Đại cương về công nghệ phần mềm

Mục lục

CHƯƠNG 1

Đại cương về công nghệ phần mềm

I. Khái quát về lịch sử lập trình

Lập trình (programming), hay *lập chương trình cho máy tính điện tử* (MTĐT) là một ngành còn rất mới mẻ. MTĐT đầu tiên lập trình được mới chỉ xuất hiện cách đây hơn bốn mươi năm [1]. Suốt hơn bốn thập kỷ qua, lập trình không ngừng được cải tiến và phát triển, càng ngày càng hướng về nhu cầu của người lập trình.

Lập trình là một công việc nặng nhọc, năng suất thấp so với các hoạt động trí tuệ khác. Ví dụ nếu một sản phẩm phần mềm khoảng 2000 – 3000 dòng lệnh đòi hỏi 3 người lập trình chính trong vòng 6 tháng thì năng suất mỗi người chỉ dao động trong khoảng từ 5 đến 6 lệnh mỗi ngày (?!).

Chính vì các sản phẩm phần mềm khi tung ra thị trường chưa thực sự hoàn hảo ngay nên người ta thường dùng mẹo thương mại bằng cách gán cho sản phẩm một cái đuôi "phiên bản" (version) để nói rằng phiên bản ra sau đã khắc phục được những khiếm khuyết của phiên bản trước đó.

Ví dụ 1 :

Hệ điều hành MS–DOS đã có các phiên bản 1.0, 3.3, 5.0, 6.0, 7.0 v.v...

Microsoft Windows đã có các phiên bản 1.0, 2.0, 3.0, 3.1, 3.11.

Nay là Windows 95, 97, 98 v.v...

Turbo Psacal của hãng Borland Inc. đã có các phiên bản 5.0, 6.0, 7.0, 8.0 v.v...

I.1. Lập trình tuyến tính

Với những MTĐT đầu tiên, người ta sử dụng ngôn ngữ máy (machine language) hay ngôn ngữ bậc thấp (low level) để lập trình và dùng các khoá cơ khí để nạp chương trình vào máy. Theo đà phát triển của các thiết bị phần cứng, các ngôn ngữ bậc cao (high level) với các dòng lệnh tựa tiếng Anh bắt đầu được sử dụng. Máy sẽ dịch chương trình đó sang ngôn ngữ máy trước khi thực hiện.

Với những ngôn ngữ lập trình ban đầu, chương trình viết ra gồm những dòng lệnh có khuynh hướng nối nhau theo dãy dài, khó hiểu về mặt logic. Người ta sử

[1] ENIAC (Electronic Numerical Integrator and Computer) là chiếc MTĐT đầu tiên ra đời năm 1945 tại trường Đại học Tổng hợp Pensylvania, nước Mỹ.

dụng các lệnh nhảy (goto) để điều khiển chương trình một cách tuỳ tiện. Chương trình là một mớ rối rắm không khác gì món mì sợi (spaghetti) của nước Ý.

Các ngôn ngữ lập trình tuyến tính không kiểm soát được những sư thay đổi của dữ liệu. Mọi dữ liệu sử dụng trong chương trình đều có tính toàn cục và có thể bị thay đổi vào bất cứ lúc nào. Vào giai đoạn này, người ta xem việc lập trình như một hoạt động nghệ thuật nhuốm màu sắc tài nghệ cá nhân hơn là khoa học, với thuật ngữ "the art of programming".

I.2. Lập trình có cấu trúc

Vào cuối những năm 1960 và đầu 1970, khuynh hướng lập trình cấu trúc (structured programming) ra đời. Theo phương pháp này, một chương trình có cấu trúc được tổ chức theo các phép toán mà nó phải thực hiện. Chương trình bao gồm nhiều thủ tục, hay hàm, riêng rẽ. Các thủ tục hay hàm này độc lập với nhau, có dữ liệu riêng, giải quyết những vấn đề riêng, nhưng có thể trao đổi qua lại với nhau bằng các tham biến.

Lập trình cấu trúc làm cho việc kiểm soát chương trình dễ dàng hơn, và do vậy, giải quyết bài toán dễ dàng hơn. Tính hiệu quả của lập trình cấu trúc thể hiện ở khả năng trừu tượng hoá. Trong một chương trình có cấu trúc, người ta chỉ quan tâm về mặt chức năng : một thủ tục hay hàm nào đó có thực hiện được công việc đã cho hay không ? Còn việc thực hiện như thế nào là không quan trọng, chùng nào còn đủ tin cậy.

Mặc dù kỹ thuật thiết kế và lập trình cấu trúc được sử dụng rộng rãi nhưng vẫn bộc lộ những khiếm khuyết. Khi độ phức tạp tăng lên thì sự phụ thuộc của chương trình vào kiểu dữ liệu mà nó xử lý cũng tăng theo. Cấu trúc dữ liệu trong một chương trình có vai trò quan trọng cũng như các phép toán thực hiện trên chúng. Một khi có sự thay đổi trên một kiểu dữ liệu thì một thủ tục nào đó tác động lên kiểu dữ liệu này cũng phải thay đổi theo.

Khiếm khuyết trên cũng ảnh hưởng đến tính hợp tác giữa các thành viên lập trình. Một chương trình có cấu trúc được giao cho nhiều người thì khi có sự thay đổi về cấu trúc dữ liệu của một người sẽ ảnh hưởng đến công việc của những người khác.

I.3. Lập trình định hướng đối tượng (ĐHĐT)

Lập trình ĐHĐT (oriented-object programming) được xây dựng trên nền tảng của lập trình cấu trúc và trừu tượng hoá dữ liệu (data abstraction).

Chương trình ĐHĐT được thiết kế xung quanh dữ liệu mà nó thao tác chứ không bản thân các thao tác. Tính ĐHĐT làm rõ mối quan hệ giữa dữ liệu và thao tác trên dữ liệu.

Trừu tượng hoá dữ liệu là làm cho việc sử dụng các cấu trúc dữ liệu trở nên độc lập đối với việc cài đặt cụ thể. Ví dụ số dấu chấm động (floating point number) đã được trừu tượng hoá trong mọi ngôn ngữ lập trình. NSD thao tác trên các số dấu chấm động mà không quan tâm đến cách biểu diễn nhị phân trong máy của chúng như thế nào.

Lập trình ĐHĐT liên kết các cấu trúc dữ liệu với các phép toán. Một cấu trúc nào đó thì tương ứng, ta có những phép toán nào đó. Ví dụ : một bản ghi về nhân sự có thể được đọc, cập nhật sự thay đổi và được cất giữ, còn một số phức thì được dùng trong tính toán. Không thể viết số phức lên tệp như một bản ghi nhân sự, cũng không thể cộng trừ nhân chia hai bản ghi nhân sự với nhau như cách của số phức.

Lập trình ĐHĐT đưa vào nhiều thuật ngữ và khái niệm mới, chẳng hạn khái niệm lớp (class), khái niệm kế thừa (inheritence).

Ưu điểm của lập trình ĐHĐT là làm cho việc phát triển phần mềm nhanh chóng hơn với khả năng dùng lại các chương trình cũ. Một lớp mới được xem như lớp suy diễn, có thể được kế thừa cấu trúc dữ liệu và các phương pháp của lớp gốc hoặc lớp cơ sở.

Một trong những ngôn ngữ lập trình ĐHĐT được nói đến là SMALLTALK, được phát triển năm 1980 tại Xerox Palo Alto Recearch Center (PARC). Hiện nay, nhiều ngôn ngữ lập trình thông dụng cũng được trang bị thêm khả năng ĐHĐT, như là C++, Delphi, v.v...

I.4. Lập trình trực quan

Lập trình trực quan (visual programming) được phát triển trên nền tảng của lập trình ĐHĐT. Khi thiết kế chương trình, người lập trình nhìn thấy ngay kết quả qua từng thao tác và giao diện người dùng (user interface) khi chương trình được thực hiện. Người lập trình có thể dễ dàng chỉnh sửa về màu sắc, kích thước, hình dáng và các xử lý thích hợp lên các đối tượng có mặt trong giao diện.

Các ngôn ngữ lập trình trực quan thông dụng hiện nay thường được phát triển trong môi trường Microsoft Windows, như Visual Basic, Visual C++, Visual Foxpro, Java. v.v...

I.5. Những tư tưởng cách mạng trong lập trình

Lập trình là một trong những lĩnh vực khó nhất của toán học ứng dụng. Người ta coi lập trình là một khoa học nhằm đề xuất những nguyên lý và phương pháp để nâng cao năng suất lao động của lập trình viên. Năng suất ở đây được hiểu là tính đúng đắn của chương trình, tính dễ đọc, dễ sửa, tận dụng hết khả năng của thiết bị mà không phụ thuộc vào thiết bị.

Thực chất của quá trình lập trình là người ta không lập trình trên một ngôn ngữ cụ thể mà lập trình hướng tới nó. Chương trình phải được viết dưới dạng các thao tác có cấu trúc trên các đối tượng có cấu trúc và các mệnh đề nhằm khẳng định tính đúng đắn của kết quả.

Những tư tưởng cách mạng trong lập trình thể hiện ở hai điểm sau :

- Chương trình và lập trình viên trở thành đối tượng nghiên cứu của lý thuyết lập trình.
- Làm thế nào để làm chủ được sự phức tạp của hoạt động lập trình ?

II. Các phương diện của công nghệ phần mềm

II.1. Công nghệ phần mềm là gì?

Theo từ điển *Computer Dictionary* của Microsoft Press® (1994), Software Engineering : The design and development of sofware (computer program), from concept through execution and documentation.

Từ điển Larousse (1996) định nghĩa chi tiết hơn : *Công nghệ phần mềm là tập hợp các phương pháp, mô hình, kỹ thuật, công cụ và thủ tục liên quan đến các giai đoạn xây dựng một sản phẩm phần mềm*. Các giai đoạn đó là : *đặc tả* (specifiction), *thiết kế* (design), *lập trình* (programming), *thử nghiệm* (testing), *sửa sai* (debugging), *cài đặt* (setup) để đem vào *ứng dụng* (application), *bảo trì* (maintenance) và *lập hồ sơ* (documentation).

Mục đích chính của công nghệ phần mềm là để sản xuất ra những phần mềm có chất lượng. Chất lượng phần mềm không là một khái niệm đơn giản, bao gồm nhiều yếu tố. Chẳng hạn chương trình chạy nhanh, dễ sử dụng, có tính cấu trúc, dễ đọc dễ hiểu, v.v...

Người ta thường đánh giá theo hai kiểu chất lượng : những yếu tố chất lượng bên ngoài và những yếu tố chất lượng bên trong.

II.2. Những yếu tố chất lượng bên ngoài và bên trong

Những yếu tố chất lượng bên ngoài người dùng có thể nhận biết được, như tốc độ nhanh, chạy ổn định, tính dễ sử dụng, dễ thích nghi với những thay đổi (tính mở rộng), tính công thái học (ergonomy, human factor), v.v...

Những yếu tố chất lượng bên ngoài của một sản phẩm phần mềm là :

Tính đúng đắn	Khả năng thực hiện chính xác công việc đặt ra.
Tính bền vững	Có thể hoạt động trong những điều kiện bất thường.
Tính có thể mở rộng	Khả năng dễ sửa đổi để thích nghi với những thay đổi mới

Tính sử dụng lại	Khả năng sử dụng lại toàn bộ hay một phần của hệ thống cho những ứng dụng mới.
Tính tương thích	Có thể dễ dàng kết hợp với các sản phẩm phần mềm khác.
Các chất lượng khác	Hiệu quả đối với nguồn tài nguyên của MTĐT như bộ xử lý, bộ nhớ..., dễ chuyển đổi (không phụ thuộc vào cấu hình phần cứng), dễ kiểm chứng và an toàn (được bảo vệ quyền truy nhập), dễ sử dụng, v.v...

Những yếu tố chất lượng bên trong là là tính đơn thể, tính dễ đọc, dễ hiểu mà chỉ những người làm Tin học chuyên nghiệp mới biết được. Yếu tố chất lượng bên ngoài là mục đích cuối cùng nhưng yếu tố chất lượng bên trong lại là mấu chốt để đạt được những yếu tố chất lượng bên ngoài.

II.3. Sản phẩm phần mềm là gì ?

Mặc dù người ta không định nghĩa nhưng khái niệm sản phẩm phần mềm được hiểu như là một hệ thống chương trình thực hiện một nhiệm vụ tương đối độc lập nhằm phục vụ cho một ứng dụng cụ thể trong cuộc sống của con người (và có thể được thương mại hoá). Ví dụ các sản phẩm phần mềm :

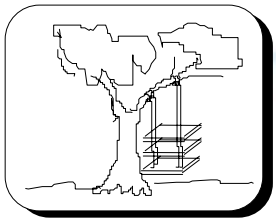
- Hệ điều hành : MS – DOS, OS/2, Unix, MAC OS...
- Hệ điều hành mạng máy tính : Unix, Novell Netware, Windows NT... và các ứng dụng trên mạng LAN, WAN, Internet/Intranet (các Browsers, các dịch vụ khai thác Internet...).
- Các ngôn ngữ lập trình (chương trình dịch) : Turbo Pascal, Turbo C, C^{++}...
- Hệ quản trị cơ sở dữ liệu : Microsoft Foxpro, Microsoft Access, Oracle, Paradox...
- Microsoft Windows và các ứng dụng trên Windows.
- Các trò chơi (games).
- Các phần mềm trợ giúp thiết kế (CAD, Designers...), trợ giúp giảng dạy...
- Các hệ chuyên gia, trí tuệ nhân tạo, người máy, v.v...
- Các chương trình phòng chống virus, v.v...

Dưới đây là bảng tóm tắt quá trình tiến hóa của sản phẩm phần mềm :

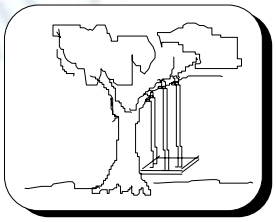
Thời kỳ đầu tiên 1950 – 1960	Xử lý theo lô (Batch processing) Phần mềm được viết theo đơn đặt hàng
Thời kỳ thứ hai 1960 – 1970	Đa người dùng (Multiusers) Thời gian thực (Real time) Cơ sở dữ liệu (Database) Phần mềm sản phẩm

Thời kỳ thứ ba 1970 – 1990	Hệ thống xử lý phân bổ (Distributed processing system) Thông minh (Intelligence) Phần cứng giá thành hạ Hiệu quả tiêu thụ
Thời kỳ thứ tư 1990 trở đi	Hệ thống để bàn (Desktop – Personal – Notebook computers) Lập trình hướng đối tượng (Object oriented programming) Lập trình trực quan (Visual programming) Hệ chuyên gia (Expert system) Mạng thông tin toàn cầu (Worldwide communication network) Xử lý song song (Paralell processing) ...

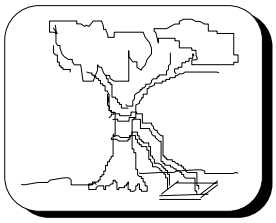
Sau đây là một tranh vui về quá trình tạo ra một sản phẩm phần mềm đã khá quen thuộc đối với những người làm Tin học từ hơn 20 năm nay (theo J. CLAVIER, "Diriger un projet informatique", Edition J. C. I. Inc, Canada 1993) :



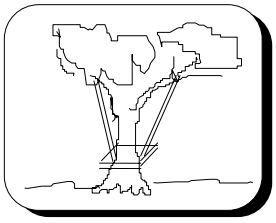
1. Người đặt hàng
Ví dụ : Công ty Công viên



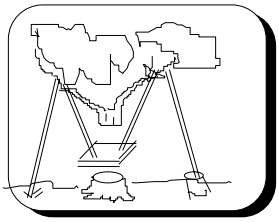
2. Thiết kế của chủ trì đề tài



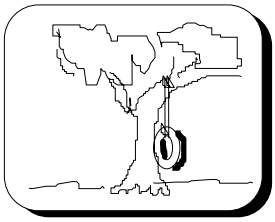
3. Sản phẩm của người lập trình



4. Sau khi sửa sai với nhiều sáng kiến cải tiến



5. Triển khai cho khách hàng



6. Ước mơ của người sử dụng !

Hình 1.1. Quá trình tạo ra một sản phẩm phần mềm

III. Những nội dung cơ bản của CNPM

III.1. Tổng quan về công nghệ phần mềm

Công nghệ phần mềm đặc trưng bởi tập hợp các phương pháp để phát triển một chương trình (phần mềm nói chung). Sự phát triển một chương trình, hay *tiến trình phần mềm* (software process), không chỉ nằm ở chỗ lập trình theo nghĩa hẹp mà còn là việc triển khai các giai đoạn dẫn đến lập trình. Tập hợp các giai đoạn này được gọi là *chu kỳ sống* (hay vòng đời) của phần mềm (life cycle).

Với một dự án Tin học lớn, nhiều người lập trình tham gia được chia thành nhóm, mỗi nhóm phụ trách giải quyết một phần của dự án. Người phụ trách dự án có nhiệm vụ phân bổ công việc cho từng nhóm, đảm bảo mối liên lạc giữa các nhóm, kiểm tra tiến trình phát triển của dự án, chất lượng của sản phẩm phần mềm khi hoàn tất.

Tiến trình phát triển phần mềm gồm 3 giai đoạn chính là *xác định*, *phát triển* và *bảo trì*, không phụ thuộc vào miến áp dụng, độ lớn và độ phức tạp của dự án phát triển, cũng như mô hình được lựa chọn.

Giai đoạn xác định :

Giai đoạn này trả lời câu hỏi *là cái gì* ? (What?) và khi nào (When?) về dữ liệu (thông tin) cần xử lý, mục đích chức năng và môi trường phát triển. Gồm 3 bước :

- Phân tích hệ thống.

- Lập kế hoạch dự án phần mềm.

- Phân tích yêu cầu thực tiễn.

Giai đoạn phát triển :

Giai đoạn này trả lời câu hỏi *làm như thế nào* ? (How?). Gồm 3 bước :

- Thiết kế phần mềm : Sử dụng các công cụ đặc tả và lập trình cấu trúc.

- Chọn công cụ hoặc các ngôn ngữ lập trình để tiến hành viết chương trình.

- Kiểm thử (phát hiện sai sót, nhầm lẫn...).

Giai đoạn bảo trì :

Giai đoạn này tập trung vào các thay đổi (Modify). Có 3 kiểu thay đổi :

- *Sửa đổi* : Dù phần mềm có chất lượng tốt, vẫn tồn tại những khiếm khuyết từ việc sử dụng của khách hàng (người sử dụng). Bảo trì sửa đổi làm thay đổi phần mềm, khắc phục khiếm khuyết.

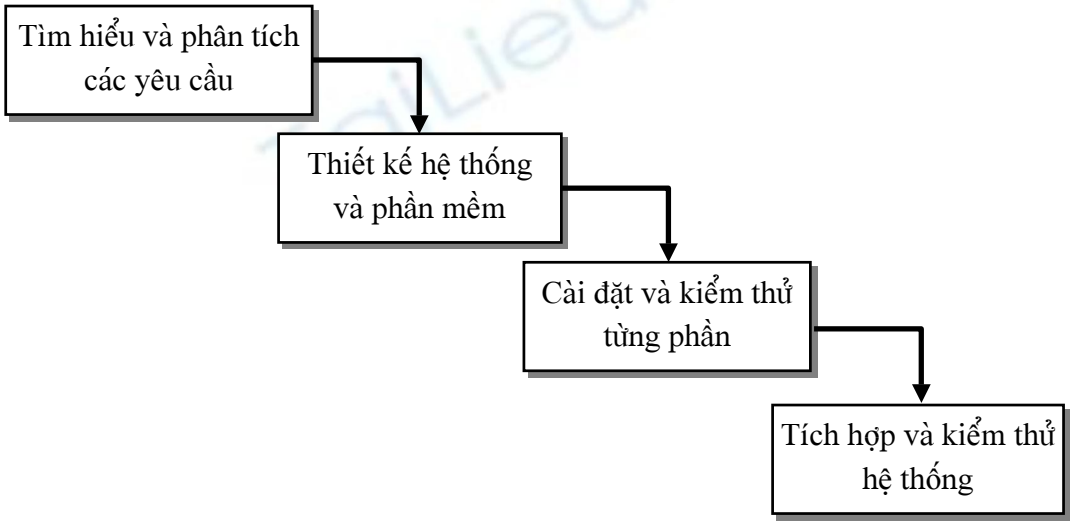
- *Thích nghi* : Nhằm làm phần mềm thích nghi với môi trường phần cứng, như CPU, OS, các thiết bị ngoại vi.

- *Nâng cao* : Khách hàng tìm ra những chức năng phụ của phần mềm. Bảo trì hoàn thiện để mở rộng phần mềm ra ngoài những chức năng vốn có.

III.2. Chu kỳ sống của phần mềm

Có nhiều mô hình khác nhau để thể hiện một chu kỳ sống (life cycle). Sau đây là một chu kỳ sống kiểu cổ điển theo mô hình thác nước ("waterfall" model) gồm các giai đoạn như sau :

- Tìm hiểu và phân tích các yêu cầu (RAD – Requirements analysis and definition)
- Thiết kế hệ thống và phần mềm (SSD – System and software design)
- Cài đặt và kiểm thử từng phần (IUT – Inplementtation and Unit testing)
- Tích hợp và kiểm thử hệ thống (IST – Integrgion and system testing)



Hình 1.2. Mô hình thác nước

Dẫu rằng mô hình thác nước trên đây có ích lợi trong việc quản lý (management), lập kế hoạch và lập báo cáo tiến độ phát triển phần mềm nhưng chỉ thích hợp với một lớp hệ thống phần mềm nào đó mà thôi, không phù hợp với các hoạt động đã chỉ ra trong mô hình.

Tiến trình phần mềm gồm các hoạt động phức tạp và biến động mà không thể biểu diễn trên một mô hình đơn giản. Những mô hình tốt về tiến trình phần mềm vẫn còn là chứng chủ đề nghiên cứu. Hiện nay, các mô hình tổng quát khác nhau hay tính thực dụng của sự phát triển phần mềm, gắn bó chặt chẽ với nhau.

Mô hình thác nước nguyên thuỷ (original) là một trong những mô hình tổng quát mang tính thực dụng sâu sắc.

Sau đây là một số tiếp cận :

1. *Tiếp cận thác nước* (the waterfall approach) : Bao gồm các giai đoạn đặc tả yêu cầu, thiết kế phần mềm, cài đặt, kiểm thử, v.v..., sau mỗi giai đoạn là sự kết thúc (signed-off) và tiếp tục giai đoạn tiếp theo.
2. *Lập trình thăm dò* (exloratory programming) : Cho phép tăng nhanh quá trình để dẫn đến tính thỏa đáng của hệ thống. Lập trình thăm dò thường được áp dụng trong lĩnh vực trí tuệ nhân tạo, khi NSD không thể định hình được các đặc tả yêu cầu. NSD quan tâm đến tính thỏa đáng của kết quả hơn là tính chính xác.
3. *Bản mẫu* (prototyping) : Tương tự tiếp cận lập trình thăm dò. Pha đầu tiên bao gồm phát triển một chương trình cho phép thử nghiệm. Tuy nhiên, mục đích của phát triển là thiết lập các yêu cầu hệ thống. Sau đó là sự cài đặt lại phần mềm để đưa đến hệ thống chất lượng - sản phẩm.

Hình 1.3. Tiếp cận kiểu bản mẫu

4. *Biến đổi hình thức* (formal transformation) : Là sự biến đổi các đặc tả hình thức (formal specification) của hệ thống phần mềm đang xét để thành một chương trình khả thi nhưng bảo toàn được tính chính xác (correctness - preserving transformations).
5. *Lắp ráp hệ thống từ các thành phần dùng lại được* (system assembly from reusable components). Kỹ thuật này cho phép xây dựng hệ thống từ các thành phần đã có. Tiến trình phát triển hệ thống là sự lắp ráp hơn là sự sáng tạo.

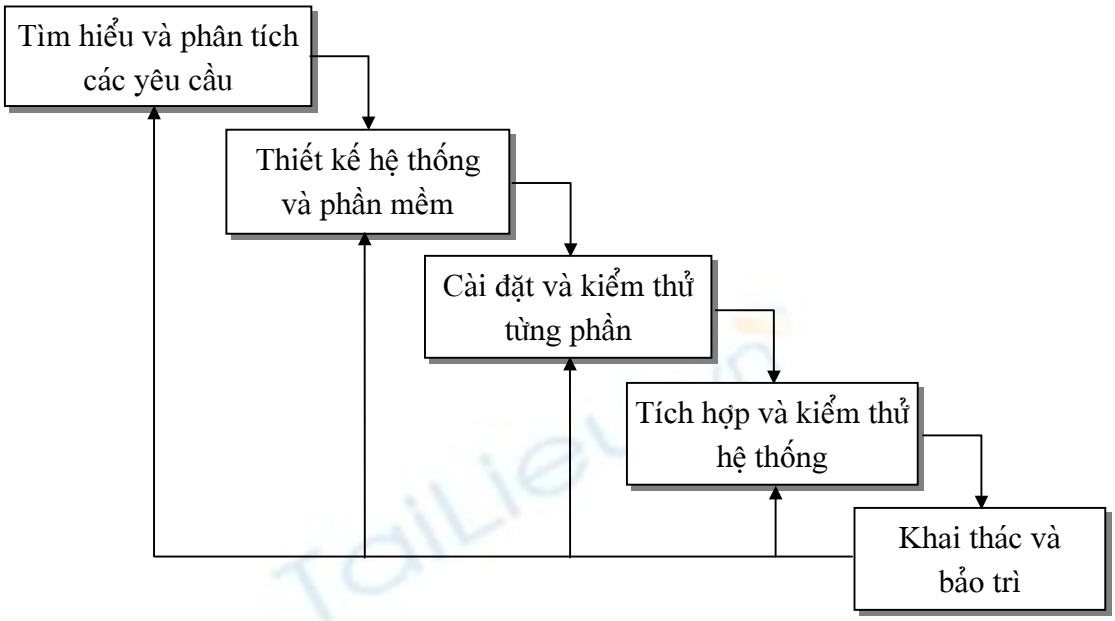
Hiện nay, các tiếp cận 1, 2, 3 được ứng dụng nhiều trong thực tiễn.

Trên thực tế, các giai đoạn phát triển phần mềm không phải rời riêng mà là gối lên nhau (overlap) và thông tin được cung cấp lẫn nhau.

Trong khi thiết kế, những vấn đề và các yêu cầu gắn bó với nhau, trong khi lập trình, những vấn đề thiết kế được tìm thấy, v.v... Lúc này, tiến trình phần mềm không đơn giản là một mô hình tuyến tính mà bao gồm một dãy các tương tác của các hoạt động phát triển.

Tuy nhiên, một mô hình chứa các vòng lặp sẽ làm khó khăn cho việc quản lý và báo cáo. Có nhiều dạng mô hình trong tiến trình phần mềm. Sau đây là một số mô hình :

1. Mô hình thác nước cải tiến

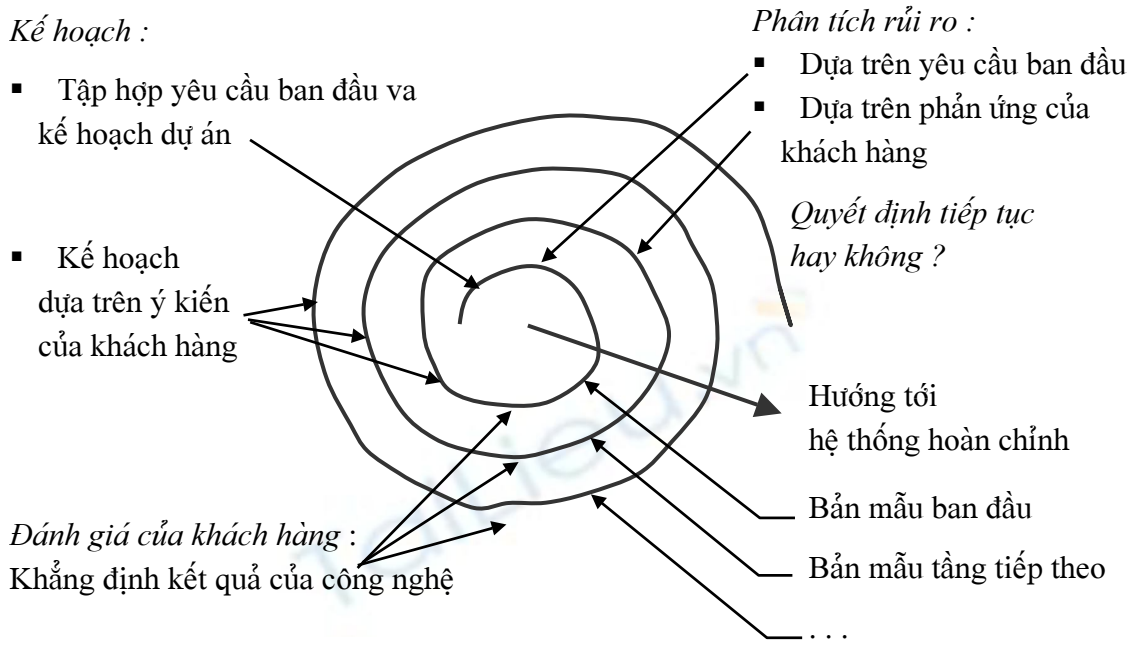


Hình 1.4. Mô hình thác nước cải tiến

1. *Tìm hiểu và phân tích các yêu cầu*: NSD hệ thống và người phát triển hệ thống bàn bạc, trao đổi (consultation) với nhau để thiết lập mục đích, ràng buộc và các dịch vu của hệ thống phần mềm, lĩnh hội được những đòi hỏi của bài toán.
2. *Thiết kế hệ thống và phần mềm* : Tiến trình thiết kế hệ thống phân chia các yêu cầu thành các hệ thống phần cứng, phần mềm và thiết lập một kiến trúc hệ thống toàn bộ (overall system architecture). Việc thiết kế phần mềm bao gồm việc thể hiện các chức năng hệ thống phần mềm (software system functions) để biến đổi thành các chương trình khả thi.
3. *Cài đặt và kiểm thử từng phần* : Trong giai đoạn này, các đơn vị chương trình hay tập hợp các chương trình được kiểm thử lần lượt sao cho thỏa mãn các đặc tả tương ứng.
4. *Tích hợp và kiểm thử hệ thống* : Các đơn vị chương trình được tích hợp và kiểm thử như là một hệ thống đầy đủ để đảm bảo các yêu cầu đặt ra ban đầu. Sau giai đoạn này, hệ thống phần mềm được giao cho khách hàng.
5. *Khai thác và bảo trì* (operation and maintenance) : Đây là một pha dài nhất của chu kỳ sống. Hệ thống được cài đặt và đưa vào sử dụng thực tế. Việc bảo trì bao gồm việc khắc phục những sai sót xảy ra đã không xuất hiện trong các giao đoạn trước đó của chu kỳ sống. Việc tối ưu hóa các dịch vụ của hệ thống được xem như là những yêu cầu mới được phát hiện.

2. Mô hình xoắn ốc

Phát triển trên tính ưu việt của vòng đời cổ điên và bản mẫu, bổ sung nhũng yếu tố còn thiếu và thêm các yếu tố mới, phân tích rủi ro.



Hình 1.5. Mô hình xoắn ốc

Ưu điểm :

Các phiên bản (hay sản phẩm) được hoàn thiện dần theo chiều xoáy ốc từ trong ra ngoài.

Nhược điểm :

- Khó đánh giá chính xác, nhất là khi gặp rủi ro, khó kiểm soát. Do đó khó thuyết phục được các khách hàng lớn
- Mô hình nàuy còn mới, chưa được kiểm nghiệm nhiều trong thực tiễn.

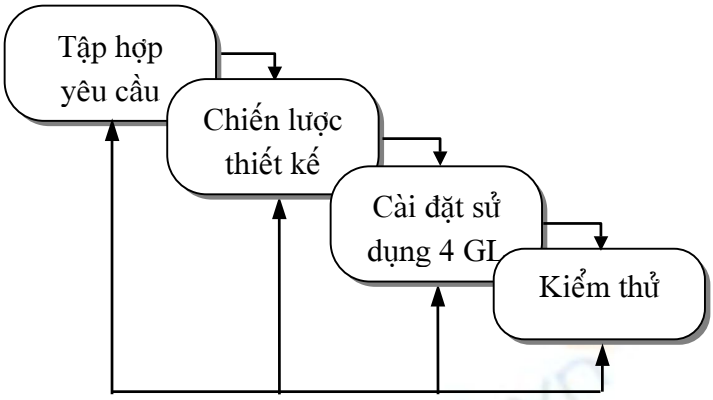
3. Kỹ thuật thế hệ 4 (4th Generation Technology)

Bao gồm các công cụ phần mềm trên cơ sở tự động sản sinh mã chương trình gốc theo nhu cầu của người phát triển :

- Ngôn ngữ phi thủ tục[2] (non procedural language) để truy cập cơ sở dữ liệu.
- Bộ sinh báo cáo.
- Bộ thao tác dữ liệu.

[2] là ngôn ngữ lập trình không tuân theo cách gọi thủ tục hay gọi chương trình con thông thường, không sử dụng các cấu trúc điều khiển, tuần tự, mà dựa trên tập hợp các yếu tố và quan hệ để dẫn về kết quả yêu cầu. Ví dụ ngôn ngữ vấn tin SQL thuộc loại này.

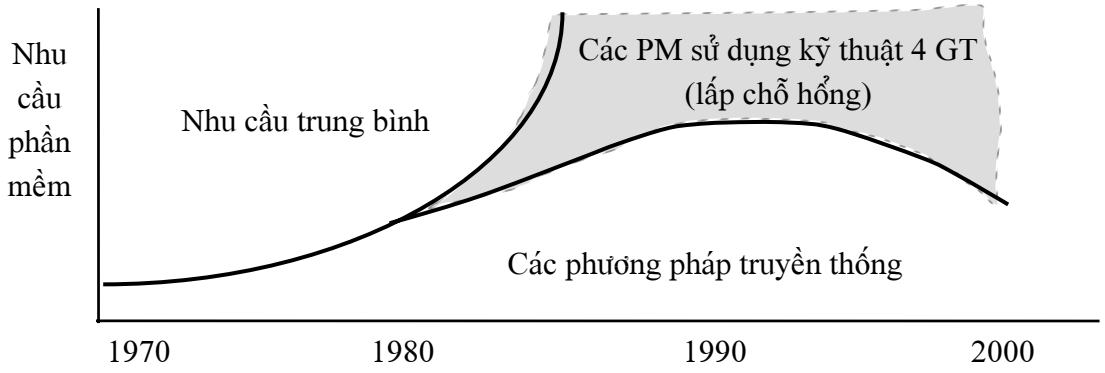
- Bộ tương tác và thiết kế màn hình.
- Bộ sinh chương trình.
- Bảng tính.
- Công cụ đồ họa.



Hình 1.6. Kỹ thuật thế hệ 4

Ưu điểm :

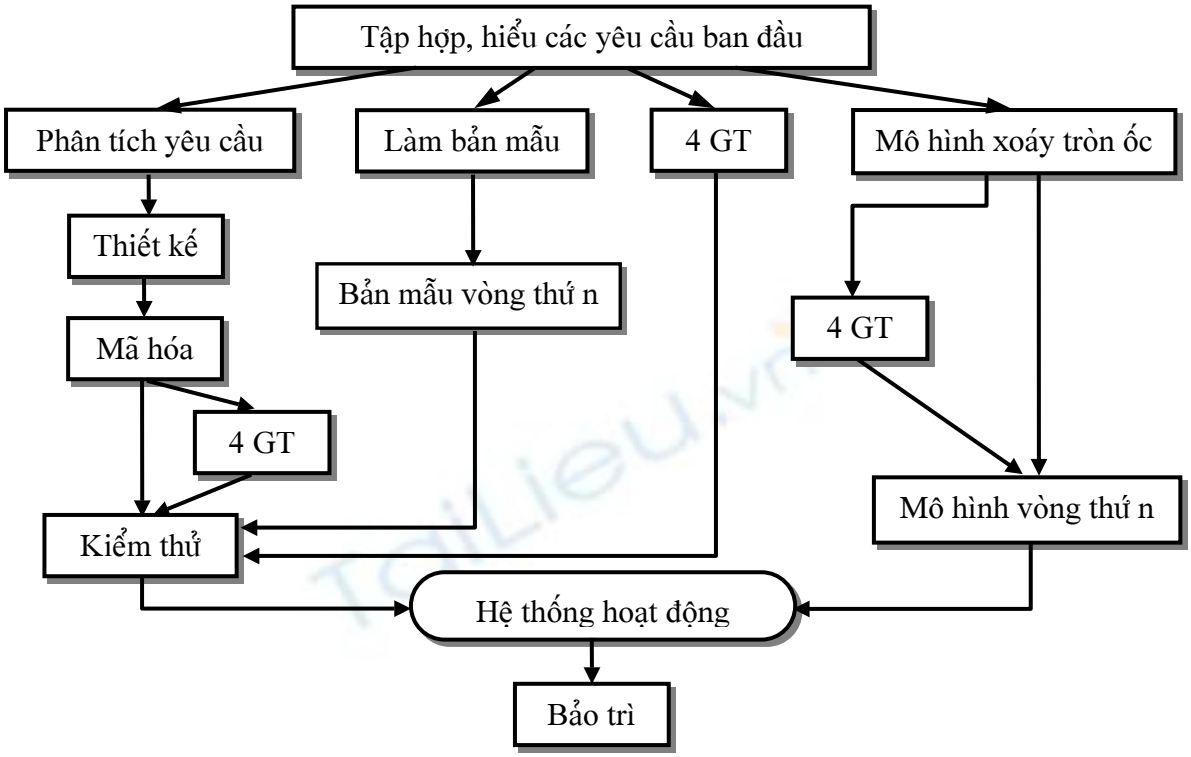
Thường được sử dụng để xây dựng các hệ thông tin và tương lai là các ứng dụng kỹ nghệ phát triển phần mềm thời gian thực.



Hình 1.7. Nhu cầu phần mềm

5. Tích hợp các kỹ thuật

Nhằm tăng cường tính tối ưu trong phát triển phần mềm, người ta có xu hướng tích hợp các kỹ thuật cổ điển, xoáy tròn ốc và 4GT đã nêu.



Hình 1.8. Tích hợp các kỹ thuật

CHƯƠNG 2

Thiết kế phần mềm

I. Nền tảng của thiết kế phần mềm