**Chap 4**

**Backup programmer :**  chỉ cần thiết vì trưởng lập trình viên là con người và do đó có thể bị ốm, ngã dưới gầm xe buýt hoặc thay đổi công việc. lập trình viên dự phòng phải có năng lực như lập trình viên trưởng về mọi mặt và phải biết nhiều về dự án như lập trình viên trưởng

**Chief programmer :** vừa là một nhà quản lý thành công vừa là một lập trình viên có kỹ năng cao, người đã thiết kế kiến trúc và bất kỳ phần quan trọng hoặc phức tạp nào của mã. Các thành viên khác trong nhóm làm việc trên thiết kế chi tiết và mã hóa, dưới sự chỉ đạo của Chief programmer.

**Chief programmer team :** Nó bao gồm lập trình viên trưởng, người được hỗ trợ bởi lập trình viên dự phòng, thư ký lập trình và từ một đến ba lập trình viên

**Democratic team :** Một nhóm gồm tối đa 10 lập trình viên Egoless tạo thành một nhóm .Một nhóm dân chủ không có người lãnh đạo

**Định luật Brooks :** là một quan sát về [quản lý dự án phần mềm](https://en.wikipedia.org/wiki/Software_project_management), theo đó việc bổ sung nhân lực cho một dự án phần mềm bị chậm tiến độ sẽ làm nó bị trì hoãn lâu hơn.

**Egoless programming :** là một phong cách [lập trình máy tính](https://en.wikipedia.org/wiki/Computer_programming) trong đó các yếu tố cá nhân (cái tôi cá nhân) được giảm thiểu để chất lượng có thể được cải thiện.

**Hierarchy (Tính phân cấp)** : Lập trình viên trưởng chỉ đạo hành động của tất cả các thành viên khác trong nhóm và chịu trách nhiệm về mọi khía cạnh của hoạt động.

**Key process areas :** là một nhóm các khối xây dựng mà một tổ chức nên tập trung để cải tiến quy trình phần mềm. Mỗi cấp độ trưởng thành đều có các **key process areas** (KPA) riêng, mỗi trong số đó cần được giải quyết thỏa đáng trước khi một tổ chức có thể được coi là đã đạt được mức độ trưởng thành đó. Ví dụ: đối với cấp độ 2, cấp độ được quản lý, KPA là nhân sự, giao tiếp và phối hợp, môi trường làm việc, quản lý hiệu suất, đào tạo và phát triển, và bồi thường. Ngược lại, KPA cho cấp độ 5, cấp độ tối ưu hóa, là cải tiến năng lực liên tục, liên kết hiệu quả hoạt động của tổ chức và đổi mới lực lượng lao động liên tục.

**Librarian:** bởi vì *programming secretary* duy trì thư viện sản xuất dự án, một số tổ chức đã sử dụng chức danh **librarian**

Mọi lập trình viên phải khuyến khích các thành viên khác trong nhóm tìm lỗi trong mã của họ. Sự hiện diện của một lỗi không được coi là một điều gì đó tồi tệ mà là một sự kiện bình thường và được chấp nhận; thái độ của người đánh giá nên được đánh giá cao khi được hỏi lời khuyên, hơn là chế nhạo lập trình viên vì đã mắc lỗi viết mã.

**Pair programming**  là kiểu lập trình đòi hỏi hai [kỹ sư phần mềm](https://vi.wikipedia.org/wiki/K%E1%BB%B9_s%C6%B0_ph%E1%BA%A7n_m%E1%BB%81m) cùng tham gia một nỗ lực lập trình chung trên một máy trạm, nghĩa là chỉ có một màn hình, một bàn phím. Mỗi người thực hiện việc mà người kia hiện không làm. Ví dụ, người này gõ các bộ test đơn vị (*unit test*), người kia nghĩ về các lớp đầu vào (*input*) sẽ thỏa mãn bộ test đó; hoặc người này viết mã còn người kia quan sát để hướng dẫn hoặc kiểm lỗi. Người ta khuyên rằng hai người nên luân phiên đổi vai trò, khoảng nửa giờ một lần.

**Programmers :** nhiệm vụ chính là lập trình

**Programming secretary :** chịu trách nhiệm duy trì thư viện sản xuất dự án, tài liệu của dự án. Điều này bao gồm danh sách mã nguồn, JCL và dữ liệu thử nghiệm. Các lập trình viên giao mã nguồn của họ cho thư ký, người chịu trách nhiệm chuyển đổi sang dạng máy có thể đọc được, biên dịch, liên kết, tải, thực thi và chạy các trường hợp thử nghiệm

**Specialization (Tính chuẩn hóa)** : Mỗi thành viên trong nhóm chỉ thực hiện những nhiệm vụ mà họ đã được đào tạo.

**Superprogrammer :** là một lập trình viên có kết quả gấp bốn hoặc năm lần so với một lập trình viên giỏi trung bình

**Task** là một đơn vị thực thi hoặc một đơn vị công việc, trong chương 4 là: vẽ các trường hợp thử nghiệm và sau đó triển khai đoạn mã đó

**Team :** đội ngũ tập hợp bởi các chuyên gia phát triển phần mềm

**Team leader :** phụ trách các khía cạnh kỹ thuật của các hoạt động của nhóm

**Team manager :** nhóm chịu trách nhiệm về tất cả các quyết định quản lý phi kỹ thuật, người quản lý nhóm thu thập kiến thức về các kỹ năng kỹ thuật của từng lập trình viên trong nhóm trong các cuộc họp nhóm được lên lịch thường xuyên.

**Chap 6**

**Correctness** là độ chính xác của 1 phần mềm. 1 phần mềm đạt được độ chính xác nếu nó đáp ứng được các thông số đầu ra của nó, không phụ thuộc việc sử dụng tài nguyên của máy, khi được vận hành trong điều kiện cho phép.

**Correctness proof** là phương pháp chứng minh độ chính xác, là kỹ thuật toán học để đánh giá sự chính xác của 1 sản phẩm, đáp ứng các thông số kĩ thuật của sản phẩm

**Defect** là lỗi nói chung, bao gồm fault(lỗi do người gây ra), failure (lỗi hành vi có thể thấy được của sản phẩm), error (sản phẩm trả về kết quả sai)

**Desk check** là kiểm tra thủ công mã nguồn của chương trình bằng cách thử qua lưu đồ hoặc giả mã với nhiều trường hợp test khác nhau

**Error** là lỗi khi chương tình sản phẩm trả về kết quả sai

**Execution-based testing** là quá trình phán đoán các hành vi nhất định của 1 sản phẩm, dựa trên 1 phần kết quả thực nghiệm sản phẩm trong môi trường đã biết với các yêu cầu đầu vào đã chọn lọc

**Failure** : thất bại – là sự khác biệt giữa kết quả thực tế trên màn hình và kết quả mong đợi của 1 hàm, hệ thống hoặc dịch vụ nào đó

**Fault** : lỗi – khuyết điểm

**Fault density** : mật độ lỗi , là số lượng lỗi đực xác định trong 1 thành phần hoặc hệthoongs chia cho kích thước của thành phần hoặc hệ thống – được đo bằng lỗi trên mỗi trang được kiểm tra hoặc trên 1000 dòng mà được kiểm tra

**Fault detection efficiency** : hiệu quả phát hiện lỗi, là số lượng lỗi lớn và nhỏ được phát hiện mỗi giờ

**Follow-up** : quá trình theo dõi – người điều hành phải đảm bảo rằng mọi vấn đề được giải quyết thỏa đáng bằng cách sửa tài liệu hoặc làm rõ các mục gán cờ sai là lỗi

**Inspection ( kiểm tra )**: xem sản phẩm trên thực tế có phù hợp với các yêu cầu đã đặt ra hay không, và đảm bảo rằng không có lỗi hay khiếm khuyết.

**Inspection rate ( tỷ lệ kiểm tra ):** Khi kiểm tra thông số kỹ thuật và thiết kế, có thể đo được số trang được kiểm tra mỗi giờ; đối với kiểm tra mã, số liệu thích hợp là các dòng mã được kiểm tra mỗi giờ.

**Loop invariant ( Bất biến vòng lặp ):** là điều kiện (condition) về mối quan hệ giữa các biến (variable) trong chương trình của chúng ta mà nó chắc chắn đúng (true) ngay trước và ngay sau mỗi lần lặp (iteration) của vòng lặp (loop).

**Managerial independence ( sự độc lập về mặt quản lý ):** với mỗi giai đoạn cần có 1 người quản lý khác nhau để đảm bảo chất lượng của phần mềm.

**Mean time between failures( thời gian trung bình giữa các lần lỗi ):** tần suất sản phẩm bị lỗi.

**Mean time to repair** là thời gian trung bình để sửa chữa 1 mục cụ thể, 1 phần cụ thể và đưa nó lại trạng thái có thể làm việc

**Mistake** là lỗi của chuyên gia khi phát triển phần mềm

**Model checking** là kỹ thuật mới có thể thay thế correctness proof. Là kỹ thuật kiểm định tự động một cách hình thức một mô hình với một đặc tả các tính năng của mô hình đó

**Moderator** vừa là quản lý, vừa là người đứng đầu đội ngũ kiểm tra. Là người chịu trách nhiệm về quy trình làm việc hiện tại, cũng như những quy trình tiếp theo

**Non-execution-based testing** là thực hiện kiểm tra phần mềm mà không chạy các bài kiểm thử. Bao gồm việc xem xét cẩn thận phần mềm và phân tích phần mềm 1 cách logic

**Overview** là khâu xem xét, kiểm tra tổng quan về tài liệu được đưa ra bởi một trong những cá nhân chịu trách nhiệm về tài liệu đó

**Performance** là hiệu suất. Đây là 1 trong những khía cạnh của sản phẩm cần được kiểm tra. Là yếu tố cần thiết để biết mức độ sản phẩm đáp ứng các hạn chế, yêu cầu về thời gian và không gian

**Preparation** là khâu chuẩn bị. Khi ấy những người tham gia phải hiểu được chi tiết tài liệu

**Quality** : chất lượng của phần mềm là mức độ đáp ứng các đặc tả của nó

**Reader** : người đọc dẫn dắt nhóm thông qua thiết kế

**Realiability** : độ tin cậy, là thước đo tần suất và mức độ nghiêm trọng của lỗi sản phẩm

**Recorder** : người ghi, chịu trách nhiệm tạo báo cáo bằng văn bản về các lỗi được phát hiện

**Regresstion testing** : kiểm thử hồi quy, là các trường hợp thử nghiệm được lưu trữ mà sản phẩm đã thực hiện chính xác trước đó, phải được chạy lại để đảm bảo rằng các sửa đổi được thực hiện để thêm chức năng mới cho sản phẩm, k phá hủy chức năng hiện có của sản phẩm

**Rework** : quá trình làm lại – cá nhân chịu trách nhiệm về tài liệu sẽ giải quyết tất cả các lỗi và vấn đề được ghi trong báo cáo bằng văn bản

**Robustness** : độ bền, là 1 chức năng của 1 số yếu tố như phạm vi điều kiện hoạt động, khả năng xảy ra các kết quả không thể chấp nhận được với đầu vào hợp lệ và khẳ năng chấp nhận các tác động khi sản phẩm được cung cấp đầu vào không hợp lệ

**Simulator** : trình mô phỏng – là hoạt đọng của môi trường có thể là 1 sự gần đúng của một mô hình trung thực về 1 số khía cạnh của hệ thống – k phải hệ thống

**Software quality assurance (SQA)( đảm bảo chất lượng phần mềm ):** một quy trình đảm bảo rằng tất cả các quy trình, phương pháp, hoạt động và hạng mục công việc kỹ thuật phần mềm đều được giám sát và tuân thủ các tiêu chuẩn đã xác định.

**Systematic testing ( kiểm thử hệ thống ):** là một phương pháp theo dõi và đánh giá hành vi của sản phẩm hoặc hệ thống phần mềm hoàn chỉnh và đã được tích hợp đầy đủ, dựa vào đặc tả và các yêu cầu chức năng đã được xác định trước.

**Test workflow ( kiểm tra quy trình làm việc ):** được thực hiện để đảm bảo rằng mỗi quy trình làm việc phản ánh chính xác quy trình nghiệp vụ.

**Testing ( kiểm thử ):** là hoạt động nhằm tìm kiếm và phát hiện ra các lỗi.

**Utility ( tính thiết thực ):** sát với những yêu cầu được đưa ra.

V & V : viết tắt của cụm từ verification and validation ( xác minh và xác thực)

**Validation (xác thực):** là quá trình đánh giá chuyên sâu diễn ra ngay trước khi sản phẩm được giao cho khách hàng.

**Verification ( xác minh ):** quá trình xác định liệu một quy trình công việc có được thực hiện chính xác hay không.