rằng nó có tác dụng phụ khiến anh thức giấc vào ban đêm. Kate tra cứu hồ sơ của Jim và được nhắc cho cụm từ khóa của mình để giải mã hồ sơ. Cô kiểm tra thuốc theo quy định và truy vấn các tác dụng phụ của thuốc. Mất ngủ là một tác dụng phụ đã biết nên cô ghi nhận vấn đề trong hồ sơ của Jim và gợi ý rằng anh đến phòng khám để thay đổi thuốc. Anh đồng ý vì vậy Kate đi vào một lời nhắc gọi anh khi cô quay lại phòng khám để hẹn gặp bác sĩ. Cô kết thúc buổi tư vấn và hệ thống mã hóa lại hồ sơ của Jim.  Sau khi hoàn thành việc tham vấn, Kate trở về phòng khám và tải lên hồ sơ bệnh nhân đến cơ sở dữ liệu. Hệ thống này tạo ra một danh sách cuộc gọi cho Kate của những bệnh nhân mà cô ấy phải liên lạc để theo dõi thông tin và thực hiện các cuộc hẹn khám. |

## 11.38 Kiểm tra năng suất

* Một phần của thử nghiệm phát hành có thể bao gồm kiểm tra các thuộc tính nổi bật của một hệ thống, chẳng hạn như hiệu suất và độ tin cậy.
* Các xét nghiệm phải phản ánh hồ sơ sử dụng của hệ thống.
* Các thử nghiệm hiệu suất thường liên quan đến việc lên kế hoạch cho một loạt các thử nghiệm mà tải được tăng đều đặn cho đến khi hiệu năng hệ thống trở nên không thể chấp nhận được.
* Thử nghiệm ứng suất là một hình thức kiểm tra hiệu năng trong đó hệ thống được cố tình quá tải để kiểm tra hành vi lỗi của nó.

## 11.39 Kiểm tra người dùng

* Thử nghiệm của người dùng hoặc khách hàng là một giai đoạn trong quá trình thử nghiệm trong đó người dùng hoặc khách hàng cung cấp đầu vào và tư vấn về kiểm tra hệ thống.
* Kiểm tra người dùng là cần thiết, ngay cả khi hệ thống toàn diện và thử nghiệm phát hành đã được thực hiện.
  + Lý do cho điều này là ảnh hưởng từ môi trường làm việc của người dùng có ảnh hưởng lớn đến độ tin cậy, hiệu suất, khả năng sử dụng và tính chắc chắn của một hệ thống. Đây không thể được nhân rộng trong một môi trường thử nghiệm.

### 11.39.1 Các loại thử nghiệm người dùng

Thử nghiệm alpha

* Người dùng phần mềm làm việc với nhóm phát triển để kiểm tra phần mềm tại trang web của nhà phát triển.

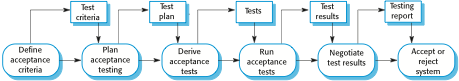
Thử nghiệm beta

* Việc phát hành phần mềm được cung cấp cho người dùng để cho phép họ thử nghiệm và nâng cao các vấn đề mà họ phát hiện với các nhà phát triển hệ thống.

Kiểm tra chấp nhận

* Khách hàng thử nghiệm một hệ thống để quyết định xem nó có sẵn sàng để được chấp nhận từ các nhà phát triển hệ thống và được triển khai trong môi trường khách hàng hay không. Chủ yếu cho các hệ thống tùy chỉnh.

## 11.40 Quá trình kiểm tra nghiệm thu



## 11.40.1 Các giai đoạn trong quá trình kiểm tra nghiệm thu

* Xác định tiêu chí chấp nhận
* Thử nghiệm chấp nhận kế hoạch
* Thử nghiệm chấp nhận có nguồn gốc
* Chạy thử nghiệm chấp nhận
* Thương lượng kết quả kiểm tra
* Từ chối / chấp nhận hệ thống

## 11.41 Phương thức nhanh và thử nghiệm chấp nhận

* Trong các phương thức nhanh, người dùng / khách hàng là một phần của nhóm phát triển và chịu trách nhiệm đưa ra các quyết định về khả năng chấp nhận của hệ thống.
* Kiểm tra được xác định bởi người dùng / khách hàng và được tích hợp với các thử nghiệm khác ở chỗ chúng được chạy tự động khi thay đổi được thực hiện.
* Không có quy trình thử nghiệm chấp nhận riêng biệt.
* Vấn đề chính ở đây là liệu người dùng được nhúng có 'điển hình' hay không và có thể đại diện cho quyền lợi của tất cả các bên liên quan của hệ thống.

**Key points**

* Khi kiểm tra phần mềm, bạn nên cố gắng 'phá vỡ' phần mềm bằng cách sử dụng kinh nghiệm và hướng dẫn để chọn loại trường hợp thử nghiệm đã có hiệu quả trong việc phát hiện khuyết tật trong các hệ thống khác.
* Bất cứ nơi nào có thể, bạn nên viết các bài kiểm tra tự động. Các bài kiểm tra được nhúng trong một chương trình có thể chạy mỗi lần thay đổi được thực hiện cho một hệ thống.
* Phát triển thử nghiệm đầu tiên là một cách tiếp cận để phát triển, nơi các bài kiểm tra được viết trước khi mã được kiểm tra.
* Thử nghiệm kịch bản bao gồm việc phát minh ra một kịch bản sử dụng điển hình và sử dụng điều này để lấy ra các trường hợp thử nghiệm.
* Thử nghiệm chấp nhận là một quá trình thử nghiệm người dùng, nơi mục đích là để quyết định xem phần mềm có đủ tốt để được triển khai và sử dụng trong môi trường hoạt động của nó hay không.

# Chapter 12: Project Planning(lập kế hoạch dự án)

# Chủ đề được đề cập:

• Giá phần mềm

• Phát triển theo kế hoạch

• Lập kế hoạch dự án

• Lập kế hoạch nhanh

• Kỹ thuật ước tính

# Lập kế hoạch dự án

* Lập kế hoạch dự án liên quan đến việc chia nhỏ công việc thành các phần và gán chúng cho các thành viên nhóm dự án, dự đoán các vấn đề có thể phát sinh và chuẩn bị các giải pháp dự kiến ​​cho những vấn đề đó.
* Kế hoạch dự án, được tạo ra khi bắt đầu dự án, được sử dụng để truyền đạt cách thức công việc sẽ được thực hiện cho nhóm dự án và khách hàng, đồng thời giúp đánh giá tiến độ dự án.

Các giai đoạn lập kế hoạch

1. Ở giai đoạn đề xuất, khi bạn đặt giá thầu cho một hợp đồng để phát triển hoặc cung cấp một hệ thống phần mềm.
2. Trong giai đoạn khởi động dự án, khi bạn phải lập kế hoạch ai sẽ làm việc trên dự án, cách dự án sẽ được chia nhỏ thành số gia tăng, cách tài nguyên sẽ được phân bổ trên công ty của bạn, v.v.
3. Định kỳ trong suốt dự án, khi bạn sửa đổi kế hoạch của bạn trong ánh sáng của kinh nghiệm thu được và thông tin từ giám sát tiến độ của công việc.

Lập kế hoạch đề xuất

🡪Việc lập kế hoạch có thể cần thiết chỉ với các yêu cầu phần mềm phác thảo. Mục đích của việc lập kế hoạch ở giai đoạn này là cung cấp thông tin sẽ được sử dụng trong việc thiết lập giá cho hệ thống cho khách hàng.

Giá phần mềm

* Các ước tính được thực hiện để khám phá chi phí, cho nhà phát triển, sản xuất một hệ thống phần mềm.
* Bạn đưa vào tài khoản, phần cứng, phần mềm, du lịch, đào tạo và chi phí công sức.
* Không có mối quan hệ đơn giản giữa chi phí phát triển và giá cả được tính cho khách hàng.
* Các cân nhắc về tổ chức, kinh tế, chính trị và kinh doanh rộng hơn ảnh hưởng đến giá được tính.

Các yếu tố ảnh hưởng đến việc định giá phần mềm

|  |  |
| --- | --- |
| ***Yếu tố ảnh hưởng*** | ***Mô tả*** |
| Cơ hội thị trường | Một tổ chức phát triển có thể báo giá thấp vì nó muốn chuyển sang một phân đoạn mới của thị trường phần mềm. Việc chấp nhận lợi nhuận thấp trên một dự án có thể mang lại cho tổ chức cơ hội kiếm lợi nhuận lớn hơn sau đó. Kinh nghiệm thu được cũng có thể giúp nó phát triển sản phẩm mới. |
| Chi phí ước tính không chắc chắn | Nếu một tổ chức không chắc chắn về ước tính chi phí của nó, nó có thể làm tăng giá của nó bằng một dự phòng hơn và cao hơn lợi nhuận bình thường của nó. |
| Điều khoản hợp đồng | Khách hàng có thể sẵn lòng cho phép nhà phát triển giữ quyền sở hữu mã nguồn và sử dụng lại nó trong các dự án khác. Giá được tính sau đó có thể nhỏ hơn nếu mã nguồn phần mềm được bàn giao cho khách hàng. |
| Yêu cầu biến động | Nếu các yêu cầu có khả năng thay đổi, một tổ chức có thể hạ thấp giá của nó để giành được hợp đồng. Sau khi hợp đồng được trao, giá cao có thể được tính cho những thay đổi đối với các yêu cầu. |
| Nguồn lực tài chính | Các nhà phát triển gặp khó khăn về tài chính có thể giảm giá của họ để có được hợp đồng. Nó là tốt hơn để làm cho một lợi nhuận nhỏ hơn bình thường hoặc phá vỡ thậm chí hơn để đi ra khỏi kinh doanh. Dòng tiền là quan trọng hơn lợi nhuận trong thời kỳ kinh tế khó khăn. |

# Phát triển theo kế hoạch:

* Lập kế hoạch theo định hướng hoặc phát triển dựa trên kế hoạch là một cách tiếp cận cho kỹ nghệ phần mềm, nơi quy trình phát triển được lên kế hoạch chi tiết.
* Việc phát triển theo kế hoạch dựa trên các kỹ thuật quản lý dự án kỹ thuật và là cách "truyền thống" để quản lý các dự án phát triển phần mềm lớn.
* Một kế hoạch dự án được tạo ra để ghi lại công việc cần làm, ai sẽ làm, lịch phát triển và các sản phẩm công việc.
* Các nhà quản lý sử dụng kế hoạch để hỗ trợ việc ra quyết định dự án và như một cách đo lường tiến độ.

ưu và nhược điểm

* Các lập luận ủng hộ phương pháp định hướng kế hoạch là lập kế hoạch sớm cho phép các vấn đề về tổ chức (tính sẵn có của nhân viên, các dự án khác, v.v.) được xem xét cẩn thận, và các vấn đề tiềm năng và phụ thuộc được phát hiện trước khi dự án bắt đầu, thay vì khi dự án đang được tiến hành.
* Lập luận chính chống lại sự phát triển theo hướng lập kế hoạch là nhiều quyết định ban đầu phải được sửa đổi vì những thay đổi đối với môi trường mà phần mềm sẽ được phát triển và sử dụng.

Kế hoạch dự án

1. Trong một dự án phát triển theo kế hoạch định hướng, một kế hoạch dự án đặt ra các nguồn lực sẵn có cho dự án, bảng phân tích công việc và lịch trình thực hiện công việc.
2. Phần kế hoạch
3. Giới thiệu
4. Tổ chức dự án
5. Phân tích rủi ro
6. Yêu cầu về phần cứng và phần mềm
7. Phân tích công việc
8. Lịch trình dự án
9. Cơ chế giám sát và báo cáo

Bổ sung kế hoạch dự án

|  |  |
| --- | --- |
| **Plan** | **Description** |
| Chất lượng kế hoạch | Mô tả các quy trình và tiêu chuẩn chất lượng sẽ được sử dụng trong một dự án. |
| Kế hoạch xác thực | Mô tả cách tiếp cận, tài nguyên và lịch biểu được sử dụng để xác thực hệ thống. |
| Kế hoạch quản lý cấu hình | Mô tả các quy trình và cấu trúc quản lý cấu hình được sử dụng. |
| Kế hoạch bảo trì | Dự đoán các yêu cầu bảo trì, chi phí và công sức. |
| Kế hoạch phát triển nhân viên | Mô tả các kỹ năng và kinh nghiệm của các thành viên trong nhóm dự án sẽ được phát triển như thế nào. |

**Quy trình lập kế hoạch**

* Lập kế hoạch dự án là một quá trình lặp lại bắt đầu khi bạn tạo một kế hoạch dự án ban đầu trong giai đoạn khởi động dự án.
* Thay đổi kế hoạch là không thể tránh khỏi.
* Khi có thêm thông tin về hệ thống và nhóm dự án có sẵn trong dự án, bạn nên thường xuyên sửa đổi kế hoạch để phản ánh các yêu cầu, lịch biểu và các thay đổi rủi ro.
* Thay đổi mục tiêu kinh doanh cũng dẫn đến những thay đổi trong kế hoạch dự án. Khi mục tiêu kinh doanh thay đổi, điều này có thể ảnh hưởng đến tất cả các dự án, sau đó có thể phải được lên kế hoạch lại.

**1/** Lập kế hoạch dự án:

* Lập kế hoạch dự án là quá trình quyết định cách thức công việc trong một dự án sẽ được tổ chức như các nhiệm vụ riêng biệt, và khi nào và làm thế nào các tác vụ này sẽ được thực hiện.
* Bạn ước tính thời gian lịch cần thiết để hoàn thành mỗi nhiệm vụ, nỗ lực cần thiết và ai sẽ làm việc trên các nhiệm vụ đã được xác định.
* Bạn cũng phải ước tính các tài nguyên cần thiết để hoàn thành mỗi tác vụ, chẳng hạn như dung lượng đĩa cần thiết trên máy chủ, thời gian cần thiết trên phần cứng chuyên dụng, chẳng hạn như trình giả lập và ngân sách du lịch sẽ là bao nhiêu.

**2/** Hoạt động lập kế hoạch dự án

* Chia dự án thành các nhiệm vụ và ước tính thời gian và nguồn lực cần thiết để hoàn thành từng nhiệm vụ.
* Tổ chức các nhiệm vụ đồng thời để tận dụng tối đa lực lượng lao động.
* Giảm thiểu phụ thuộc nhiệm vụ để tránh sự chậm trễ gây ra bởi một nhiệm vụ đang chờ người khác hoàn thành.
* Phụ thuộc vào trực giác quản lý dự án và kinh nghiệm.

3/Cột mốc và phân phối

🡪Các mốc quan trọng là các điểm trong lịch biểu mà bạn có thể đánh giá tiến độ, ví dụ như việc bàn giao hệ thống để thử nghiệm.

* Phân phối là sản phẩm công việc được phân phối cho khách hàng, ví dụ: một tài liệu yêu cầu cho hệ thống.

**4/** Lập lịch sự cố

* Ước tính khó khăn của các vấn đề và do đó chi phí phát triển một giải pháp là khó khăn.
* Năng suất không tỷ lệ thuận với số lượng người làm việc trên một nhiệm vụ.
* Thêm người vào một dự án muộn làm cho nó sau này vì chi phí truyền thông.
* Điều bất ngờ luôn xảy ra. Luôn cho phép dự phòng trong quy hoạch.

**5/**Biểu diễn lịch biểu

* Ký hiệu đồ họa thường được sử dụng để minh họa tiến độ dự án.
* Chúng hiển thị phân tích dự án thành các nhiệm vụ. Nhiệm vụ không nên quá nhỏ. Họ sẽ mất khoảng một hoặc hai tuần.
* Biểu đồ thanh là biểu diễn thường được sử dụng nhất cho lịch biểu dự án.Chúng hiển thị lịch làm hoạt động hoặc tài nguyên theo thời gian

# Agile planning

* Các phương pháp phát triển phần mềm nhanh nhẹn là các phương pháp lặp đi lặp lại, nơi phần mềm được phát triển và chuyển giao cho khách hàng theo từng bước.
* Không giống như phương pháp tiếp cận theo định hướng kế hoạch, chức năng của các gia số này không được lên kế hoạch trước nhưng được quyết định trong quá trình phát triển.
* Quyết định về những gì cần bao gồm trong một khoản tăng phụ thuộc vào tiến độ và các ưu tiên của khách hàng.
* Các ưu tiên và yêu cầu của khách hàng thay đổi nên có ý nghĩa để có một gói linh hoạt có thể đáp ứng những thay đổi này.

Các giai đoạn lập kế hoạch nhanh

* Lập kế hoạch phát hành, trông sẽ được tiến hành trong vài tháng và quyết định các tính năng cần được đưa vào bản phát hành của hệ thống.
* Lập kế hoạch lặp lại, có triển vọng ngắn hạn hơn, và tập trung vào việc lập kế hoạch gia tăng tiếp theo của một hệ thống. Điều này thường là 2-4 tuần làm việc cho nhóm.

Lập kế hoạch dựa trên câu chuyện

* Đặc tả hệ thống trong XP dựa trên các câu chuyện của người dùng phản ánh các tính năng cần được đưa vào hệ thống.
* Nhóm dự án đọc và thảo luận các câu chuyện và xếp hạng chúng theo thứ tự thời gian mà họ nghĩ sẽ cần để thực hiện câu chuyện.
* Lập kế hoạch phát hành bao gồm việc lựa chọn và tinh chỉnh các câu chuyện sẽ phản ánh các tính năng được thực hiện trong một bản phát hành của một hệ thống và thứ tự các câu chuyện cần được thực hiện.
* Các câu chuyện được thực hiện trong mỗi lần lặp được chọn, với số lượng câu chuyện phản ánh thời gian để cung cấp một lần lặp (thường là 2 hoặc 3 tuần).

Những điểm chính

* Giá được tính cho một hệ thống không chỉ phụ thuộc vào chi phí phát triển ước tính của nó; nó có thể được điều chỉnh tùy theo thị trường và ưu tiên tổ chức.
* Phát triển theo kế hoạch định hướng được tổ chức xung quanh một kế hoạch dự án hoàn chỉnh xác định các hoạt động của dự án, nỗ lực được lên kế hoạch, lịch hoạt động và ai chịu trách nhiệm cho từng hoạt động.
* Lập kế hoạch dự án liên quan đến việc tạo ra các đại diện đồ họa kế hoạch dự án. Biểu đồ thanh trình bày thời lượng hoạt động và lịch trình nhân sự, là các biểu diễn lịch biểu thường được sử dụng nhất.
* Trò chơi lập kế hoạch XP liên quan đến toàn bộ nhóm trong việc lập kế hoạch dự án. Kế hoạch được phát triển theo từng bước và, nếu có vấn đề phát sinh, được điều chỉnh. Chức năng phần mềm bị giảm thay vì trì hoãn việc phân phối số gia tăng.

Kỹ thuật ước tính

* Các tổ chức cần nỗ lực phần mềm và ước tính chi phí. Có hai loại kỹ thuật có thể được sử dụng để thực hiện việc này:
* Kỹ thuật dựa trên kinh nghiệm Ước tính các yêu cầu nỗ lực trong tương lai dựa trên kinh nghiệm của người quản lý về các dự án trước đây và miền ứng dụng. Về cơ bản, người quản lý đưa ra một đánh giá thông tin về những yêu cầu nỗ lực có thể là gì.
* Mô hình hóa chi phí thuật toán Trong phương pháp này, một cách tiếp cận công thức được sử dụng để tính toán nỗ lực của dự án dựa trên các ước tính của các thuộc tính sản phẩm, chẳng hạn như kích thước và đặc điểm quy trình, chẳng hạn như kinh nghiệm của nhân viên.

Phương pháp tiếp cận dựa trên kinh nghiệm

* Các kỹ thuật dựa trên kinh nghiệm dựa trên các phán đoán dựa trên kinh nghiệm của các dự án trong quá khứ và nỗ lực chi tiêu trong các dự án này về các hoạt động phát triển phần mềm.
* Thông thường, bạn xác định các phân phôi được sản xuất trong một dự án và các thành phần hoặc hệ thống phần mềm khác nhau sẽ được phát triển.
* Bạn ghi lại các tài liệu này trong bảng tính, ước tính riêng từng cá nhân và tính toán tổng nỗ lực cần thiết.
* Nó thường giúp để có được một nhóm người tham gia vào dự toán nỗ lực và yêu cầu mỗi thành viên của nhóm giải thích ước tính của họ.

Mô hình hóa chi phí thuật toán(Công thưc tính toán)

* Chi phí được ước tính là hàm toán học của các thuộc tính sản phẩm, dự án và quy trình có giá trị được ước tính bởi người quản lý dự án:

***Effort = A \* Size^b \* M***

A là hằng số phụ thuộc vào tổ chức

B phản ánh nỗ lực không cân xứng cho các dự án lớn

M là một số nhân phản ánh sản phẩm, quá trình và các thuộc tính của con người. Thuộc tính sản phẩm thường được sử dụng nhất để ước tính chi phí là kích thước mã.

Hầu hết các mô hình đều giống nhau nhưng chúng sử dụng các giá trị khác nhau cho A, B và M.

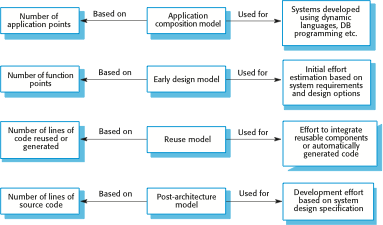
Độ chính xác ước tính

* Kích thước của một hệ thống phần mềm chỉ có thể được biết chính xác khi nó được hoàn thành.
* Một số yếu tố ảnh hưởng đến kích thước cuối cùng
* Sử dụng COTS và các thành phần;
* Ngôn ngữ lập trình;
* Phân phối hệ thống.
* Khi quá trình phát triển diễn ra thì ước tính kích thước trở nên chính xác hơn.
* Các ước tính về các yếu tố đóng góp cho B và M là chủ quan và thay đổi theo phán đoán của người ước lượng.

Mô hình COCOMO 2

* Một mô hình thực nghiệm dựa trên kinh nghiệm dự án.
* Mô hình 'độc lập' được ghi chép đầy đủ, không gắn với một nhà cung cấp phần mềm cụ thể.
* Lịch sử lâu đời từ phiên bản ban đầu được xuất bản vào năm 1981 (COCOMO-81) thông qua các phiên bản khác nhau cho COCOMO 2.
* COCOMO 2 xem xét các cách tiếp cận khác nhau để phát triển phần mềm, tái sử dụng, v.v.
* COCOMO 2 kết hợp một loạt các mô hình phụ tạo ra các ước tính phần mềm ngày càng chi tiết. Các mô hình con trong COCOMO 2 là:
* Mô hình thành phần ứng dụng. Được sử dụng khi phần mềm được tạo từ các phần hiện có.
* Mô hình thiết kế ban đầu. Được sử dụng khi có yêu cầu nhưng thiết kế vẫn chưa bắt đầu.
* Tái sử dụng mô hình. Được sử dụng để tính toán nỗ lực tích hợp các thành phần có thể tái sử dụng.

🡪Mô hình kiến ​​trúc hậu. Được sử dụng khi kiến ​​trúc hệ thống được thiết kế và có thêm thông tin về hệ thống.



Mô hình thành phần ứng dụng

🡪Hỗ trợ các dự án và dự án tạo mẫu, nơi có tái sử dụng rộng rãi. Dựa trên ước tính chuẩn về năng suất của nhà phát triển trong ứng dụng (đối tượng) điểm / tháng. Sử dụng công cụ CASE.

***Công Thức bây giờ là:***

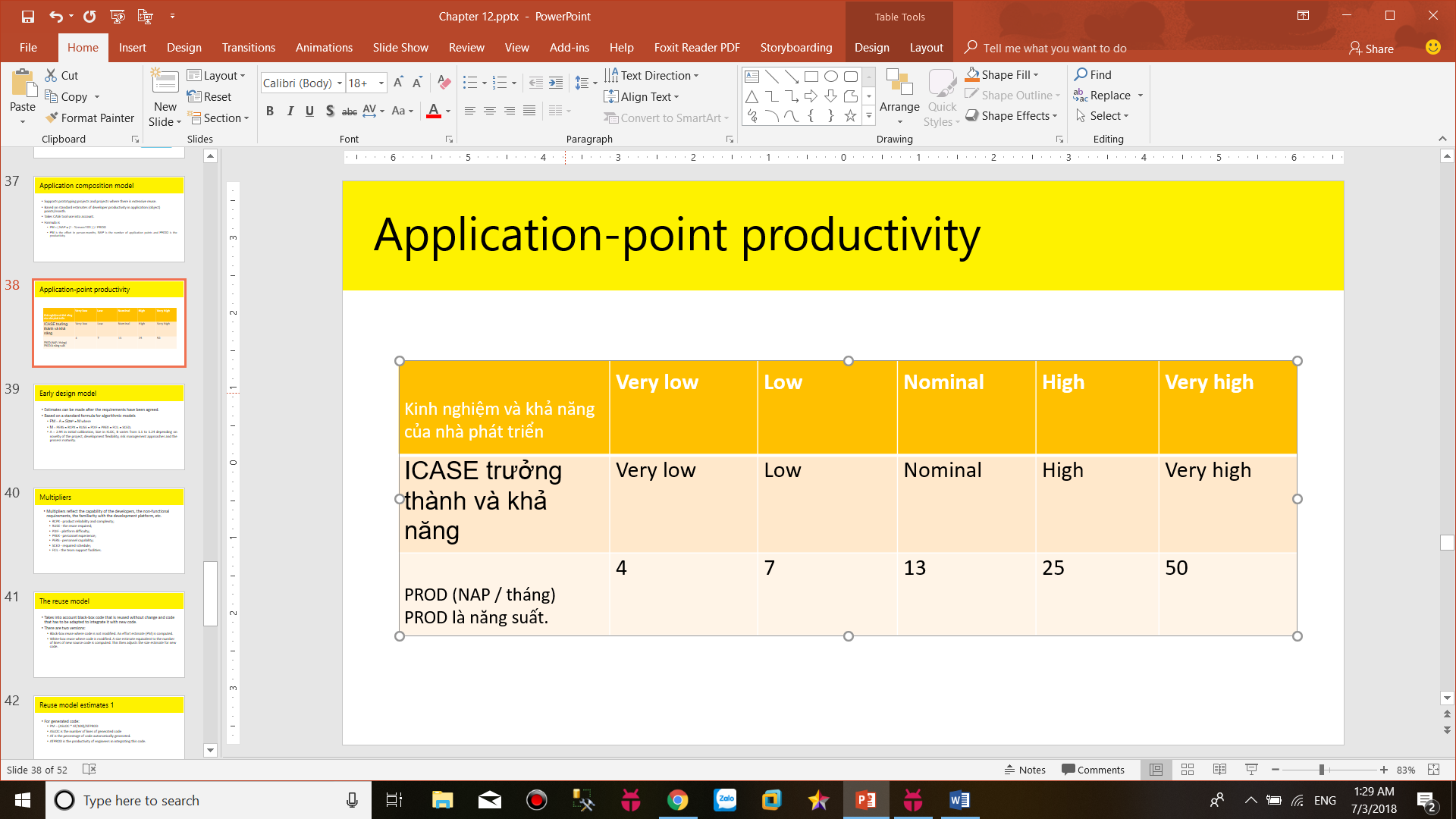
***PM = (NAP \* (1 -% sử dụng lại / 100)) / PROD***

PM là nỗ lực trong người-tháng,

NAP là số lượng điểm ứng dụng

PROD là năng suất.

**Năng suất điểm ứng dụng**



Mô hình thiết kế ban đầu

Các ước tính có thể được thực hiện sau khi các yêu cầu đã được thỏa thuận.

Dựa trên công thức chuẩn cho các mô hình thuật toán

**PM = A \*SizeB\* M ở đâu**

**M = PERS \*RCPX \* RUSE \* PDIF ´ PREX \* FCIL \* SCED;**

A = 2,94 trong hiệu chuẩn ban đầu, Kích thước trong KLOC, B thay đổi từ 1,1 đến 1,24 tùy thuộc vào tính mới của dự án, tính linh hoạt phát triển, phương pháp quản lý rủi ro và quá trình trưởng thành.

Hệ số

Số nhân phản ánh khả năng của các nhà phát triển, các yêu cầu phi chức năng, sự quen thuộc với nền tảng phát triển, v.v.

RCPX - độ tin cậy và độ phức tạp của sản phẩm;

RUSE - yêu cầu tái sử dụng;

PDIF - nền tảng khó khăn;

PREX - kinh nghiệm nhân sự;

PERS - khả năng của nhân viên;

SCED - lịch bắt buộc;

FCIL - các cơ sở hỗ trợ nhóm.

Mô hình tái sử dụng

* Đưa vào tài khoản mã hộp đen được tái sử dụng mà không cần thay đổi và mã phải được điều chỉnh để tích hợp nó với mã mới.
* Có hai phiên bản:
* Tái sử dụng hộp đen khi mã không được sửa đổi. Ước tính nỗ lực (PM) được tính toán.
* Tái sử dụng hộp trắng khi mã được sửa đổi. Ước tính kích thước tương đương với số dòng mã nguồn mới được tính toán. Điều này sau đó điều chỉnh ước tính kích thước cho mã mới.

Tái sử dụng các ước tính mô hình 1

**Đối với mã được tạo: PM = (ASLOC \* AT / 100) / ATPROD**

ASLOC là số dòng mã được tạo ra

AT là phần trăm mã được tạo tự động.

ATPROD là năng suất của các kỹ sư trong việc tích hợp mã này.

Tái sử dụng các ước tính mô hình 2

**Khi mã phải được hiểu và tích hợp:**

**ESLOC = ASLOC \* (1-AT / 100) \* AAM.**

ASLOC và AT như trước.

AAM là hệ số điều chỉnh thích ứng được tính toán từ chi phí thay đổi mã được sử dụng lại, chi phí của việc hiểu cách tích hợp mã và chi phí cho việc ra quyết định sử dụng lại.

Kiến trúc sau  
Sử dụng cùng một công thức như mô hình thiết kế ban đầu nhưng với 17 chứ không phải 7 số nhân liên kết.

**Kích thước mã được ước tính là:**

🡪Số dòng mã mới sẽ được phát triển;

🡪Ước tính số lượng dòng mã mới tương đương được tính toán bằng mô hình tái sử dụng;

🡪Ước tính số lượng dòng mã phải được sửa đổi theo yêu cầu thay đổi.

Thuật ngữ số mũ

Điều này phụ thuộc vào 5 yếu tố tỷ lệ (xem trang tiếp theo). Tổng của họ / 100 được thêm vào 1,01

Một công ty thực hiện dự án trong một miền mới. Khách hàng chưa xác định quy trình được sử dụng và không cho phép thời gian phân tích rủi ro. Công ty có xếp hạng CMM 2.

**Precedenteness - dự án mới (4)**

**Phát triển linh hoạt - không có sự tham gia của khách hàng - Rất cao (1)**

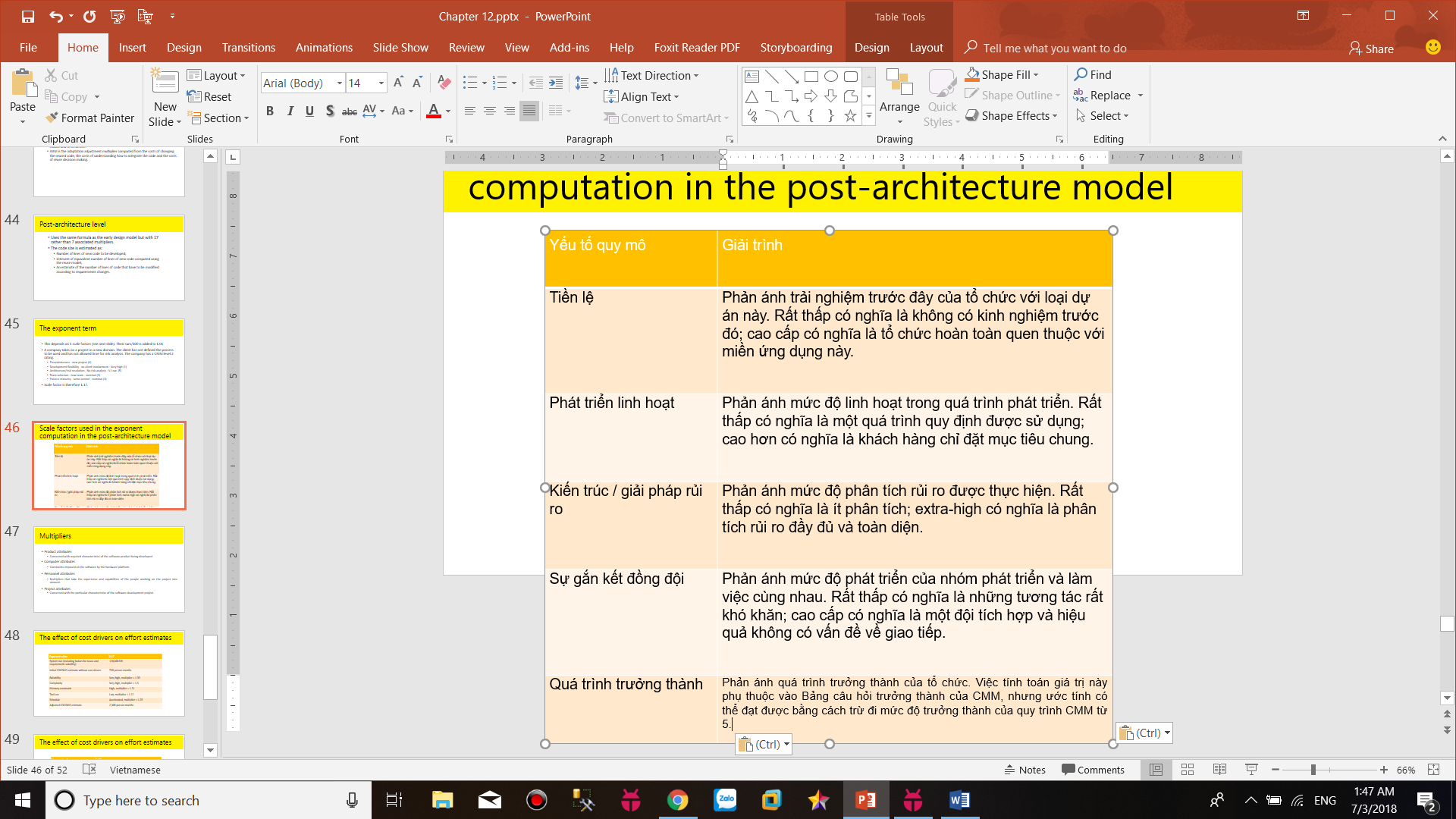
**Kiến trúc / giải pháp rủi ro - Không phân tích rủi ro - V. Thấp. (5)**

**Sự gắn kết của đội - đội mới - danh nghĩa (3)**

**Quá trình trưởng thành - một số kiểm soát - danh nghĩa (3)**

**Do đó, hệ số quy mô là 1,17.**

***Các yếu tố tỷ lệ được sử dụng trong tính toán số mũ trong mô hình sau kiến ​​trúc***



Hệ số

***Thuộc tính sản phẩm***

Quan tâm đến các đặc tính cần thiết của sản phẩm phần mềm đang được phát triển.

***Thuộc tính máy tính***

Các ràng buộc áp đặt trên phần mềm bởi nền tảng phần cứng.

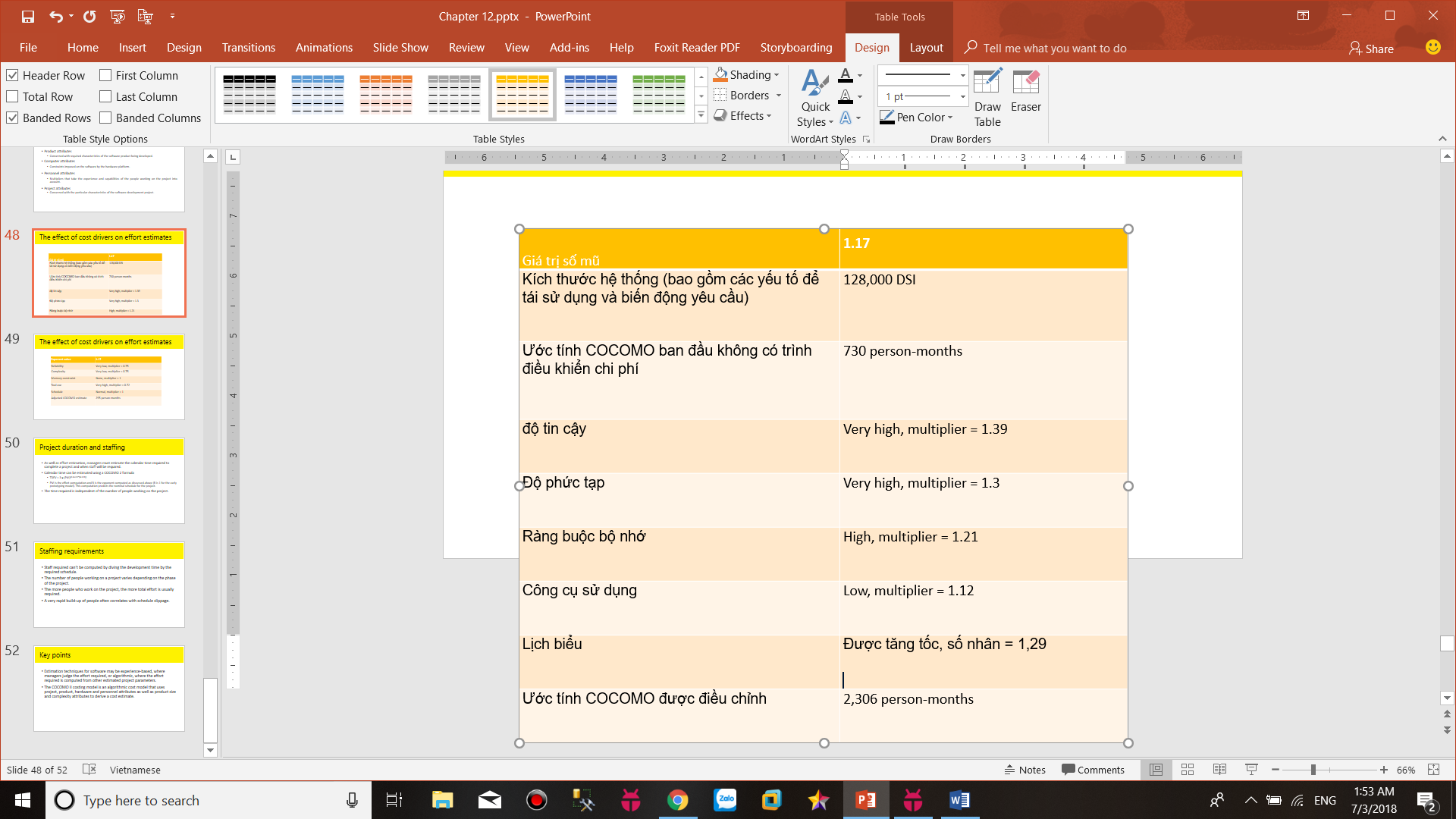
***Thuộc tính nhân sự***

Các nhân viên có kinh nghiệm và khả năng của những người làm việc trong dự án.

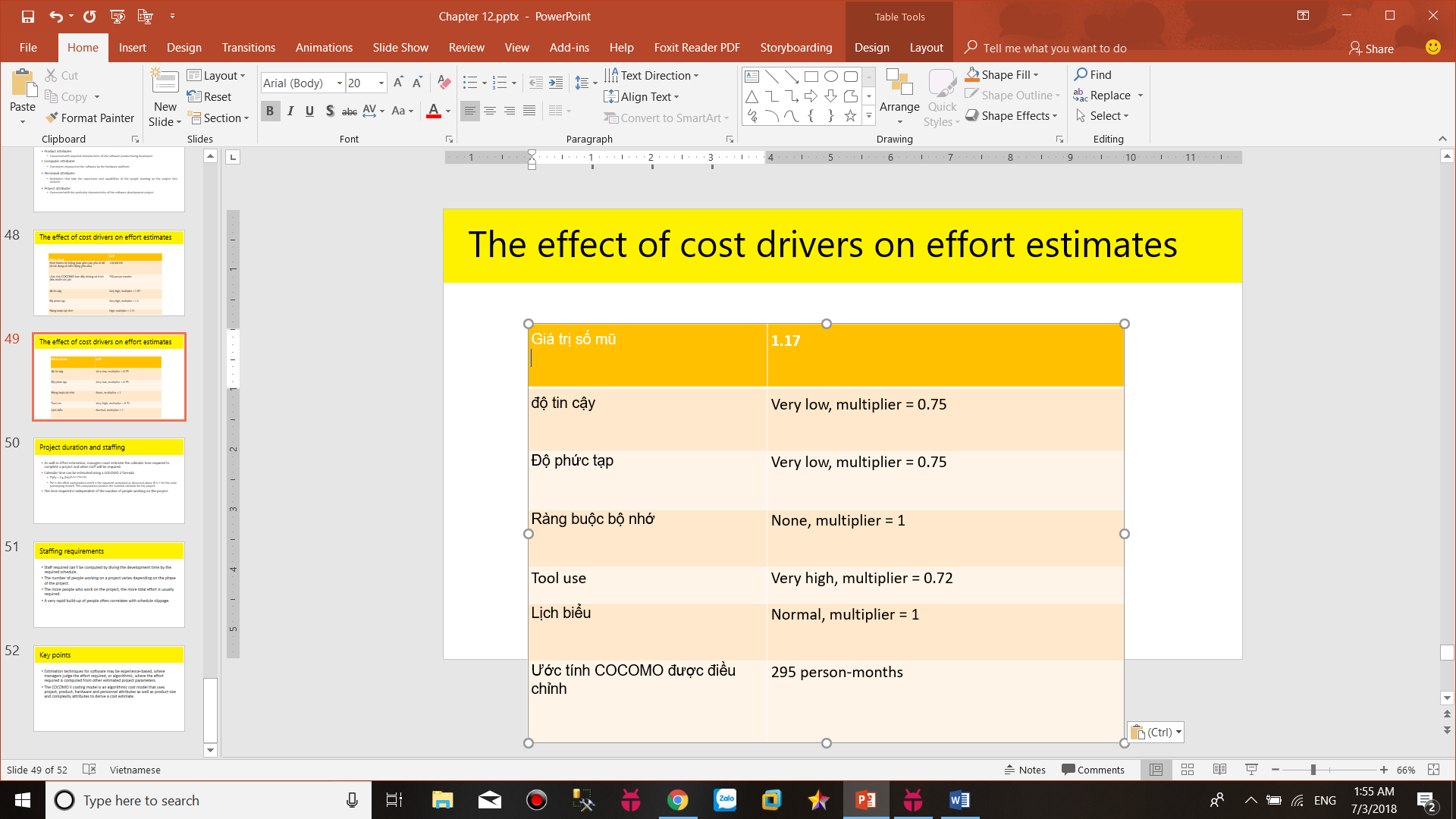
***Thuộc tính dự án***

Quan tâm đến các đặc điểm cụ thể của dự án phát triển phần mềm.

**Ảnh hưởng của trình điều khiển chi phí đến ước tính nỗ lực**



***Ảnh hưởng của trình điều khiển chi phí đến ước tính nỗ lực***



Thời gian dự án và nhân sự

Cũng như ước lượng nỗ lực, các nhà quản lý phải ước tính thời gian lịch cần thiết để hoàn thành một dự án và khi nào nhân viên sẽ được yêu cầu.

***Thời gian lịch có thể được ước tính bằng cách sử dụng công thức COCOMO 2***

***TDEV = 3 \* (PM) (0,33 + 0,2 \* (B-1,01))***

PM là tính toán nỗ lực và B là số mũ được tính như đã thảo luận ở trên (B là 1 cho mô hình tạo mẫu ban đầu). Tính toán này dự đoán lịch biểu danh nghĩa cho dự án.

Thời gian cần thiết là độc lập với số người làm việc trong dự án.

Yêu cầu nhân sự

***Nhân viên yêu cầu không thể tính toán bằng cách lặn thời gian phát triển theo lịch trình yêu cầu.***

***Số lượng người làm việc trên một dự án khác nhau tùy thuộc vào giai đoạn của dự án.***

***Càng có nhiều người làm việc trong dự án, thì càng cần nhiều nỗ lực hơn.***

***Việc xây dựng rất nhanh người thường tương quan với việc trượt lịch trình.***

# ***Những điểm chính***

* ***Các kỹ thuật ước lượng cho phần mềm có thể dựa trên kinh nghiệm, nơi các nhà quản lý đánh giá nỗ lực cần thiết, hoặc thuật toán, nơi nỗ lực cần thiết được tính toán từ các thông số dự án khác ước tính.***
* ***Mô hình chi phí COCOMO II là mô hình chi phí thuật toán sử dụng các thuộc tính dự án, sản phẩm, phần cứng và nhân sự cũng như các thuộc tính kích thước và độ phức tạp của sản phẩm để lấy được ước tính chi phí.***

Xác thực(Validation) là quá trình kiểm tra xem đặc tả có nắm bắt được nhu cầu của khách hàng hay không. "Tôi đã xây dựng những gì tôi nói tôi sẽ?"

Xác minh(Verification) là quá trình kiểm tra xem phần mềm có đáp ứng được đặc điểm kỹ thuật hay không. "Tôi đã xây dựng những gì tôi cần?"

|  |  |
| --- | --- |
| **Verification** | **Validation** |
| 1. Xác minh là một thực hành tĩnh về xác minh tài liệu, thiết kế, mã và chương trình. | 1. Xác nhận là một cơ chế năng động để xác nhận và kiểm tra sản phẩm thực tế. |
| 1. Nó không liên quan đến việc thực thi mã. | 2. Nó luôn luôn liên quan đến việc thực thi mã. |
| 1. Nó là con người dựa trên kiểm tra các tài liệu và các tập tin. | 3. Đó là thực thi dựa trên máy tính của chương trình. |
| 1. Xác minh sử dụng các phương pháp như kiểm tra, đánh giá, hướng dẫn và desk-checking, v.v. | 4. Xác nhận sử dụng các phương pháp như kiểm tra hộp đen (chức năng), kiểm tra hộp màu xám và thử nghiệm hộp màu trắng (kết cấu), v.v. |
| 1. Xác minh là kiểm tra xem phần mềm có tuân thủ các thông số kỹ thuật hay không. | 5. Xác nhận là kiểm tra xem phần mềm có đáp ứng được các yêu cầu và yêu cầu của khách hàng hay không. |
| 1. Nó có thể bắt lỗi mà xác nhận không thể bắt được. Đó là bài tập ở mức độ thấp. | 6. Nó có thể bắt lỗi mà xác minh không thể bắt được. Đó là bài tập ở mức độ cao. |
| 1. Mục tiêu là đặc tả yêu cầu, kiến trúc ứng dụng và phần mềm, mức cao, thiết kế hoàn chỉnh và thiết kế cơ sở dữ liệu, v.v. | 7. Mục tiêu là sản phẩm thực tế-một đơn vị, một mô-đun, một uốn cong của mô-đun tích hợp, và sản phẩm cuối cùng hiệu quả. |
| 1. Quá trình xác minh được thực hiện bởi nhóm QA để đảm bảo rằng phần mềm theo thông số kỹ thuật trong tài liệu SRS. | 8. Việc xác nhận được thực hiện với sự tham gia của nhóm thử nghiệm. |
| 1. Nó thường được thực hiện trước khi xác nhận. | 9. Nó thường sau khi xác minh. |

|  |  |
| --- | --- |
| **Alpha testing** | **Beta testing** |
| Thử nghiệm alpha được thực hiện bởi những người kiểm tra thường là nhân viên nội bộ của tổ chức | Thử nghiệm beta được thực hiện bởi khách hàng hoặc người dùng cuối không phải là nhân viên của tổ chức |
| Thử nghiệm alpha được thực hiện tại trang web của nhà phát triển | Thử nghiệm beta được thực hiện tại vị trí của khách hàng hoặc người dùng cuối của sản phẩm |
| Độ tin cậy và kiểm tra bảo mật không được thực hiện trong thử nghiệm alpha chuyên sâu | Độ tin cậy, bảo mật, độ mạnh được kiểm tra trong khi thử nghiệm beta |
| Thử nghiệm alpha liên quan đến cả hộp màu trắng và kỹ thuật hộp đen | Thử nghiệm beta thường sử dụng Kiểm tra hộp đen |
| Thử nghiệm alpha yêu cầu môi trường phòng thí nghiệm hoặc môi trường thử nghiệm | Thử nghiệm beta không yêu cầu bất kỳ môi trường phòng thí nghiệm hoặc môi trường thử nghiệm nào. Phần mềm được cung cấp cho công chúng và được cho là môi trường thời gian thực |
| Có thể cần có chu kỳ thực thi dài để thử nghiệm Alpha | Chỉ cần vài tuần thực thi để thử nghiệm Beta |
| Các vấn đề quan trọng hoặc bản sửa lỗi có thể được các nhà phát triển giải quyết ngay lập tức trong thử nghiệm Alpha | Hầu hết các vấn đề hoặc phản hồi được thu thập từ thử nghiệm Beta sẽ được triển khai trong các phiên bản tương lai của sản phẩm |
| Thử nghiệm alpha là để đảm bảo chất lượng của sản phẩm trước khi chuyển sang thử nghiệm Beta | Thử nghiệm beta cũng tập trung vào chất lượng của sản phẩm, nhưng tập hợp người dùng nhập vào sản phẩm và đảm bảo rằng sản phẩm đã sẵn sàng cho người dùng thời gian thực. |