22520912 - Lê Xuân Nam

22520927 - Lê Ngọc Ngà

22521083 - Đỗ Văn Phong

22520892 - Trần Văn Minh

Câu 9: Mô tả năm luồng công việc (workflows) cốt lõi của quy trình hợp (Unified Process)

Dựa trên sách **"Object-Oriented and Classical Software Engineering"** của **Stephen R. Schach**, năm luồng công việc (workflows) cốt lõi của **quy trình hợp nhất (Unified Process - UP)** là các nhóm hoạt động chính xuyên suốt quá trình phát triển phần mềm. Các luồng công việc này có sự lặp lại và giao thoa qua các pha khác nhau (Khởi tạo, Làm rõ, Xây dựng, Chuyển giao). Dưới đây là mô tả chi tiết về mỗi luồng công việc dựa theo sách:

1. Luồng công việc Yêu cầu (The Requirements Workflow)

- **Mục đích:** Xác định chính xác và đầy đủ những gì khách hàng cần cho sản phẩm phần mềm, đồng thời xây dựng mô hình kinh doanh của khách hàng.
- **Hoạt động:**
    - **Hiểu rõ lĩnh vực (Domain Understanding):** Kỹ sư phần mềm cần hiểu rõ lĩnh vực mà sản phẩm phần mềm sẽ được áp dụng. Ví dụ, nếu khách hàng cần một phần mềm quản lý kho hàng, thì kỹ sư phần mềm cần tìm hiểu về các khái niệm như hàng hóa, nhà cung cấp, kho hàng, luồng hàng hóa,...
    - **Xây dựng mô hình kinh doanh (Building Business Model):** Kỹ sư phần mềm cần mô tả các quy trình kinh doanh chính của khách hàng. Ví dụ, nếu khách hàng là một công ty sản xuất quần áo, thì mô hình kinh doanh có thể bao gồm các quy trình như: thiết kế sản phẩm, sản xuất, đóng gói, vận chuyển, bán hàng,... Mô hình kinh doanh có thể được thể hiện bằng các sơ đồ UML như Use Case Diagram.

- **Xây dựng yêu cầu ban đầu (Initial Requirements):** Dựa trên sự hiểu biết về lĩnh vực và mô hình kinh doanh, kỹ sư phần mềm sẽ xây dựng các yêu cầu ban đầu cho sản phẩm phần mềm.
- **Lặp lại (Iterate):** Quá trình này liên tục diễn ra cho đến khi nhóm phát triển phần mềm và khách hàng đạt được sự đồng thuận về các yêu cầu.
    - Nắm bắt thêm thông tin từ khách hàng.
    - Hoàn thiện mô hình kinh doanh.
    - Điều chỉnh các yêu cầu ban đầu.
    - Đạt được sự đồng ý của khách hàng về các yêu cầu đã được điều chỉnh.
- **Xác định đối tượng (Identifying Actors):** Xác định các đối tượng (actors) tương tác với sản phẩm phần mềm. Đối tượng có thể là người dùng (user), hệ thống khác (another system), hoặc các thiết bị (device).
- **Xây dựng Use Case:** Dựa trên các đối tượng đã xác định, nhóm phát triển phần mềm sẽ xây dựng các Use Case. Use Case là một mô hình cho tương tác giữa sản phẩm phần mềm và các đối tượng. Mỗi Use Case mô tả một chuỗi các hành động mà sản phẩm sẽ thực hiện để đáp ứng nhu cầu của đối tượng. Ví dụ, trong trường hợp của một phần mềm ngân hàng, Use Case có thể bao gồm:
    - Đăng nhập (login).
    - Rút tiền (withdraw).
    - Chuyển khoản (transfer).
    - Kiểm tra số dư (check balance).
    - Đóng tài khoản (close account).

- **Thách thức:**
    - **Khách hàng không biết họ cần gì:** Nhiều khách hàng không có hiểu biết đầy đủ về lĩnh vực và công nghệ nên khó có thể nêu rõ nhu cầu của họ.
    - **Khách hàng không diễn đạt nhu cầu một cách chính xác:** Khách hàng có thể sử dụng ngôn ngữ mơ hồ, thiếu rõ ràng hoặc mâu thuẫn khi diễn đạt nhu cầu.
    - **Khách hàng không hiểu rõ hoạt động của doanh nghiệp của mình:** Khách hàng có thể không hiểu rõ các quy trình kinh doanh của mình, dẫn đến việc yêu cầu những thứ không cần thiết hoặc không hiệu quả.

- **Khách hàng yêu cầu những thứ không thực tế hoặc không thể thực hiện được:** Ví dụ, khách hàng có thể yêu cầu một sản phẩm phần mềm có thể hoàn thành trong thời gian quá ngắn hoặc với chi phí quá thấp.
- **Khách hàng thay đổi yêu cầu khi quá trình phát triển sản phẩm đã bắt đầu**: Điều này có thể làm chậm tiến độ phát triển, tăng chi phí và giảm chất lượng sản phẩm.
- **Khó khăn trong việc xác định đối tượng:** Việc xác định các đối tượng (actors) tương tác với sản phẩm phần mềm có thể gặp khó khăn, đặc biệt là trong trường hợp sản phẩm có nhiều đối tượng liên quan phức tạp.
- **Xây dựng Use Case khó:** Xây dựng Use Case đòi hỏi sự hiểu biết sâu sắc về lĩnh vực, kỹ năng phân tích và khả năng mô hình hóa các tương tác.

2. Luồng công việc Phân tích (The Analysis Workflow)

- **Mục tiêu:**
  - Mục tiêu chính của luồng công việc này là phân tích và tinh chỉnh các yêu cầu đã thu thập được trong luồng công việc Yêu cầu (Requirements Workflow) để đảm bảo sự rõ ràng, đầy đủ và nhất quán.
  - Mục tiêu này giúp nhóm phát triển phần mềm hiểu rõ và sâu sắc các yêu cầu của khách hàng để có thể thiết kế và triển khai sản phẩm phần mềm một cách chính xác, hiệu quả và dễ dàng bảo trì.
- **Hoạt động:**
  - **Phân tích các yêu cầu (Analyze Requirements):**
    - **Xác định và loại bỏ các lỗi, mâu thuẫn và chỗ thiếu sót:** Nhóm phát triển phần mềm cần xem xét kỹ lưỡng các yêu cầu, để xác định và loại bỏ các lỗi, sự mâu thuẫn và các chỗ thiếu sót. Ví dụ:
      - Lỗi: Yêu cầu có thể sai lệch hoặc mâu thuẫn với các yêu cầu khác. Ví dụ, khách hàng yêu cầu một sản phẩm phần mềm có thể xử lý tối đa 1000 đơn hàng mỗi ngày, nhưng đồng thời yêu cầu sản phẩm phải có thể xử lý 2000 đơn hàng mỗi giờ.
      - Mâu thuẫn: Các yêu cầu có thể mâu thuẫn với nhau. Ví dụ, khách hàng yêu cầu sản phẩm phần mềm phải

dễ sử dụng, nhưng đồng thời yêu cầu sản phẩm phải có tính năng phức tạp.
- Thiếu sót: Các yêu cầu có thể thiếu sót một số thông tin quan trọng. Ví dụ, khách hàng yêu cầu một sản phẩm phần mềm có thể hiển thị thông tin về khách hàng, nhưng không nói rõ những thông tin nào cần hiển thị.

- **Phân tích các yêu cầu:** Kỹ sư phần mềm cần phân tích các yêu cầu để hiểu rõ hơn về nhu cầu của khách hàng. Điều này có thể bao gồm việc:
  - Xác định các trường hợp sử dụng (use case) cho sản phẩm phần mềm.
  - Phân tích luồng dữ liệu (data flow) của sản phẩm.
  - Xây dựng các mô hình, chẳng hạn như mô hình thực thể - quan hệ (entity-relationship model) để mô tả dữ liệu của sản phẩm.

- **Xây dựng tài liệu mô tả sản phẩm (Documenting the Product):**
  - **Tài liệu mô tả sản phẩm (Specification Document):** Tài liệu này đóng vai trò quan trọng trong việc đảm bảo sự nhất quán giữa các giai đoạn phát triển phần mềm. Nhóm phát triển phần mềm và khách hàng cần đồng ý về nội dung của tài liệu này. Tài liệu này cần phải:
    - Rõ ràng: Dễ hiểu cho khách hàng, những người thường không phải là chuyên gia về công nghệ.
    - Đầy đủ: Bao gồm tất cả các yêu cầu của sản phẩm, bao gồm cả các ràng buộc (constraints).
    - Nhất quán: Không có mâu thuẫn hoặc lỗi logic.
    - Dễ bảo trì: Dễ dàng cập nhật và sửa đổi khi cần thiết.
  - **Sử dụng các kỹ thuật mô tả chính xác (Formal Specification Language):** Ngôn ngữ mô tả chính xác (Formal Specification Language) có thể giúp nhóm phân tích yêu cầu diễn đạt các yêu cầu một cách chính xác và chặt chẽ, đồng thời giảm thiểu các lỗi logic và mâu thuẫn. Tuy nhiên, việc sử dụng ngôn ngữ mô tả chính xác đòi hỏi nhóm phát triển phần mềm phải có kiến thức chuyên môn về toán học và logic.
- **Hoàn thiện tài liệu (Refining Documentation):**

        - Quá trình này liên tục diễn ra cho đến khi nhóm phát triển phần mềm và khách hàng đạt được sự đồng thuận về tài liệu mô tả sản phẩm.
- **Thách thức:**
    - **Tài liệu yêu cầu có thể mơ hồ, thiếu rõ ràng hoặc mâu thuẫn:** Sách nhấn mạnh rằng việc sử dụng ngôn ngữ tự nhiên (natural language) có thể dẫn đến các lỗi logic, mơ hồ và mâu thuẫn trong tài liệu yêu cầu. Do đó, việc sử dụng các kỹ thuật mô tả chính xác (formal specification) có thể là lựa chọn tốt hơn.
    - **Khách hàng có thể không hiểu tài liệu mô tả sản phẩm:** Sách đề cập đến việc khách hàng thường không hiểu rõ các thuật ngữ kỹ thuật được sử dụng trong tài liệu mô tả sản phẩm. Do đó, nhóm phát triển phần mềm cần phải sử dụng ngôn ngữ đơn giản, dễ hiểu khi viết tài liệu.
    - **Việc phân tích có thể dẫn đến những lỗi trong thiết kế sau này**: Sách nhấn mạnh rằng nếu nhóm phân tích yêu cầu không làm việc một cách cẩn thận, các lỗi trong phân tích có thể sẽ dẫn đến lỗi trong thiết kế và triển khai sản phẩm sau này.
- **Sử dụng các công cụ CASE (Computer-Aided Software Engineering):** Sách cũng đề cập đến việc sử dụng các công cụ CASE (Computer-Aided Software Engineering) để hỗ trợ nhóm phân tích yêu cầu. CASE tools là những phần mềm hỗ trợ nhóm phát triển phần mềm trong các công việc như phân tích, thiết kế, mã hóa và kiểm tra sản phẩm. Các công cụ CASE có thể giúp nhóm phát triển phần mềm bằng cách cung cấp các chức năng như:
    - **Vẽ sơ đồ:** Hỗ trợ nhóm phát triển vẽ các sơ đồ, chẳng hạn như sơ đồ luồng dữ liệu (data flow diagram), sơ đồ thực thể - quan hệ (entity-relationship diagram), để mô tả sản phẩm một cách trực quan.
    - **Kiểm tra nhất quán:** Kiểm tra xem các yêu cầu có nhất quán với nhau hay không, đồng thời so sánh các yêu cầu với thiết kế để đảm bảo tính nhất quán giữa hai giai đoạn.
    - **Quản lý tài liệu:** Quản lý các tài liệu liên quan đến sản phẩm phần mềm.
    - **Kiểm tra tính chính xác:** Kiểm tra xem các yêu cầu có hợp lý, khả thi và chính xác hay không.

- Bằng cách sử dụng các kỹ thuật và công cụ phù hợp, nhóm phát triển phần mềm có thể thực hiện luồng công việc Phân tích (Analysis Workflow) một cách hiệu quả, dẫn đến việc xây dựng sản phẩm phần mềm chất lượng cao, dễ bảo trì và đáp ứng đầy đủ nhu cầu của khách hàng.

3. Luồng công việc Thiết kế (Design Workflow)

Trong quy trình thống nhất (Unified Process), luồng Thiết kế (Design) là một phần quan trọng của giai đoạn phát triển phần mềm, nhằm tạo ra một mô hình chi tiết cho hệ thống phần mềm. Dưới đây là mô tả về luồng thiết kế trong quy trình thống nhất:

- Mục Tiêu của Luồng Thiết Kế :Luồng thiết kế tập trung vào việc chuyển đổi các yêu cầu hệ thống thành một mô hình thiết kế chi tiết và có thể thực hiện được. Mục tiêu là tạo ra cấu trúc phần mềm rõ ràng, bao gồm cả các lớp, đối tượng, và các mối quan hệ giữa chúng.
- Các Giai Đoạn Chính trong Luồng Thiết Kế:

    ○ Thiết kế Tổng Quát (Architectural Design): Tạo ra kiến trúc tổng thể cho hệ thống, bao gồm cấu trúc các thành phần chính, mô hình phân phối, và các kết nối giữa các thành phần. Xác định các mẫu thiết kế (design patterns) và các nguyên lý thiết kế cơ bản sẽ được áp dụng.

    ○ Thiết kế Chi Tiết (Detailed Design): Xây dựng các mô hình chi tiết cho từng phần của hệ thống, bao gồm cấu trúc lớp, giao diện và các mối quan hệ giữa các lớp. Định nghĩa các thuộc tính, phương thức của lớp, và cách thức giao tiếp giữa các đối tượng.

    ○ Thiết kế Giao Diện (Interface Design): Thiết kế giao diện người dùng và giao diện giữa các hệ thống hoặc các thành phần khác. Đảm bảo rằng giao diện dễ sử dụng và đáp ứng yêu cầu của người dùng cũng như các yêu cầu kỹ thuật.

    ○ Thiết kế Dữ Liệu (Data Design): Xác định cấu trúc dữ liệu và cách dữ liệu sẽ được lưu trữ và truy xuất. Thiết kế cơ sở dữ liệu và các đối tượng dữ liệu cần thiết cho hệ thống.

- Các Kỹ Thuật và Công Cụ Sử Dụng

○ Mô Hình UML (Unified Modeling Language): Sử dụng các sơ đồ UML như sơ đồ lớp, sơ đồ đối tượng, sơ đồ tuần tự để biểu diễn thiết kế.

○ Mẫu Thiết Kế (Design Patterns): Áp dụng các mẫu thiết kế như Singleton, Observer, Factory, và Decorator để giải quyết các vấn đề thiết kế phổ biến.

○ Kiểm Tra Thiết Kế: Đánh giá thiết kế để đảm bảo tính nhất quán, khả năng mở rộng và dễ bảo trì.

- Tài Liệu và Giao Tiếp

○ Tài Liệu Thiết Kế: Lập tài liệu thiết kế chi tiết để các thành viên trong nhóm phát triển có thể hiểu và triển khai thiết kế.

○ Giao Tiếp với Các Bên Liên Quan: Thường xuyên trao đổi với các bên liên quan để đảm bảo rằng thiết kế đáp ứng đúng yêu cầu và mục tiêu của dự án.

- Phản Hồi và Cải Tiến

○ Đánh Giá và Phản Hồi: Tiến hành các đánh giá thiết kế để nhận phản hồi và thực hiện các điều chỉnh cần thiết.

○ Cải Tiến Thiết Kế: Dựa trên phản hồi và các vấn đề phát hiện trong quá trình kiểm thử để cải tiến thiết kế.

=> Luồng thiết kế trong quy trình thống nhất là một phần quan trọng để đảm bảo rằng phần mềm được phát triển một cách có hệ thống và có thể đáp ứng các yêu cầu của người dùng.

4. Luồng công việc thực thi (The Implementation Workflow)

- **Phân chia & Thực hiện song song**:
  - Một sản phẩm phần mềm lớn được chia thành các hệ thống con nhỏ hơn, dễ quản lý hơn.
  - Các hệ thống con này sau đó được thực hiện song song bởi các nhóm lập trình riêng biệt.
  - Mỗi hệ thống con lại được chia nhỏ thành các thành phần hoặc mã nguồn cụ thể.

- **Thực hiện cá nhân**:
    - Các lập trình viên nhận được các thông số kỹ thuật thiết kế chi tiết cho từng thành phần mã nguồn.
    - Thông tin này thường đủ để họ có thể triển khai thành phần đó một cách hiệu quả.
    - Nếu có khó khăn, lập trình viên sẽ tham khảo ý kiến của nhà thiết kế chịu trách nhiệm.
    - Tuy nhiên, các lập trình viên cá nhân có tầm nhìn hạn chế về thiết kế kiến trúc tổng thể.
- **Tích hợp & Kiểm thử**:
    - Khi các thành phần mã nguồn riêng lẻ được hoàn thành, chúng sẽ được tích hợp với nhau.
    - Kiểm thử được thực hiện tại mỗi điểm tích hợp để đảm bảo chức năng tổng thể.
    - Quá trình này giúp phát hiện các vấn đề tiềm ẩn từ tương tác giữa các thành phần, những vấn đề có thể không được phát hiện trong quá trình triển khai cá nhân.
- **Vấn đề của các lỗi thiết kế**:
    - Nếu có lỗi xảy ra sau khi tích hợp, thường lập trình viên thực hiện thành phần lỗi sẽ bị quy trách nhiệm.
    - Tuy nhiên, nguyên nhân có thể nằm ở thiết kế kiến trúc thay vì ở mã nguồn cụ thể.
    - Việc lập trình viên có tầm nhìn hạn chế về thiết kế tổng thể khiến họ khó dự đoán các vấn đề tiềm tàng.
- **Tầm quan trọng của sự chính xác trong thiết kế**:
    - The Implementation Workflow nhấn mạnh vai trò quan trọng của sự chính xác trong thiết kế đối với sự thành công của dự án.
    - Các lỗi trong thiết kế kiến trúc có thể dẫn đến các vấn đề tích hợp và làm trì hoãn dự án.
    - Quy trình kiểm thử chặt chẽ là cần thiết để phát hiện lỗi thiết kế và đảm bảo tính toàn vẹn của hệ thống tổng thể.

5. Luồng công việc Kiểm tra (The Test Workflow)

- **Requirements Artifacts**
    - Khả năng truy xuất (traceability) là yếu tố quan trọng để đảm bảo các tài liệu yêu cầu có thể kiểm tra được trong suốt vòng đời của phần mềm.

    - Cần phải có khả năng truy vết từ các tài liệu phân tích, thiết kế, triển khai ngược lại tới tài liệu yêu cầu.
    - Nếu yêu cầu được trình bày một cách phương pháp, đánh số, tham chiếu chéo, và lập chỉ mục hợp lý, sẽ giúp nhà phát triển dễ dàng truy vết và đảm bảo phản ánh đúng yêu cầu của khách hàng.
    - Nhóm Đảm bảo chất lượng phần mềm (SQA) cũng được hỗ trợ đơn giản hóa công việc khi kiểm tra nhờ khả năng truy xuất này.
- **Analysis Artifacts**
    - **Nguồn gốc lỗi:** Một nguồn lớn gây lỗi trong phần mềm là các lỗi trong tài liệu yêu cầu không được phát hiện cho đến khi phần mềm được cài đặt và sử dụng.
    - **Kiểm tra tài liệu phân tích:** Nhóm phân tích và nhóm Đảm bảo chất lượng (SQA) cần kiểm tra kỹ lưỡng tài liệu phân tích để đảm bảo rằng các yêu cầu khả thi, như phần cứng đủ nhanh hoặc dung lượng lưu trữ phù hợp.
    - **Review (đánh giá):** Cách tốt nhất để kiểm tra tài liệu phân tích là thực hiện đánh giá, với sự tham gia của nhóm phân tích và đại diện khách hàng, do một thành viên SQA điều hành.
    - **Walkthroughs và Inspections:** Là hai kỹ thuật đánh giá phần mềm:
        - Walkthroughs: Là quá trình lập trình viên hoặc người phát triển hướng dẫn nhóm đánh giá qua từng phần của tài liệu hoặc mã nguồn, nhằm phát hiện lỗi hoặc cải thiện chất lượng. Đây là hình thức kiểm tra không chính thức và mang tính hướng dẫn.
        - Inspections: Là quy trình đánh giá chính thức hơn, trong đó nhóm đánh giá xem xét tài liệu hoặc mã nguồn một cách kỹ lưỡng để tìm lỗi, đặc biệt tập trung vào tuân thủ yêu cầu và tiêu chuẩn. Inspections thường có cấu trúc và quy trình rõ ràng hơn so với walkthroughs.
    - **Kiểm tra kế hoạch và ước lượng chi tiết:** Sau khi khách hàng phê duyệt các yêu cầu, cần kiểm tra kỹ lưỡng kế hoạch phát triển và ước tính thời gian, chi phí. Nên có nhiều ước tính độc lập về thời gian và chi phí để so sánh và điều chỉnh.
    - **Kiểm tra tài liệu SPMP:** Cách kiểm tra tài liệu SPMP cũng giống như kiểm tra tài liệu phân tích, và khi mọi thứ đều thỏa đáng, khách hàng sẽ cho phép dự án tiếp tục.

- **Design Artifacts**
  - **Khả năng truy xuất (traceability):** Mỗi phần của thiết kế phải có thể liên kết với một tài liệu phân tích. Việc tham chiếu chéo này giúp kiểm tra xem thiết kế có phù hợp với yêu cầu và đảm bảo mọi phần của yêu cầu đều được phản ánh trong thiết kế.
  - **Đánh giá thiết kế:** Quá trình đánh giá thiết kế tương tự như đánh giá tài liệu yêu cầu, nhưng thường không có mặt khách hàng do tính chất kỹ thuật của thiết kế.
  - **Kiểm tra bởi nhóm thiết kế và SQA:** Nhóm thiết kế và SQA kiểm tra toàn bộ thiết kế cũng như từng phần riêng lẻ để đảm bảo tính chính xác. Các lỗi cần chú ý bao gồm lỗi logic, lỗi giao diện, thiếu xử lý ngoại lệ, và quan trọng nhất là không tuân thủ yêu cầu.
  - **Cần lưu ý lỗi từ giai đoạn phân tích:** Nhóm đánh giá cũng cần cảnh giác với khả năng một số lỗi phân tích chưa được phát hiện trong giai đoạn trước.
  - **Quy trình đánh giá chi tiết:** Quá trình đánh giá thiết kế (design review) giúp kiểm tra tính đúng đắn của thiết kế, tập trung vào việc phát hiện các lỗi như lỗi logic, lỗi giao diện, thiếu xử lý ngoại lệ, và không tuân thủ yêu cầu. Nhóm thiết kế và SQA cùng tham gia đánh giá, nhưng thường không có mặt khách hàng.
- **Implementation Artifacts**
  - **Kiểm tra không chính thức:** Mỗi thành phần được kiểm tra khi lập trình (desk checking) và sau đó chạy thử với các test case. Lập trình viên thực hiện kiểm tra này.
  - **Kiểm tra đơn vị (unit testing):** Nhóm SQA thực hiện kiểm tra đơn vị một cách có phương pháp sau khi lập trình viên hoàn thành kiểm tra không chính thức.
  - **Review mã nguồn:** Là kỹ thuật quan trọng giúp phát hiện lỗi lập trình, do lập trình viên hướng dẫn nhóm review (bao gồm SQA) qua mã nguồn.
  - **Tích hợp các thành phần:** Sau khi lập trình xong, các thành phần được tích hợp và kiểm tra. Quá trình tích hợp có thể từ dưới lên (bottom-up) hoặc từ trên xuống (top-down), và mỗi phương pháp có ưu nhược điểm riêng.
  - **Kiểm tra tích hợp:** Mục tiêu là kiểm tra sự kết hợp giữa các thành phần để đảm bảo sản phẩm đạt yêu cầu. Đặc biệt quan trọng là kiểm tra giao diện giữa các thành phần.

- **Kiểm tra sản phẩm:** Sau khi tích hợp, nhóm SQA kiểm tra tính năng sản phẩm toàn bộ và các ràng buộc trong yêu cầu, như thời gian phản hồi. Kiểm tra cả độ ổn định của sản phẩm với dữ liệu lỗi.
- **Kiểm tra chấp nhận (acceptance testing):** Sản phẩm được bàn giao cho khách hàng kiểm tra trên phần cứng thực tế và dữ liệu thực. Sản phẩm chỉ được coi là hoàn thành khi vượt qua bài kiểm tra này.
- **Alpha và beta releases:** Với phần mềm thương mại (COTS), phiên bản alpha và beta được cung cấp cho các khách hàng tiềm năng để kiểm tra thực tế. Phiên bản alpha thường chứa nhiều lỗi, còn beta là phiên bản gần hoàn thiện.