LỜI CẢM ƠN

Trước hết em xin được bày tỏ lòng biết ơn sâu sắc đối với thầy giáo hướng dẫn Thạc sĩ Nguyễn Trọng Thể, Khoa Công Nghệ Thông Tin - Trường Đại học Dân lập Hải Phòng; cô giáo hướng dẫn Thạc sĩ Đào Thị Kiên, Khoa Công Nghệ Thông tin - Trường Cao đẳng Cộng đồng Hải Phòng đã tận tình giúp đỡ, chỉ bảo em trong những năm học qua và đã dành rất nhiều thời gian quí báu để giúp em hoàn thành báo cáo thực tập được giao.

Em xin gửi lời cảm ơn đến Ban giám hiệu, các thầy cô giáo của Trường Đại học Dân lập Hải Phòng đã giảng dạy chúng em trong suốt quãng thời gian qua, cung cấp cho chúng em những kiến thức chuyên môn cần thiết và quý báu giúp chúng em hiểu rõ hơn các lĩnh vực đã nghiên cứu để hoàn thành đề tài được giao .

Xin cảm ơn các bạn bè và gia đình đã động viên cổ vũ, đóng góp ý kiến, trao đổi, động viên trong suốt quá trình học cũng như làm tốt nghiệp, giúp em hoàn thành đề tài đúng thời hạn.

Hải Phòng, tháng 7 năm 2009

Sinh viên

Vũ Thị Tuyết Minh

MỤC LỤC

MỞ ĐẦU

Chúng ta đang bước vào kỷ nguyên của công nghệ thông tin. Máy tính với hàng loạt hệ thống các phần mềm đang ngày càng trở nên thân thiện, cần thiết và không thể thiếu trong mọi lĩnh vực hoạt động của con người.

Phần mềm ngày càng được hoàn thiện, nâng cao chất lượng và phát triển với kích thước rất lớn. Nhưng bên cạnh đó, sự bùng nổ thông tin làm cho một loạt các hoạt động luôn bị thay đổi. Đó là sự thay đổi của môi trường, thay đổi của công nghệ, thay đổi của nghiệp vụ. Để khắc phục những sự thay đổi đó người ta thường đưa hệ thống vào bảo trì. Công việc bảo trì phần mềm được xem xét như là một pha tốn kém nhất trong các pha trong vòng đời của một phần mềm. Người ta ước tính chi phí cho nó xấp xỉ 70% tổng công sức chi phí trong sự phát triển phần mềm[1]. Nhưng nếu xây dựng lại hệ thống mới thì chưa phải là giải pháp hay, vì khi đó ta phải bỏ đi cả những phần rất hữu dụng trong phần mềm. Hơn thế nữa, chi phí cho việc làm ra phần mềm mới là rất tốn kém.

Làm thế nào để hàng loạt những hệ thống phần mềm lớn, cũ, đang hoạt động thích nghi được với những thay đổi với mức chi phí thay đổi chấp nhận được. Tái kỹ nghệ phần mềm chính là một sự trả lời cho câu hỏi đó.

Tái kỹ nghệ phần mềm là hoạt động tiến hóa hệ thống phần mềm để nó có thể tiếp tục được sử dụng cho hiệu quả, giúp ta dễ dàng và đỡ tốn kém hơn trong việc bảo trì sau này.

Những phần mềm đã sử dụng trong một thời gian dài có thể có nhiều nhược điểm như: xây dựng trên ngôn ngữ cũ mà hiện nay không còn dùng nữa, tài liệu viết cho phần mềm này cũng đã bị hỏng và thiếu do việc cất giữ và cập nhật chưa tốt, các tính năng hoạt động bị hạn chế do hoạt động nghiệp vụ đã có những thay đổi, … Giải pháp tốt nhất giúp ta tiếp tục sử dụng phần mềm này là tái kỹ nghệ. Tái kỹ nghệ là giải pháp tốt nhất và cũng có thể nói là giải pháp duy nhất để đạt được mục đích với chi phí rẻ. Hơn thế nữa, nó đảm bảo an toàn cho hoạt động nghiệp vụ của hệ thống đang làm việc.

Về mặt khoa học, tái kỹ nghệ đưa ra một giải pháp tiến hóa hệ thống phần mềm bằng những công cụ và phương tiện mới với quy trình khép kín khá hoàn thiện và tiện

dụng. Về mặt thực tiễn, nó là một hướng giải quyết tốt, vừa đáp ứng nhu cầu tái thiết kế hệ thống cũ, vừa đem lại hiệu quả lớn và thiết thực về mặt kinh tế.

Đồ án đề cập tới việc tái kỹ nghệ phần mềm qua đó minh hoạ sự kết hợp thiết kế hướng đối tượng với công nghệ tái kỹ nghệ hiện có được sử dụng như một quy trình tái kỹ nghệ cho một ứng dụng cho hệ thống cảnh báo hiểm hoạ thiên tai sử dụng hệ thống mạng cảm nhận không dây WSN.

Đề tài gồm ba chương:

Chương 1: Trình bày về quy trình tái kỹ nghệ hệ thống phần mềm.

Chương 2: Trình bày các công cụ trợ giúp quá trình tái kỹ nghệ phần mềm

Chương 3: Tái kĩ nghệ trong hệ thống cảnh báo thiên tai

CHƯƠNG 1: TÁI KỸ NGHỆ PHẦN MỀM

1.1. Tổng quan về tái kỹ nghệ

Sau một thời gian sử dụng, các phần mềm cần phải được bảo trì để đáp ứng các yêu cầu phát sinh của người sử dụng, của công nghệ mới, và sự thay đổi của các hoạt động nghiệp vụ theo thời gian... Đúng theo nghĩa vòng đời của một hệ thống phần mềm, ta lại bắt đầu các công việc: phân tích, thiết kế, cài đặt, kiểm thử... ở mức cao hơn. Có nhiều cách thực hiện việc bảo trì hệ thống phần mềm và cũng có nhiều công cụ để thiết kế lại phần mềm. Mỗi công cụ thiết kế lại phần mềm đều có những ưu và nhược điểm riêng, tuỳ theo hoàn cảnh thực tế mà ta có thể lựa chọn một công cụ sao cho hiệu quả nhất. Trong bài này em sẽ trình bày một số vấn đề về tái thiết kế hệ thống phần mềm bằng UML (Unified Modeling Language).

1.1.1. Bảo trì

Bảo trì hệ thống nhằm bảo đảm cho hệ thống được tiếp tục hoạt động sau khi thực hiện trắc nghiệm hay sau khi đưa hệ thống vào hoạt động trong thực tế. Bảo trì phần mềm bao gồm những sửa đổi làm hệ thống thích nghi với những yêu cầu thay đổi của người sử dụng, thay đổi dữ liệu cho phù hợp, gỡ rối, khử bỏ và sửa chữa các sai sót mà trước đây chưa phát hiện ra...

Các hoạt động trong bảo trì phần mềm bao gồm[5] :

- Trong quá trình kiểm thử, theo dõi quá trình hoạt động của hệ thống phần mềm, ta sẽ phát hiện ra tất cả các lỗi, các sai sót tiềm tàng trong hệ thống, tất cả các lỗi đó sẽ được gói tin cho các chuyên gia phát triển phần mềm để họ cập nhật lại. Tiến trình đó được gọi là *bảo trì sửa chữa*.
- Theo thời gian, các khía cạnh xử lý và hệ thống phần cứng thay đổi; môi trường làm việc như hệ điều hành thay đổi; các thiết bị ngoại vi và các phần tử của hệ thống được nâng cấp; các yêu cầu của khách hàng cho hệ thống sẽ thay đổi... Điều đó dẫn tới việc phải thay đổi hệ thống phần mềm sao cho phù hợp với các yêu cầu thay đổi trên, quá trình đó được gọi là *bảo trì thích nghi*.
- Khi hệ thống phần mềm thành công và được đưa vào sử dụng, người ta nhận được các khuyến cáo về khả năng mới, các chức năng cần bổ sung nâng cao…

Đó là quá trình nâng cấp hệ thống phần mềm cho phù hợp và tiện dụng hơn, được gọi là *bảo trì hoàn thiện*.

- Hệ thống cần phải thay đổi để đảm bảo tính tin cậy, an toàn trong tương lai, tạo cơ sở tốt hơn cho việc nâng cao chất lượng trong tương lai, tiến trình đó được gọi là *bảo trì phòng ngừa*, hoạt động này được đặc trưng bởi các kĩ thuật đảo ngược và tái kĩ nghệ.

Các công cụ bảo trì phần mềm có thể được chia theo các chức năng sau:

- Kĩ nghệ ngược với các công cụ đặc tả: nhận chương trình gốc làm đầu vào và sinh ra các mô hình phân tích và thiết kế có cấu trúc đồ thị, các thông tin thiết kế khác.
- Công cụ tái cấu trúc và phân tích mã : phân tích cú pháp chương trình, sinh ra đồ thị luồng điều khiển, và sinh tự động một chương trình có cấu trúc.
- Công cụ tái kĩ nghệ hệ thống trực tuyến: dùng để thay đổi các hệ thống cơ sở dữ liệu trực tuyến.

Bảo trì là giai đoạn cuối cùng trong tiến trình kĩ nghệ phần mềm, nó tiêu tốn rất nhiều thời gian, công sức và kinh phí. Tái kỹ nghệ là công nghệ đặc trưng giúp cho việc bảo trì các hệ thống phần mềm hiệu quả và nhanh chóng.

1.1.2. Tái kỹ nghệ

Các sản phẩm công nghệ đang được sử dụng nhiều, nhưng hơi cổ điển, chúng thường hay bị hỏng, mất nhiều thời gian cho việc sửa chữa và không đạt được trình độ của những công nghệ mới. Vậy ta phải làm gì? Nếu sản phẩm là phần cứng thì nó được thay bằng thiết bị mới, còn phần mềm thì các lựa chọn không có sẵn và lúc đó cần thiết phải xây dựng lại. Ta sẽ phải tạo ra một sản phẩm mà các chức năng khác có thể được thêm vào, hiệu năng tốt hơn, độ tin cậy cao hơn và khả năng bảo trì được cải thiện. Đó được gọi là tái kỹ nghệ.

Quá trình tái kỹ nghệ bao gồm phân tích, cấu trúc lại tài liệu, kỹ nghệ ngược, cấu trúc lại mã, cấu trúc lại dữ liệu, kỹ nghệ xuôi. Mục đích của các hoạt động này là tạo ra các phiên bản mới của các chương trình đang tồn tại, để nó có chất lượng cao hơn và bảo trì tốt hơn.

Sản phẩm của việc tái kỹ nghệ rất đa dạng như: các mẫu phân tích, các mẫu thiết kế, các thủ tục kiểm thử,..Đầu ra cuối cùng là việc tái kỹ nghệ các tiến trình nghiệp vụ và/hoặc tái kỹ nghệ phần mềm.

Trong thực tế, không ít các hệ thống phần mềm thương mại là các hệ thống cũ, nó cần để hỗ trợ cho các tiến trình nghiệp vụ. Các công ty cần các hệ thống này đến mức họ phải giữ chúng trong hoạt động. Chiến lược phát triển phần mềm bao gồm việc giữ lại, thay thế và phát triển kiến trúc chính là quá trình tái kỹ nghệ phần mềm.

Tái kỹ nghệ phần mềm đề cập đến việc làm lại hệ thống đang hoạt động để chúng có thể tiếp tục hoạt động tốt và dễ bảo trì sau này. Tái kỹ nghệ có thể bao gồm việc làm lại tài liệu hệ thống, tổ chức và cấu trúc lại hệ thống, biên dịch hệ thống sang ngôn ngữ lập trình hiện đại hơn, chỉnh sửa, cập nhật cấu trúc và lượng giá dữ liệu hệ thống. Thông thường, các chức năng chính của phần mềm không thay đổi và hệ thống cấu trúc của nó cũng được giữ lại.

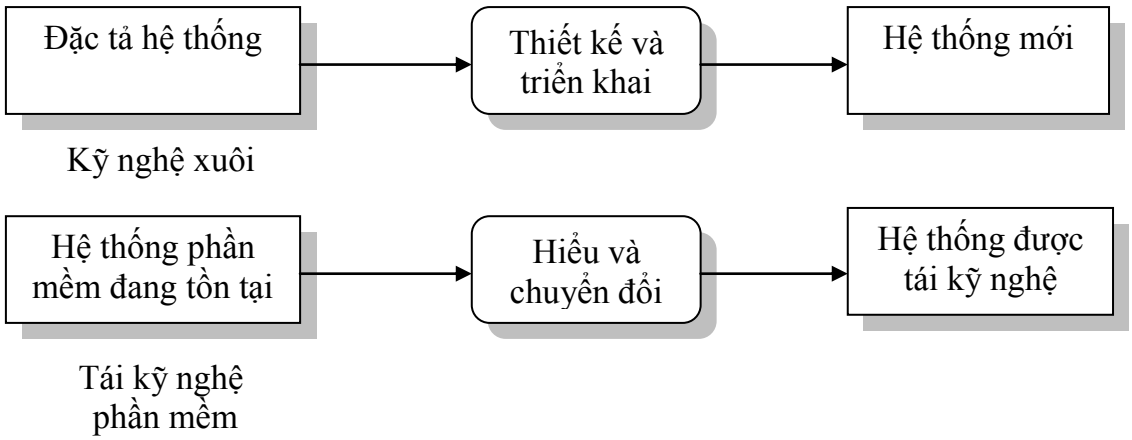
Từ khía cạnh kỹ thuật, tái kỹ nghệ phần mềm có thể xem như một giải pháp thứ hai cho những vấn đề của tiến hóa hệ thống. Kiến trúc phần mềm không được cập nhật như đối với các hệ thống trung tâm được phân tán. Nó cũng không thể thay đổi hoàn toàn ngôn ngữ lập trình hệ thống, vì hệ thống cũ không thể được chuyển đổi sang ngôn ngữ lập trình hướng đối tượng như Java hoặc C++ vốn có giới hạn trong hệ thống được giữ lại bởi chức năng phần mềm không thay đổi.

Tuy nhiên, từ quan điểm nghiệp vụ, tái kỹ nghệ phần mềm có thể chỉ nhằm để bảo đảm rằng hệ thống cũ có thể tiếp tục sử dụng. Nó có thể cũng đắt và gặp nhiều rủi ro như bất kỳ cách tiếp cận khác cho việc tiến hóa hệ thống. Để hiểu điều này, chúng ta cần đưa ra một đánh giá thô về vấn đề của các hệ thống cũ.

Tái kỹ nghệ một hệ thống phần mềm có ưu điểm hơn cách tiếp cận phát triển mới hệ thống; vì:

1. *Giảm rủi ro:* có sự rủi ro lớn trong việc phát triển mới một phần mềm, đó là tất yếu với một tổ chức. Các lỗi có thể được tạo ra trong đặc tả hệ thống, có thể nảy sinh các vấn đề, không ổn định hệ thống, v.v...
2. *Giảm giá thành:* Giá thành của việc tái kỹ nghệ là thấp hơn đáng kể so với giá phát triển phần mềm mới. Ulrich (1990) đưa ra một ví dụ của hệ thống cũ, ở đó

giá xây dựng mới được ước lượng khoảng 50 triệu đôla. Hệ thống được tái kỹ nghệ thành công với giá khoảng 12 triệu đô la. Nếu con số này là điển hình thì tái kỹ nghệ rẻ hơn viết lại.



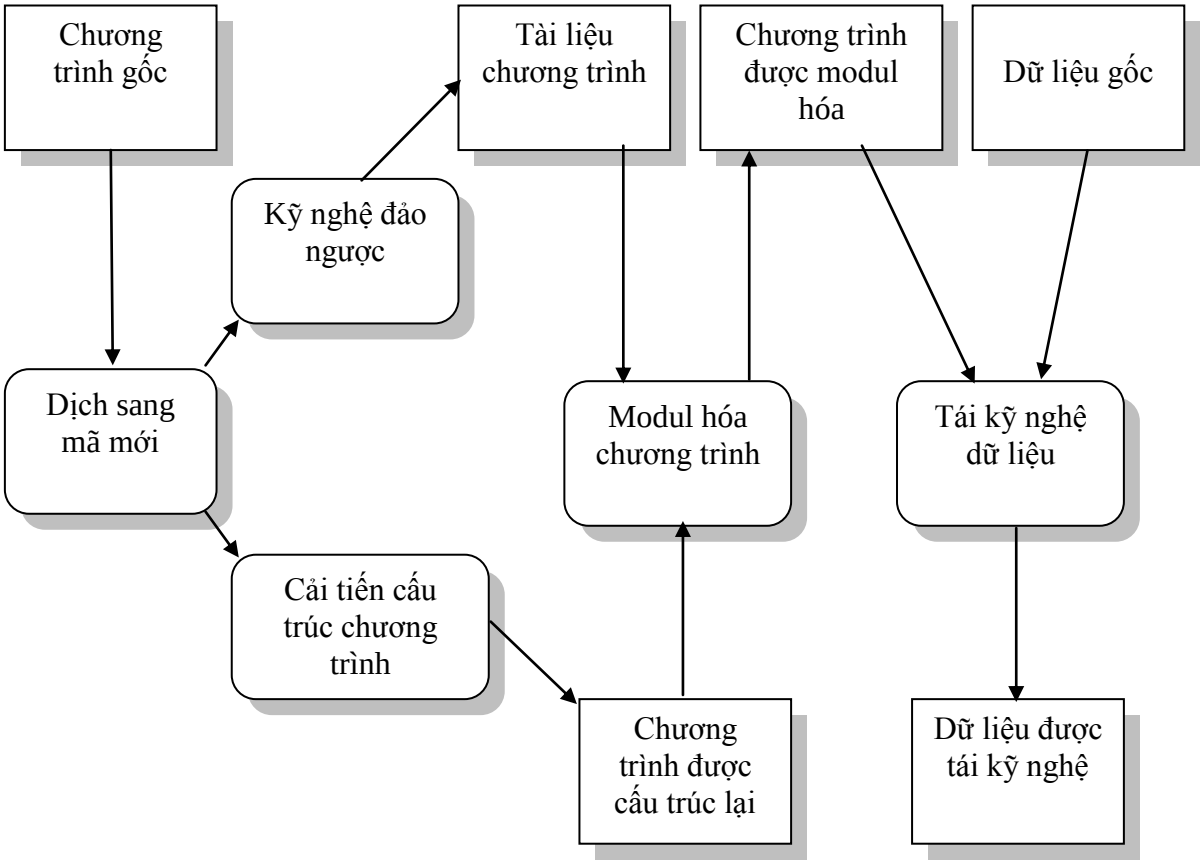
Hình 1-1 Tiến trình kỹ nghệ phần mềm xuôi và tái kỹ nghệ phần mềm

Hình 1.1 minh hoạ tiến trình tái kỹ nghệ khả thi. Đầu vào của tiến trình là một chương trình kế thừa và đầu ra là một cấu trúc, phiên bản hiệu chỉnh của chương trình. Ở cùng thời điểm như chương trình tái kỹ nghệ, dữ liệu cho hệ thống cũng có thể được tái kỹ nghệ. Các hoạt động trong tiến trình tái kỹ nghệ này là:

1. *Chuyển đổi mã nguồn*: Chương trình được chuyển đổi từ một ngôn ngữ lập trình phiên bản cũ sang một phiên bản mới hơn hoặc sang một ngôn ngữ khác.
2. *Kỹ nghệ ngược*: Chương trình được phân tích và trích ra các thông tin từ nó để giúp làm tài liệu về tổ chức và các chức năng của nó.
3. *Cải tiến cấu trúc chương trình*: Cấu trúc điều khiển của chương trình được phân tích và chỉnh sửa làm cho nó dễ đọc và dễ hiểu hơn.
4. *Modul hóa chương trình*: Việc thay thế các phần của chương trình được nhóm với nhau và được làm cho phù hợp, bổ sung modul mới và bỏ đi những dư thừa. Trong một số trường hợp, giai đoạn này có thể bao gồm cả sự biến đổi kiến trúc.
5. *Tái kỹ nghệ dữ liệu:* Dữ liệu xử lý bởi chương trình được thay đổi tương ứng với sự thay đổi chương trình.

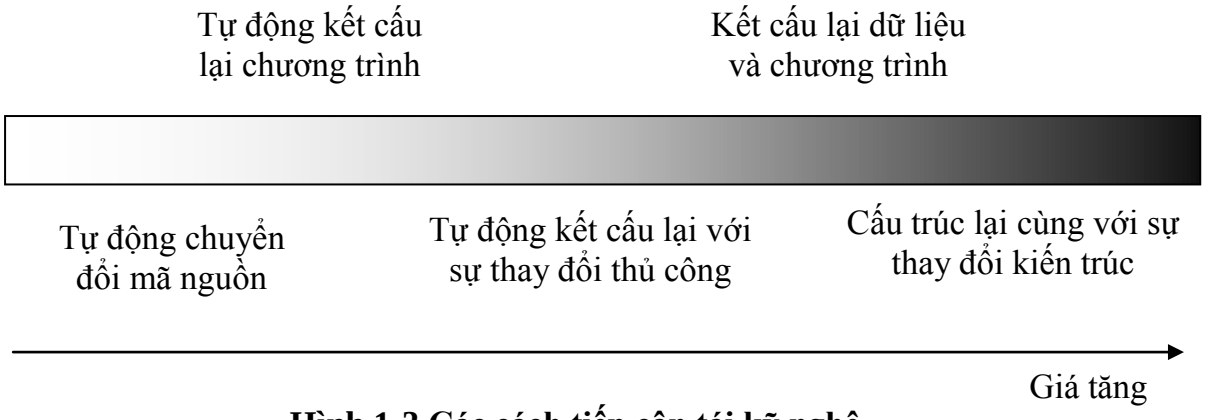
Chương trình tái kỹ nghệ có thể không cần thiết yêu cầu tất cả các bước trong hình 1.2. Việc chuyển mã nguồn có thể không cần thiết nếu ngôn ngữ lập trình dùng để phát triển hệ thống còn được hỗ trợ bởi công ty cung cấp trình biên dịch. Nếu tái kỹ

nghệ hoàn toàn dựa vào các công cụ tự động thì tài liệu lấy ra thông qua tái kỹ nghệ có thể là không cần thiết. Tái kỹ nghệ dữ liệu chỉ được yêu cầu nếu cấu trúc dữ liệu trong chương trình thay đổi khi tái kỹ nghệ hệ thống đòi hỏi. Tuy nhiên, tái kỹ nghệ phần mềm luôn bao gồm một số chương trình được cấu trúc lại.



Hình 1-2 Tiến trình tái kỹ nghệ phần mềm

Giá của việc tái kỹ nghệ rõ ràng phụ thuộc vào mức độ khó khăn của công việc thực hiện. Có nhiều cách tiếp cận khả thi với tái kỹ nghệ như chỉ ra trong hình 1.3. Giá tái kỹ nghệ tăng từ trái sang phải: từ mức chỉ phải chuyển đổi mã nguồn (là rẻ nhất) đến mức cao nhất là phải thay đổi lại toàn bộ cấu trúc.



Hình 1-3 Các cách tiếp cận tái kỹ nghệ

Ngoài các yếu tố liên quan đến việc mở rộng hoạt động tái kỹ nghệ, còn có những yếu tố khác về nguyên tắc có ảnh hưởng tới giá của việc tái kỹ nghệ như sau:

1. *Chất lượng phần mềm được tái kỹ nghệ*: Giá trị chất lượng của phần mềm và tài liệu của nó cao hơn giá của tái kỹ nghệ.
2. *Giá trị công cụ hỗ trợ cho việc tái kỹ nghệ*: Giá trị hiệu quả cho việc tái kỹ nghệ một hệ thống phần mềm có thể thấp hơn giá trị công cụ CASE để tự động thay đổi hầu hết các chương trình.
3. *Phạm vi của yêu cầu chuyển đổi dữ liệu*: Nếu tái kỹ nghệ yêu cầu chuyển đổi một số lớn dữ liệu, điều đó làm tăng đáng kể giá thành thực hiện.
4. *Giá trị của đội ngũ chuyên môn*: Nếu trách nhiệm nhân viên bảo trì hệ thống không được sử dụng trong tiến trình tái kỹ nghệ, việc này sẽ làm tăng giá thành. Đội ngũ chuyên môn tái kỹ nghệ sẽ cần nhiều thời gian để hiểu hệ thống.

Nhược điểm chính của tái kỹ nghệ phần mềm là các giới hạn thực tế đối với việc mở rộng hệ thống có thể nằm trong phạm vi của việc tái kỹ nghệ. Điều này là có thể. Ví dụ, chuyển đổi một hệ thống văn bản sử dụng tiếp cận chức năng tới hệ thống hướng đối tượng. Kiến trúc chính thay đổi hoặc tổ chức lại căn bản việc quản lý dữ liệu hệ thống không thể được thực hiện tự động, như thế liên quan cao đến việc thêm giá thành. Mặc dù tái kỹ nghệ có thể cải tiến việc bảo trì, tái kỹ nghệ hệ thống sẽ không thể duy trì được như hệ thống mới phát triển sử dụng các phương pháp tái kỹ nghệ phần mềm hiện đại.

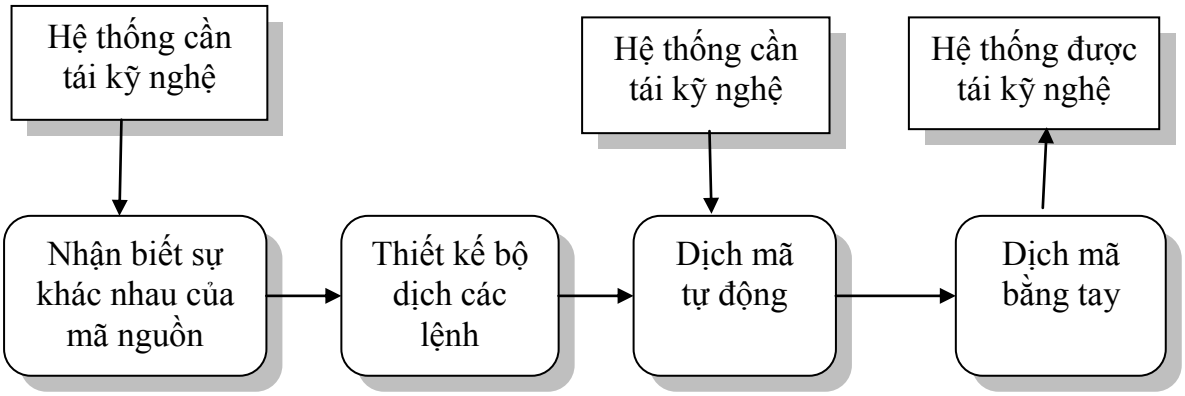
1.2. Dịch mã nguồn

Các lý do cần dịch mã nguồn:

1. *Cập nhật nền phần cứng mới*: Tổ chức muốn thay đổi nền phần cứng chuẩn. Các bộ dịch ngôn ngữ gốc không có giá trị trên phần cứng mới.
2. *Thiếu đội ngũ có kỹ năng*: Có thể thiếu đội ngũ bảo trì lành nghề cho ngôn ngữ gốc. Đây là một vấn đề thực tế ở đó các chương trình được viết bằng một ngôn ngữ không chuẩn mà hiện tại không sử dụng.

3. *Các thay đổi chính sách của tổ chức*: tổ chức có thể quyết định chuẩn hóa trên ngôn ngữ thực tế để giảm thiểu chi phí trợ giúp phần mềm trợ giúp của nó. Bảo trì một số phiên bản của bộ dịch cũ có thể rất đắt.

4. *Sự yếu kém của việc trợ giúp phần mềm*: Các nhà cung cấp chương trình dịch có thể đã bỏ việc kinh doanh hoặc không tiếp tục hỗ trợ sản phẩm của họ nữa.



Hình 1-4 Tiến trình dịch chương trình

Hình 1.4 chỉ ra tiến trình dịch mã nguồn. Có thể không cần hiểu chi tiết hoạt động của phần mềm hoặc chỉnh sửa kiến trúc hệ thống. Phân tích sự liên quan có thể tập trung vào ngôn ngữ lập trình tương đương như cấu trúc điều khiển chương trình.

Việc dịch mã nguồn chỉ thực sự là kinh tế nếu bộ dịch tự động sẵn sàng để thực hiện với một số lượng lớn. Đây có thể là một chương trình được viết đặc biệt, một công cụ được mua để chuyển đổi từ một ngôn ngữ này sang một ngôn ngữ khác hoặc một hệ thống mẫu thích hợp. Trong trường hợp thứ hai, cách thức một tập các lệnh để tạo sự dịch chuyển từ sự trình bày này sang một sự trình bày khác cần được viết các mẫu được tham số hóa trong ngôn ngữ nguồn được xác định và kết hợp với các mẫu tương đương trong ngôn ngữ đích.

Trong nhiều trường hợp, việc dịch hoàn toàn tự động là không thể. Các cấu trúc trong ngôn ngữ nguồn không có cấu trúc tương đương trực tiếp trong ngôn ngữ đích. Có thể các lệnh điều kiện biên dịch trong mã nguồn mà không được hỗ trợ trong ngôn ngữ đích. Trong trường hợp này, bạn cần tạo ra sự thay đổi thủ công làm cho phù hợp và cải tiến hệ thống sinh lệnh.

1.3. Kỹ nghệ ngược

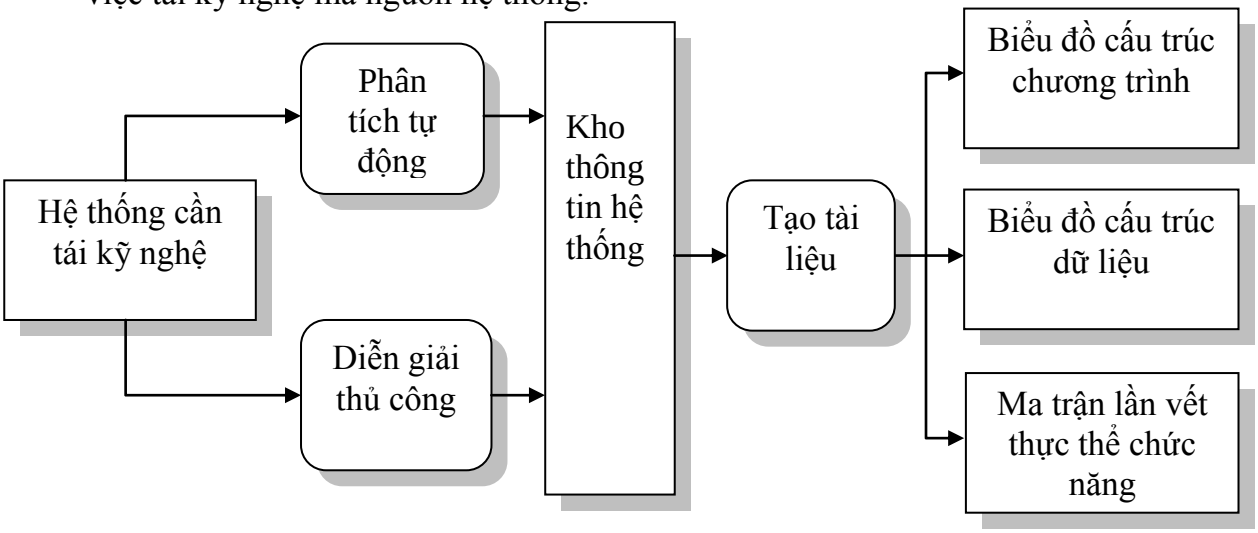
Kỹ nghệ ngược (reserve engineering) là tiến trình phân tích phần mềm mã nguồn với mục đích chuyển đổi nó về dạng thiết kế và đặc tả. Chương trình tự nó không bị thay đổi bởi tiến trình kỹ nghệ ngược. Mã nguồn phần mềm thường có giá trị như đầu vào cho tiến trình Kỹ nghệ ngược. Tuy nhiên, điều này có thể không có và Kỹ nghệ ngược cần bắt đầu với mã thực thi.

Kỹ nghệ ngược không hoàn toàn giống như tái kỹ nghệ (re-engineering). Mục tiêu của kỹ nghệ ngược thu được là thiết kế hoặc đặc tả của hệ thống từ mã nguồn của nó. Mục tiêu của tái kỹ nghệ là làm ra một hệ thống phần mềm mới, dễ bảo trì hơn.

Tất nhiên, như chúng ta có thể thấy trong hình 1.2, kỹ nghệ ngược để có được một sự hiểu biết về hệ thống và thường là một phần của tiến trình tái kỹ nghệ.

Kỹ nghệ ngược được dùng trong tiến trình tái kỹ nghệ phần mềm là để phục hồi lại thiết kế chương trình mà sẽ giúp các kỹ sư hiểu tốt một chương trình trước khi tổ chức lại cấu trúc của nó. Tuy nhiên, không nhất thiết theo sau kỹ nghệ ngược phải là tái kỹ nghệ:

1. Bản thiết kế và đặc tả của một hệ thống đang tồn tại có thể được kỹ nghệ ngược, vì chúng có thể dùng như một đầu vào cho đặc tả các yêu cầu của chương trình cần thay thế.
2. Như một sự lựa chọn, bản thiết kế và đặc tả có thể được kỹ nghệ ngược, vì chúng sẵn sàng để giúp bảo trì chương trình. Thông tin này có thể không cần thiết cho việc tái kỹ nghệ mã nguồn hệ thống.



Hình 1-5 Tiến trình kỹ nghệ ngược

Tiến trình kỹ nghệ ngược được minh họa trong hình 1.5. Tiến trình bắt đầu với một pha phân tích. Trong pha này, hệ thống được phân tích dùng công cụ tự động để nhận ra cấu trúc của nó. Trong bản thân nó, không đủ để tái tạo hệ thống thiết kế. Sau đó các kỹ sư làm việc với mã nguồn hệ thống và mô hình cấu trúc của nó. Họ thêm thông tin vào pha này khi họ hiểu hệ thống. Thông tin này được lưu trữ như một đồ thị có hướng gắn kết tới mã nguồn chương trình.

1.4. Phát triển trúc chương cấu trình

Cần sử dụng bộ nhớ tối ưu và sự thiếu hiểu biết về kỹ nghệ phần mềm bởi một số người lập trình, nghĩa là một số hệ thống cũ không được cấu trúc tốt. Sau khi cấu trúc điều khiển bị lộn xộn với một số nhánh vô điều kiện và logic điều khiển không rõ ràng. Cấu trúc này cũng có thể bị giảm giá trị bởi sự bảo dưỡng thường xuyên. Sự thay đổi chương trình có thể được tạo ra.

Hình 1.6 minh họa sự phức tạp trong điều khiển, ta có thể tạo một chương trình khác đơn giản hơn để thực hiện. Chương trình được viết tựa như FORTRAN. Trong hình 1.6 là bộ điều khiển một hệ thống lò sưởi. Một bộ công tắc có thể đặt ở một trong 3 trạng thái: on, off, controlled. Nếu hệ thống được điều khiển khi nó chuyển On và tắt phụ thuộc vào thời gian đặt và bộ điều nhiệt. Nếu lò sưởi là ON thì chuyển công tắc lò sưởi thành OFF và ngược lại.

Đặc biệt, các chương trình phát triển này có độ phức tạp logic cấu trúc vì chúng được chỉnh sửa trong khi bảo trì. Thêm các điều kiện kết hợp các hoạt động mà không thay đổi cấu trúc điều khiển đang tồn tại. Trong một thời hạn ngắn, đây là một giải pháp nhanh và ít mạo hiểm vì nó giảm sự thay đổi lỗi trong hệ thống. Tuy nhiên, trong một thời gian dài nó dẫn đến khó hiểu mã chương trình. Cấu trúc mã phức tạp có thể cũng xuất hiện do khi những người lập trình cố gắng tránh sự lặp lại mã. Điều này, đôi lúc cần thiết khi chương trình bị ràng buộc vì bộ nhớ có giới hạn.

```
Start: Get(Time-on, Time-off, Time, Setting, Temp, Switch)
 If Switch=off goto off
 If Switch= on goto on
 Goto Cntrld
Off: if Heating-status = on goto Sw-off
 Goto loop
On: if Heating-status= off goto Sw-on
 Goto loop
Cntrld:
 if Time = Time-on goto on
 if Time = Time-off goto off
 if Time < Time-on goto Start
 if Time > Time-off goto Start
 if Temp > Setting then goto off
 if Temp < Setting then goto on
Sw-off: Heating-status :=off
 Goto Switch
Sw-on: Heating-status:=on
Switch: Switch-heating
Loop: goto Start
```

Hình 1-6 Chương trình điều khiển với ống dẫn điện logic

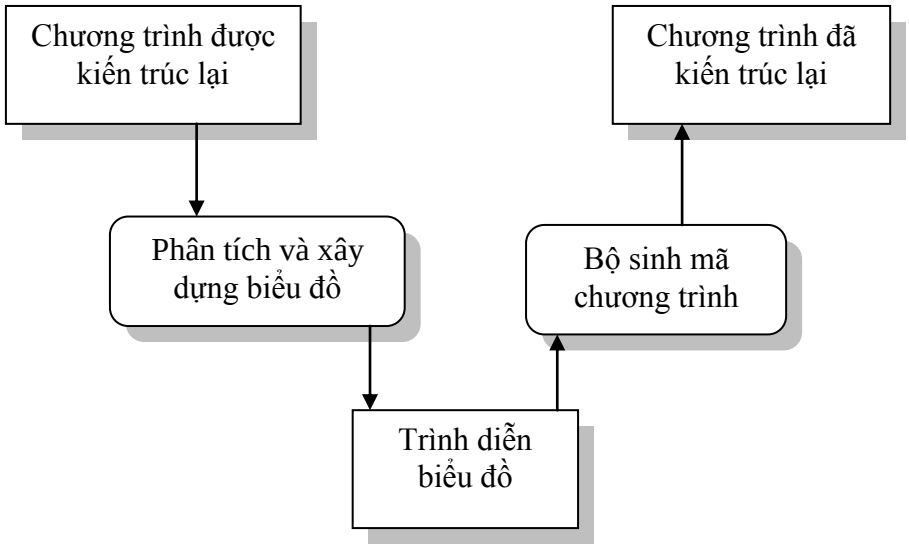
```
loop
 {Trạng thái Get tìm các giá trị để đưa các giá trị từ môi
trường hệ thống}
   Get (Time-on, Time-off, Time, Setting, Temp, Switch);
case Switch of
when On=>
  if Heating-status = off then
  Switch-heating; Heating-status :=on;
  end if;
 when Off=>
  if Heating-status = on then
  Switch-heating; Heating-status :=off;
  end if;
 when Controlled =>
 if Time >= Time-on and Time <= Time-off then
 if Temp > Setting and Heating-status = on then
 Switch-heating; Heating-status:=off;
    elseif Temp < Setting and Heating-status:=off then
    Switch-heating; Heating-status:=on;
    end if;
    end if;
   end case;
end loop;
```

Hình 1-7 Chương trình điều khiển cấu trúc

Hình 1.7 chỉ ra hệ thống điều khiển tuần tự được viết lại khi sử dụng cấu trúc điều khiển. Chương trình có thể đọc liên tục từ trên xuống dưới, như vậy nó khá dễ hiểu. Ba vị trí chuyển đổi on, off, và controlled được định nghĩa rõ ràng và liên kết tới mã của vật liên kết của nó. Ta không sử dụng Java khi chương trình nguồn không là chương trình hướng đối tượng.

```
Complex condition
If not (A>B) and (C<D or not (E>F))…
Simplified condition
If (A<=B) and (C>=D or E>F)…
```

Hình 1-8 Đơn giản hóa điều kiện



Hình 1-9 Kiến trúc lại chương trình tự động

Vấn đề cấu trúc lại chương trình tự động bao gồm:

1. *Mất các lời chú thích:* Nếu chương trình có chú thích trong một dòng thì dòng chú thích này sẽ mất giá trị khi một phần của tiến trình kiến trúc lại.
2. *Mất tài liệu:* Tương tự, sự tương ứng giữa tài liệu chương trình bên ngoài và chương trình cũng mất. Tuy nhiên trong một vài trường hợp, cả phần chú thích và tài liệu của chương trình đều quá hạn, như vậy đây không phải là một nhân tố quan trọng.
3. *Yêu cầu tính toán lớn:* giải thuật nhúng trong công cụ cấu trúc lại phức tạp. Mặc dù phần cứng hiện đại, nhanh, nó cũng có thể mất một thời gian dài để hoàn thành tiến trình cấu trúc lại cho chương trình lớn.

1.5. Môdul hóa chương trình

Mô đul hóa chương trình là tiến trình tổ chức lại chương trình sao cho những phần chương trình được tập hợp với nhau và được xem như là một modul. Khi chương trình đã được modul hóa, dễ bỏ đi những phần thừa trong các phần được thay thế, để tối ưu chương trình, để các phần của chương trình tương tác với nhau trong một giao diện đủ đơn giản.

Ví dụ, trong một chương trình xử lý dữ liệu địa trấn, tất cả các hoạt động kết hợp với sự trình diễn đồ họa của dữ liệu có thể được tập hợp với nhau trong một modul đơn giản. Nếu hệ thống bị phân tán, các modul được tạo có thể được gói gọn như các đối tượng và truy cập thông qua giao diện chung. Vài kiểu modul khác nhau có thể được tạo trong khi xử lý modul hóa chương trình. Quá trình này gồm:

1. *Trừu tượng dữ liệu:* Kiểu dữ liệu trừu tượng được tạo bởi sự kết hợp dữ liệu với các thành phần tiến trình.
2. *Modul hóa phần cứng*: Thay dữ liệu trừu tượng và tập hợp tất cả các hàm chúng với nhau một cách chặt chẽ, nó được sử dụng để điều khiển thiết bị phần cứng riêng biệt.
3. *Modul hóa chức năng*: Đó là các modul mà nó tập hợp các chức năng với nhau, các chức năng này đồng dạng hoặc có các tác vụ gần nhau. Ví dụ, tất cả các chức năng có liên quan với đầu vào và giá trị đầu vào có thể được hợp nhất trong một modul đơn giản. Kiểu này của sự modul hóa được xét đến khi không cần sửa lại trừu tượng dữ liệu chương trình.
4. *Modul trợ giúp tiến trình*: Đó là các modul mà ở đó tất cả các chức năng và các mục dữ liệu đặc biệt yêu cầu để trợ giúp tiến trình nghiệp vụ đặc biệt được nhóm lại. Ví dụ, trong một hệ thống thư viện, một modul trợ giúp tiến trình có thể gồm tất cả các chức năng yêu cầu để trợ giúp sự phát hành và phản hồi của sách.

Modul hóa chương trình thường thực hiện thủ công bởi sự kiểm tra và sửa chữa mã nguồn. Để modul hóa một chương trình, bạn cần nhận ra quan hệ, giữa các thành phần và thực hiện những gì mà các thành phần này làm. Các công cụ trình diễn và làm trực quan trợ giúp nhưng nó không thể tự động hoàn thành tiến trình này.

1.6. Tái kỹ nghệ dữ liệu

Tái kỹ nghệ dữ liệu sẽ không cần thiết nếu chức năng của hệ thống không đổi. Tuy nhiên, trong thực tế, có một số lý do ta cần sửa dữ liệu trong các chương trình của hệ thống cũ:

1. *Sự suy thoái dữ liệu*: Như trên, chất lượng của dữ liệu hướng tới sự suy giảm dần. Sự thay đổi dữ liệu mở đầu cho các lỗi, các giá trị giống nhau có thể được tạo ra và sự thay đổi môi trường bên ngoài không thể được phản ánh vào trong dữ liệu. Đây là điều không thể tránh được bởi thời gian sống của dữ liệu thường là rất dài. Ví dụ, dữ liệu phản hồi của cá nhân vào trong dữ liệu đang tồn tại khi một tài khoản được mở và có thể cần tiếp tục tồn tại ít nhất là bằng thời gian sống của khách hàng. Khi lý lịch của khách hàng thay đổi, sự thay đổi này có thể không thích hợp với dữ liệu trong nhà băng. Tái kỹ nghệ chương trình có thể mang vấn đề chất lượng dữ liệu tới sự sáng tỏ và đó là điểm mấu chốt cần kết hợp tái kỹ nghệ dữ liệu.
2. *Các hạn chế vốn có được xây dựng trong chương trình*: Khi thiết kế ban đầu, những người phát triển của nhiều chương trình đã đưa vào những ràng buộc tự xây dựng về số lượng dữ liệu mà nó có thể xử lý. Tuy nhiên, ngày nay chương trình thường yêu cầu xử lý nhiều dữ liệu hơn dự tính ban đầu của các nhà phát triển. Tái kỹ nghệ dữ liệu có thể được yêu cầu để hủy bỏ các giới hạn đó. Ví dụ, Rochester và Douglass (1993) diễn tả hệ thống quản lý tiền, nó được thiết kế ban đầu để lưu giữ 99 loại vốn. Công ty chạy hệ thống đang quản lý hơn 2000 loại vốn, và cần chạy 23 bản rời nhau của hệ thống. Như vậy, họ quyết định tái kỹ nghệ hệ thống và kết hợp dữ liệu của nó.
3. *Phát triển kiến trúc*: Nếu một hệ thống tập trung chuyển sang một kiến trúc phân tán, điều cần thiết cốt lõi của kiến trúc đó là một hệ thống quản lý dữ liệu, nó có thể được truy cập từ các máy trạm. Điều này có thể yêu cầu một sự cố gắng lớn trong việc tái kỹ nghệ để di chuyển dữ liệu từ các tệp rời vào trong hệ thống quản lý dữ liệu chủ. Sự di chuyển tới một kiến trúc chương trình phân tán có thể được khởi tạo khi một tổ chức quyết định di chuyển từ việc quản lý dữ liệu bằng các tệp cơ sở sang hệ thống quản lý cơ sở dữ liệu.

Giống như tái kỹ nghệ chương trình, có một tập các cách tiếp cận tới tái kỹ nghệ dữ liệu. Nó phản ánh lý do tại sao việc tái kỹ nghệ dữ liệu được yêu cầu. Điều này được chỉ ra trong hình 1.10.

Tiếp cận	Diễn tả
Làm sạch dữ liệu	Các bản ghi dữ liệu và các giá trị được phân tích để cải tiến chất lượng của chúng. Sự trùng lặp được bỏ đi, sự dư thừa thông tin được xóa bỏ và định dạng thích hợp được áp dụng cho tất cả các bản ghi. Thường những điều này không yêu cầu thay đổi chương trình.
Mở rộng dữ liệu	Trong trường hợp này, dữ liệu và chương trình được tái kỹ nghệ để bỏ đi giới hạn trên việc xử lý dữ liệu. Việc này có thể yêu cầu thay đổi tới chương trình để tăng chiều dài trường, sửa tăng giới hạn trên bảng,…Sau đó, dữ liệu, tự nó có thể viết lại và làm sạch để phản ánh sự thay đổi của chương trình.
Di chuyển dữ liệu	Trong trường hợp này, dữ liệu được di chuyển vào sự điều khiển của hệ quản trị cơ sở dữ liệu hiện đại. Dữ liệu có thể được lưu trữ trong các tệp độc lập hoặc có thể được quản lý bởi một kiểu cũ hơn của DBMS. Điều này được minh họa trong hình 1.11.

Hình 1-10 Tiếp cận tới việc tái kỹ nghệ dữ liệu

1.7. Kết luận

Đối tượng của hệ thống tái kỹ nghệ là cải tiến cấu trúc hệ thống và làm cho nó dễ hiểu hơn. Như vậy, giá của bảo trì hệ thống trong tương lai giảm.

Tiến trình tái kỹ nghệ bao gồm: Dịch mã nguồn, kỹ nghệ ngược, phát triển cấu trúc chương trình, modul hóa chương trình và tái kỹ nghệ dữ liệu.

Trên đây là quy trình tái kỹ nghệ một hệ thống phần mềm nói chung. Khi áp dụng cho mỗi phần mềm cụ thể ta sẽ đưa ra quy trình phù hợp nhất với nó.

Chương sau sẽ nghiên cứu một hệ thống phần mềm cụ thể, đó là hệ thống cảnh báo thiên tai và việc áp dụng quy trình tái kỹ nghệ cho hệ thống đó.

CHƯƠNG 2: CÁC CÔNG CỤ TRỢ GIÚP TÁI KỸ NGHỆ

2.1. Giới thiệu công cụ Rational Software Architecture

Công cụ phát triển phần mềm IBM Rational, dựa trên cơ sở mã nguồn mở của Eclipse. Rational Software Architecture (RSA) là phần mềm công cụ hỗ trợ mạnh cho xây dựng kiến trúc phần mềm, cho phân tích, thiết kế hệ thống phần mềm và cho xây dựng các ứng dụng phần mềm theo hướng đối tượng. Nó giúp mô hình hoá hệ thống đồng thời sinh mã nguồn chương trình, đảm bảo tính đúng đắn, hợp lý của kiến trúc hệ thống từ khi khởi đầu dự án. Mô hình RSA là bức tranh hệ thống, nó bao gồm toàn bộ các biểu đồ của UML, tác nhân, ca sử dụng, đối tượng, lớp, thành phần và các nút triển khai trong hệ thống. Nó mô tả chi tiết hệ thống bao gồm những gì và chúng làm việc ra sao, để người phát triển hệ thống có thể sử dụng mô hình lập kế hoạch chi tiết cho việc xây dựng hệ thống. RSA hỗ trợ giải quyết nhiều vấn đề quan trọng trong quá trình xây dựng và phát triển hệ thống, chẳng hạn việc đội ngũ dự án giao tiếp với khách hàng hay làm các tài liệu yêu cầu, quy trình phát trển phần mềm RUP.

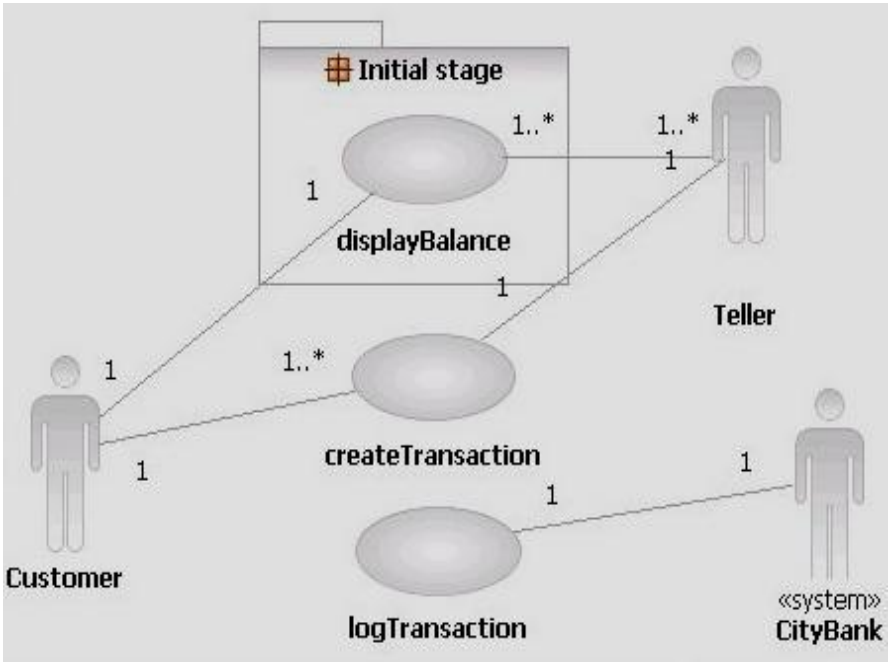
Nó cho phép phát sinh mã trình từ mô hình của UML sang một ngôn ngữ và dịch ngược từ một ngôn ngữ sang mô hình UML. Rose Eterprise cho phép phát sinh mã trình sang các ngôn ngữ Ada83, Ada95, ANSI C++, CORBA, Java, COM, Visual Basic, Visual C++, C/C++, Oracle, DB2, SQL Server, XML... và dịch ngược từ mã nguồn của các hệ trên sang mô hình của UML. Hơn nữa Rational Software Architecture cho phép mô hình hoá các ứng dụng trên website và tái thiết kế các ứng dụng trên nó.

Rational Software Architecture hỗ trợ tiến trình tái kỹ nghệ và tiến trình thiết kế tái kỹ nghệ cả với một số ngôn ngữ lập trình trên mạng như ASP, JSP, J2EE và các trang HTML. Nó gán các stereotype thích hợp cho các lớp và tạo các mối quan hệ giữa chúng. Rational Software Architecture còn cho phép tái kỹ nghệ dữ liệu với: IBM DB2, Microsoft SQL Sever, Oracle và Sysbase Adaptive Sever 12.x.

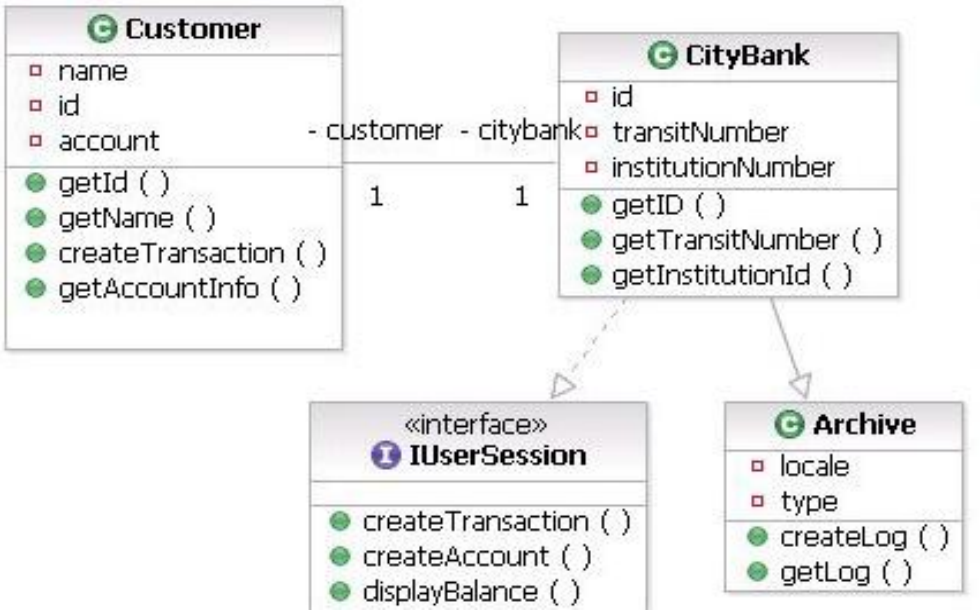
Đặc điểm chức năng của Rational Software Architecture là: mô hình hóa tiến trình nghiệp vụ hệ thống, phân tích và thiết kế hệ thống, kiến trúc hệ thống, lập trình và kiểm thử hệ thống.

Với công cụ phát trển phần mềm Rational Software Architecture Platform, chúng ta có thể sử dụng để mô hình hóa hệ thống phần mềm và tạo ra các loại biểu đồ trong UML như:

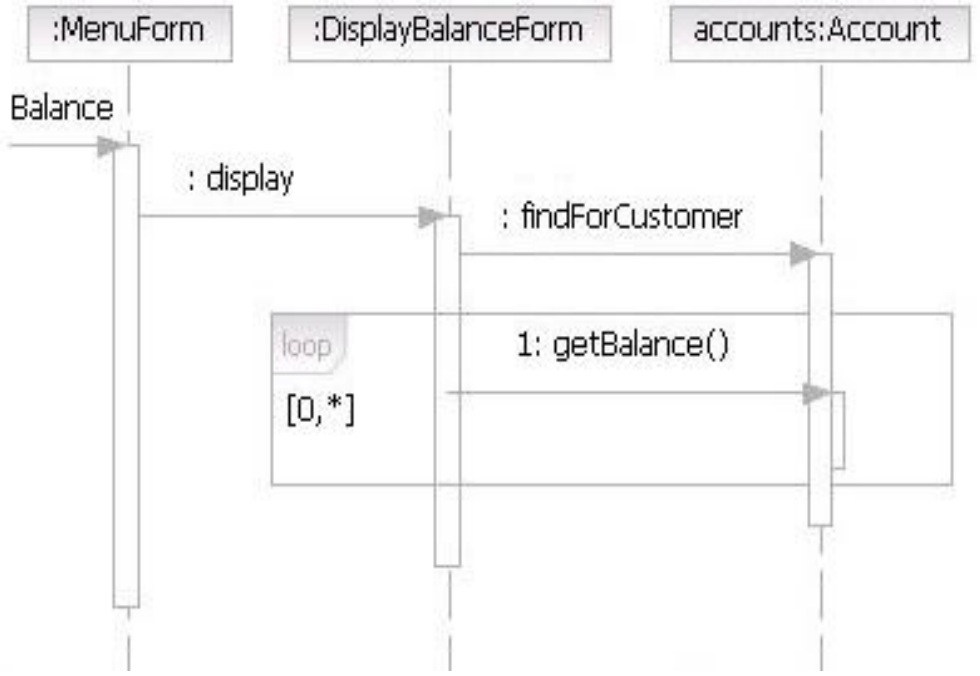
- Biểu đồ Ca sử dụng – Use Case
- Biểu đồ Lớp-Class
- Biểu đồ Tuần tự - Sequence
- Biểu đồ Truyền thông – Communication
- Biểu đồ Máy trạng thái – State Machine
- Biểu đồ Hoạt động - Activity
- Biểu đồ Thành phần – Component
- Biểu đồ Cấu trúc tổng hợp – Composite Structure
- Biểu đồ Triển khai - Deployment



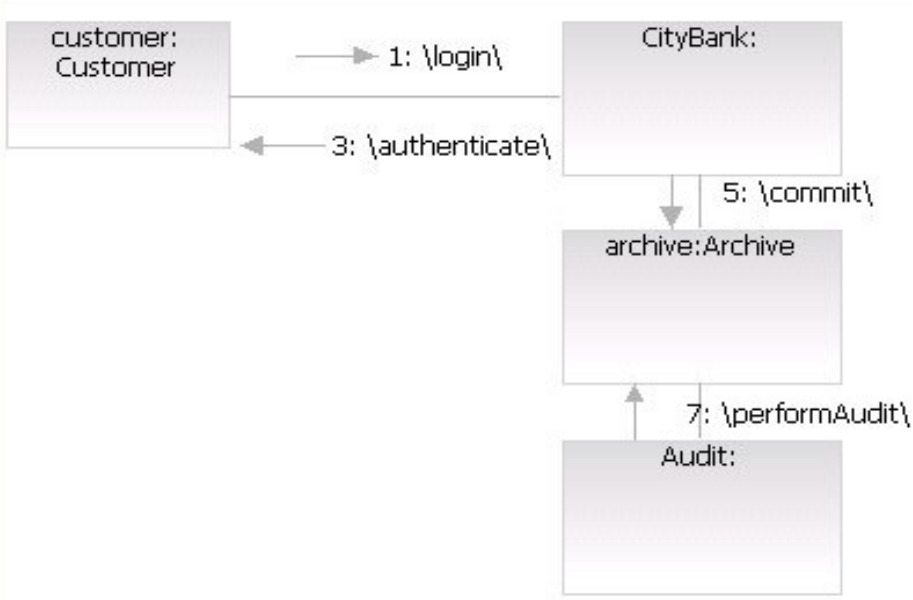
Hình 2-1Biểu đồ Ca sử dụng – Use Case



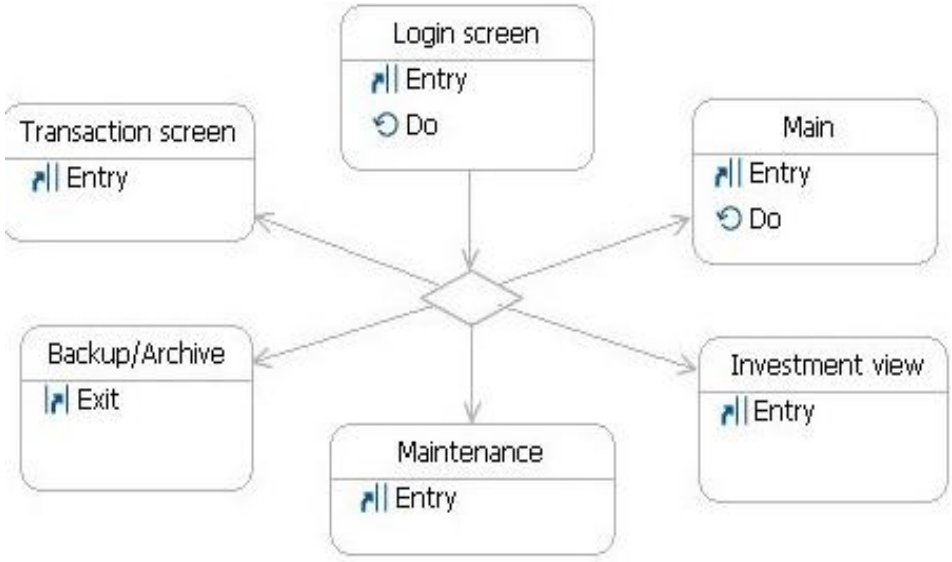
Hình 2-2 Biểu đồ Lớp-Class



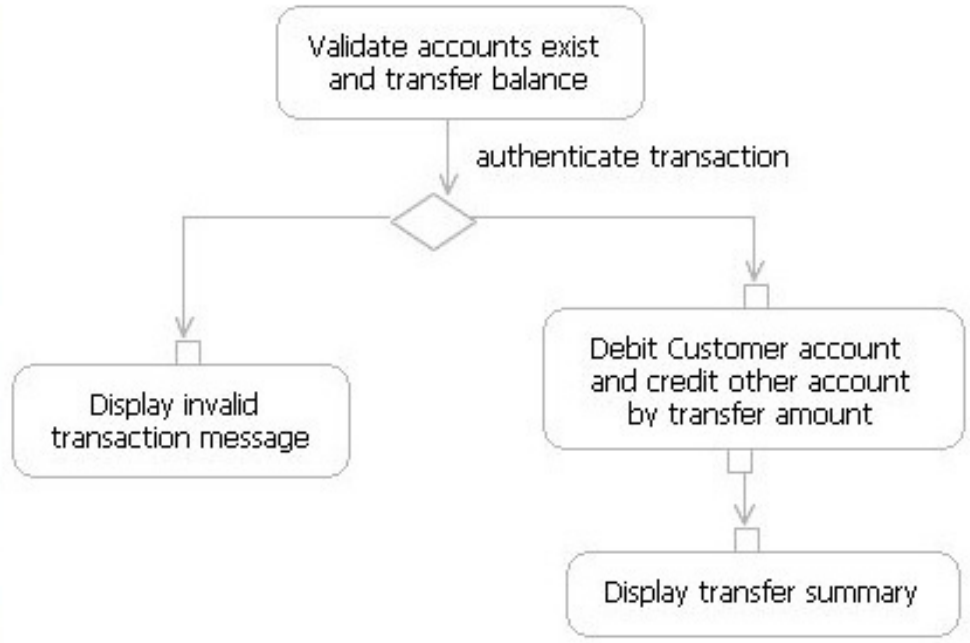
Hình 2 -3 **Biểu đồ tuần tự - Sequence**



Hình 2-4 Biểu đồ truyền thông – Communication

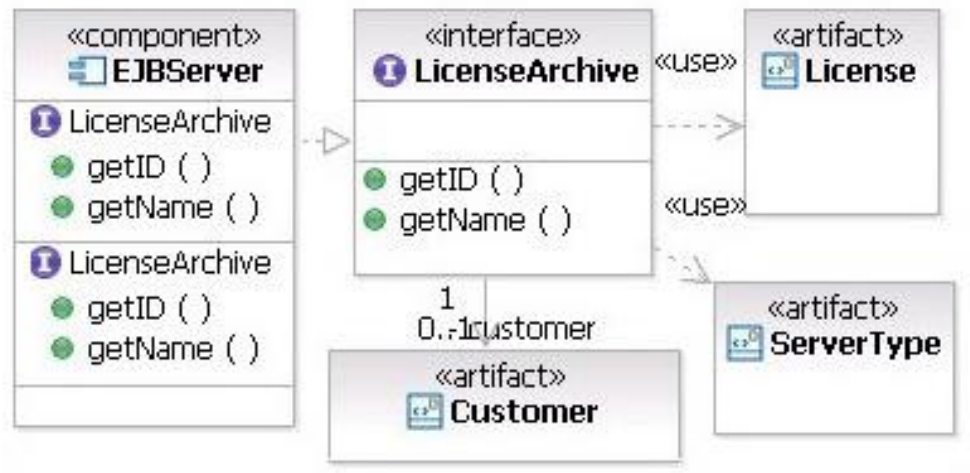


Hình 2-5 Biểu đồ máy trạng thái – State Machine

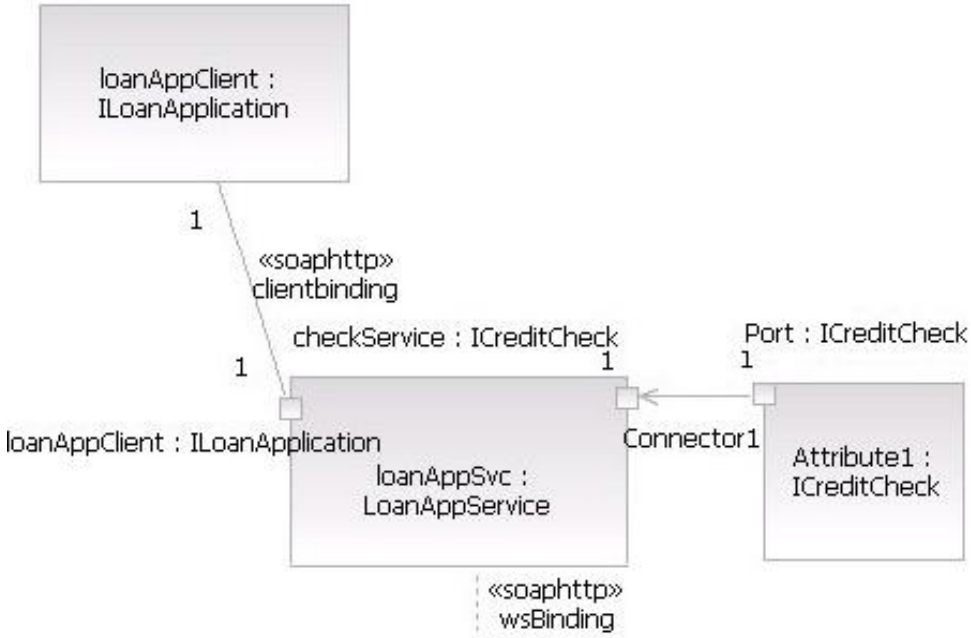


Hình 2-6 Biểu đồ hoạt động - Activity

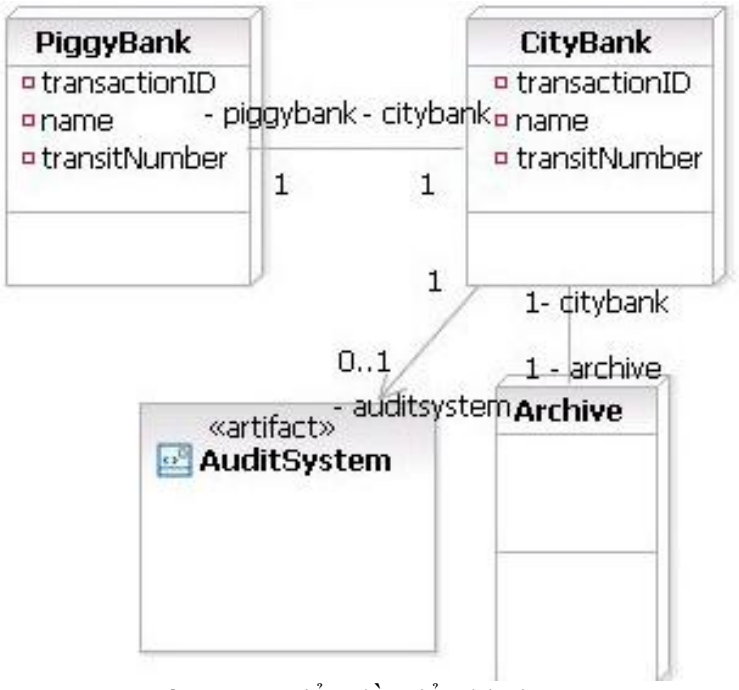
Biểu đồ hoạt động nắm bắt hành động và các kết quả của chúng. Biểu đồ hoạt động tập trung vào công việc được thực hiện trong khi thực thi mộy thủ tục (hàm), các hoạt động trong một lần thực thi một trường hợp sử dụng hoặc trong một đối tượng.



Hình 2-7 Biểu đồ thành phần – Component



Hình 2-8 Biểu đồ cấu trúc tổng hợp – Composite Structure



Hình 2-9 Biểu đồ triển khai - Deployment

2.2. Công cụ lập trình nhúng

Quản trị cấu hình và quản trị thay đổi

Trong các công cụ sử dụng thiết lập cho mạng cảm nhận phải kể đến phần mềm nhúng viết cho CC1010 được viết bằng ngôn ngữ C/C++, sử dụng các thư viện cho CC1010 do hãng Chipcon cung cấp và chương trình biên dịch Keil Vision 2.0. Nó

được dùng để xây dựng các chương trình cho các họ VĐK tương thích 8051 của Intel. Đây là bộ chương trình dịch cho phép người viết chương trình soạn thảo chương trình, dịch chương trình và gỡ lỗi trên cùng một môi trường. Chương trình dịch hỗ trợ cho cả ngôn ngữ C/C++ và Assembly. Hãng Chipcon cũng cung cấp bộ thư viện CC1010IDE hỗ trợ cho việc xây dựng phần mềm cho VĐK CC1010. Đây là bộ thư viện giúp cho việc xây dựng chương trình cho CC1010 được dễ dàng và nhanh chóng. Cuối cùng, bộ liên kết đưa ra dạng tệp thực thi dạng Intel HEX và có thể nạp vào bộ nhớ Flash của vi điều khiển.

Mô hình của một phần mềm nhúng viết cho CC1010 như sau:

<table>
<tr><td colspan="3">Chương trình ứng dụng</td></tr>
<tr><td rowspan="3">Thư viện C/C++ chuẩn</td><td colspan="2">Thư viện tiện ích Chipcon
(Chipcon utility library-CUL)</td></tr>
<tr><td colspan="2">Thư viện phần cứng
(Hardware abstraction library – HAL)</td></tr>
<tr><td colspan="2">Các tệp định nghĩa phần cứng
(Hardware definition Files - HDF)</td></tr>
</table>

Hình 2-10 Mô hình phần mềm nhúng CC1010

2.3. Dịch xuôi, dịch ngược trên Rational Software Architecture

Ngôn ngữ mô hình hoá thống nhất UML là ngôn ngữ chuẩn để viết ra các bản thiết kế phần mềm, nó có thể sử dụng để làm trực quan, đặc tả, cấu trúc và làm tài liệu chế tác cho các hệ thống phầm mềm chuyên sâu. UML được tích hợp vào Rational Software Development Platform.

Dịch xuôi là khả năng phát sinh mã nguồn từ các mô hình thiết kế. Dịch ngược là khả năng tự động hoá việc xây dựng lại các mô hình thiết kế từ các thông tin trong mã nguồn.

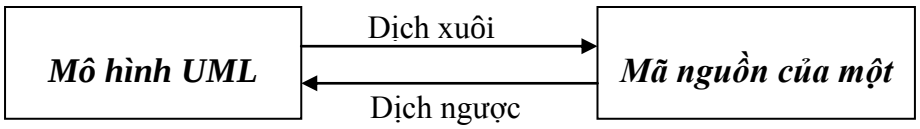
UML được thiết kế để hỗ trợ việc ánh xạ tất cả các mô hình thiết kế vào các ngôn ngữ triển khai và ngược lại ánh xạ từ mã nguồn vào mô hình. Đối với dịch xuôi, mã được phát sinh cho mỗi phần tử mô hình phụ thuộc vào đặc tả các phần tử của mô hình và các thuộc tính của nó. Cũng như vậy, khi dịch ngược, các thông tin trong mã nguồn sẽ quyết định các mô hình được phát sinh. Tuy nhiên kết quả của việc dịch xuôi và

dịch ngược còn phụ thuộc vào công cụ biểu diễn thiết kế việc dịch được hỗ trợ tới đâu, và cũng phụ thuộc vào ngôn ngữ lập trình triển khai được chọn để ánh xạ tới nó.

Rational Software Architecture là công cụ phong phú hơn bất cứ một ngôn ngữ lập trình hướng đối tượng nào hiện nay. Vì vậy, một số phần tử có trong mô hình sẽ không có phần tử tương ứng với nó trong một ngôn ngữ lập trình cụ thể, các phần tử đó sẽ bị bỏ qua trong cả hai quá trình dịch. Như vậy, khi dịch xuôi hay ngược sẽ xảy ra sự mất thông tin. Do đó, để có được hệ thống hoàn chỉnh thì sau khi dịch cần phải bổ sung thêm các thông tin bị mất mát vào kết quả thu được.

Việc dịch xuôi và ngược được thực hiện dựa trên các nguyên tắc ánh xạ một-một giữa các phần tử trong mô hình thiết kế và các phần tử trong mã nguồn. Kết quả của quá trình dịch xuôi có thể cho mã nguồn thực hiện được hoặc các tệp nhị phân lưu giữ thông tin về mô hình. Kết quả của việc dịch ngược sẽ cho các mô hình thiết kế ban đầu.

Ta có thể mô tả quá trình của tái kỹ nghệ trong UML như sau:



Hình 2-11 *Dịch xuôi và dịch ngược trong UML*

2.4. Thiết kế hệ thống bằng Rational Software Architecture

Thiết kế bằng tái kỹ nghệ là thiết kế lấy thông tin từ mã nguồn, rồi tạo hoặc cập nhật lại một mô hình trên Rational Software Architecture. Thông qua khả năng tích hợp của Rational Software Architecture với một ngôn ngữ lập trình khác, chẳng hạn như: C/C++, EJB, J2EE, XML ..., Rational Software Architecture hỗ trợ cơ chế thiết kế tái kỹ nghệ thành một mô hình UML.

Trong tiến trình thiết kế bằng tái kỹ nghệ, Rational Software Architecture sẽ thu thập các thông tin về: các lớp, các thuộc tính, các tác vụ, các mối quan hệ, các gói, các thành phần,… trong chương trình nguồn. Nó sẽ sử dụng các thông tin này để tạo mới hoặc cập nhật một mô hình đối tượng. Nếu ta có một tệp tin mã nguồn chứa một lớp, tiến trình thiết kế tái kỹ nghệ sẽ tạo ra một lớp tương ứng trong mô hình Rational Software Architecture, mỗi thuộc tính và tác vụ của lớp sẽ xuất hiện dưới dạng các

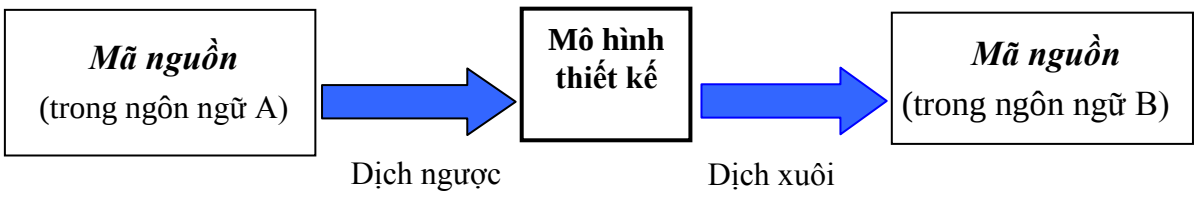
thuộc tính và các tác vụ của lớp mới trong mô hình Rational Software Architecture. Cùng với tên thuộc tính và tên tác vụ, Rational Software Architecture đưa thêm vào các thông tin về tầm hoạt động, kiểu dữ liệu và các giá trị ngầm định.

Nếu ban đầu ta dùng Rational Software Architecture để tạo ra các lớp, dùng kĩ thuật dịch xuôi để phát sinh mã trình, sau đó thực hiện một số thay đổi với các lớp trong mã trình, các thay đổi này sẽ được phản ánh trong tiến trình thiết kế với tái kỹ nghệ. Chẳng hạn, nếu xoá hay bổ sung một tác vụ trong mã trình, tác vụ này sẽ bị xoá hay bổ sung vào mô hình nhận được bằng tái kỹ nghệ.

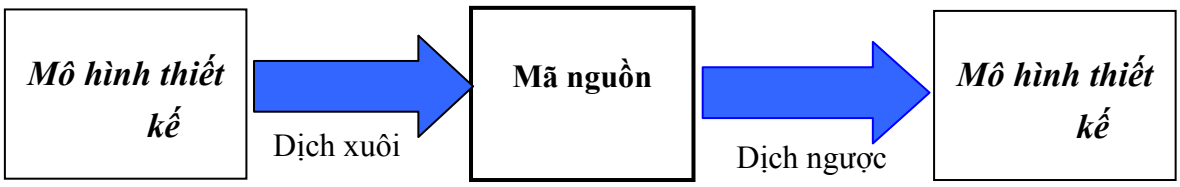
Ngoài các lớp, Rational Software Architecture sẽ thu thập thông tin về các mối quan hệ trong mã trình. Nếu một lớp chứa một thuộc tính có kiểu dữ liệu là một lớp khác, Rational Software Architecture sẽ tạo mối quan hệ giữa hai lớp. Với các mối quan hệ kế thừa Rational Software Architecture sẽ tạo các mối quan hệ tổng quát hoá để hỗ trợ mọi quan hệ trong mã trình.

Kết quả cuả quá trình đảo ngược sẽ cho mô hình của hệ thống phần mềm trên UML từ mã nguồn. Xuất phát từ đây ta có thể sửa đổi, cập nhật cho mô hình hệ thống, sau đó sẽ phát sinh mã trình cho mô hình hệ thống theo qui trình dịch xuôi.

Có thể mô tả các quá trình phân tích thiết kế và tái thiết kế như trong các sơ đồ sau đây: Trong hình 22 mô tả một bước lặp trong tiến trình tái thiết kế xuất phát là mã nguồn của một ngôn ngữ lập trình được UML hỗ trợ. Trong hình 23 mô tả một bước lặp trong tiến trình tái thiết kế xuất phát là mô hình thiết kế của UML đã được thiết lập trước đây.



Hình 2-12 *Một bước lặp của quá trình tái thiết kế với xuất phát là mã nguồn*



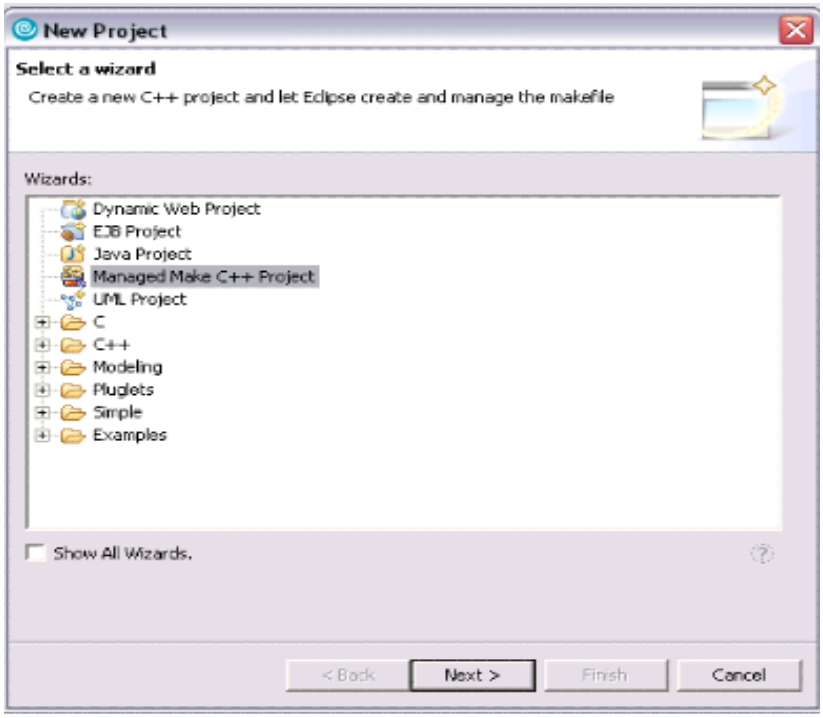
Hình 2-13 *Một bước lặp của quá trình tái thiết kế xuất phát là mô hình thiết kế*

Hình 12,13 mô tả một bước lặp trong tiến trình tái thiết kế. Ban đầu từ chương trình nguồn của hệ thống được lập trước đây, nhờ kĩ thuật thiết kế đảo ngược của Rational Software Architecture ta chuyển nó sang mô hình thiết kế trên UML. Tiếp đến là cập nhật, sửa đổi, hiệu chỉnh, bổ sung cho bản thiết kế này trên Rational Software Architecture. Sau đó lại sử dụng kĩ thuật dịch xuôi của Rational Software Architecture để chuyển bản thiết kế này sang mã nguồn. Quá trình lặp trên có thể được thực hiện nhiều lần nếu cần thiết, và sau một bước lặp đó ta sẽ được một thế hệ phần mềm mới có thêm các chức năng và những đặc tính mới. Trong quá trình tái thiết kế hệ thống phần mềm chúng ta có thể kết hợp hai sơ đồ trên với nhau.

Quá trình tái thiết kế phần mềm với Rational Software Architecture không những cho ta một hệ thống phần mềm mới hơn hẳn hệ thống ban đầu về tính năng, mà còn cho phép sinh ra hệ thống trong một ngôn ngữ lập trình khác ngôn ngữ ban đầu. Nhờ vậy, làm tăng tính khả chuyển (có khả năng hoạt động trên môi trường mới) của hệ thống mới nhận được.

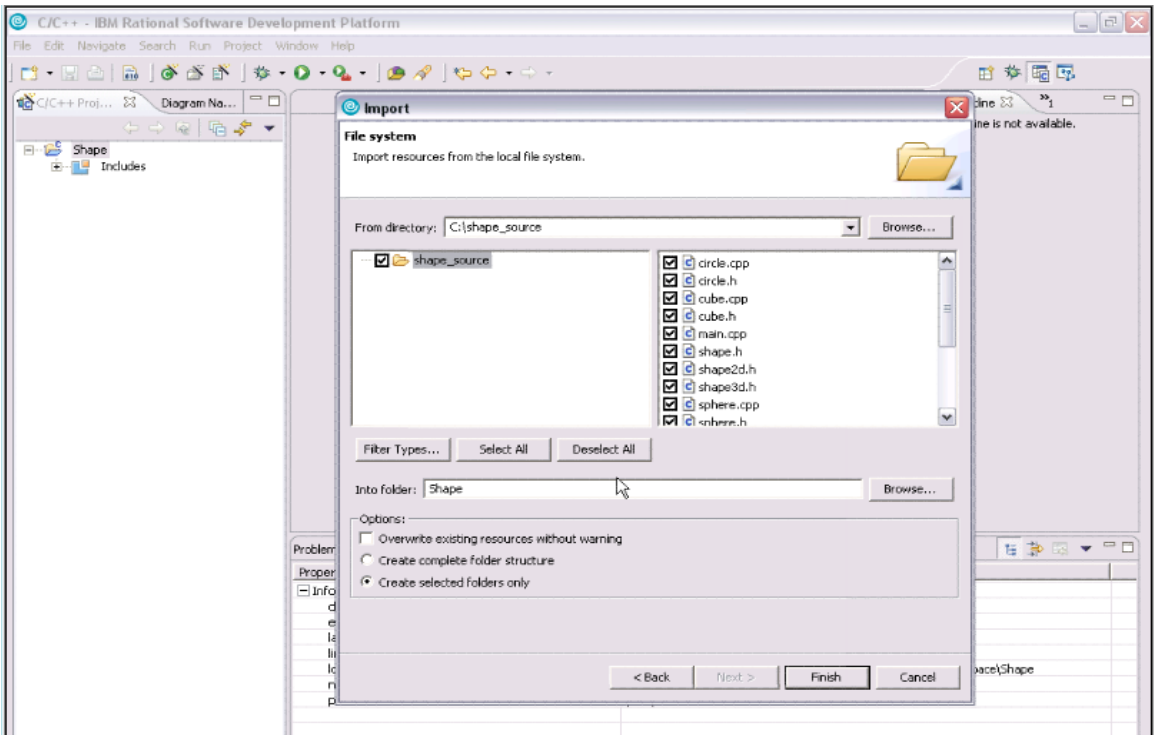
2.5. Phát triển ứng dụng C/C++ trên Rational Software Architecture

Để tạo một dự án ta vào *File –> New Project*. Rồi chọn kiểu maketệp tự động hay bằng tay dưới đây là một số kiểu bằng tay.

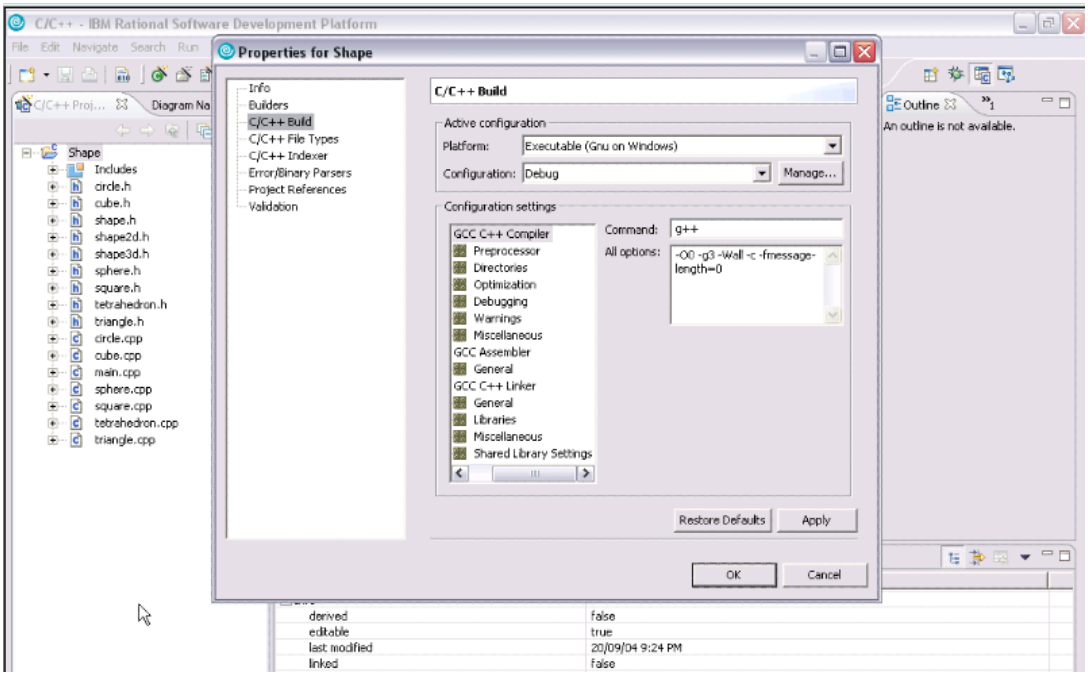


Hình 2-14 *Tạo dự án trên C/C++*

Tiếp tục chọn trình dịch và cấu hình mã nguồn. Cửa sổ làm việc gồm có Project View, Editor và của sổ biên dịch. Để tái kỹ nghệ, ta nhập các tệp chương trình nguồn vào bằng cách *File* → *Import* → *File system*



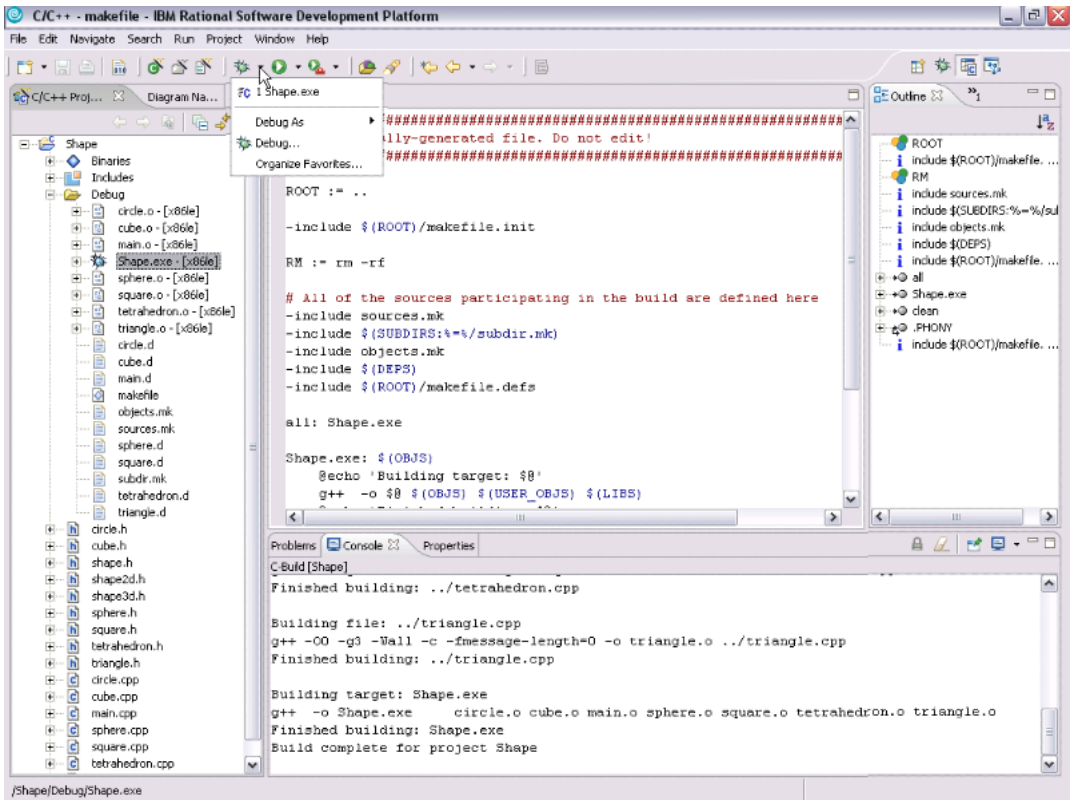
Hình 2-15 *Dịch chương trình*



Hình 2-16 *Cấu hình lại chương trình*

Sau khi nạp xong các tệp của chương trình thì có thể dịch chương trình – *build* sang chương trình đối tượng mã máy, rồi cấu hình lại chương trình, chọn dẫn hướng thiết lập cho chương trình.

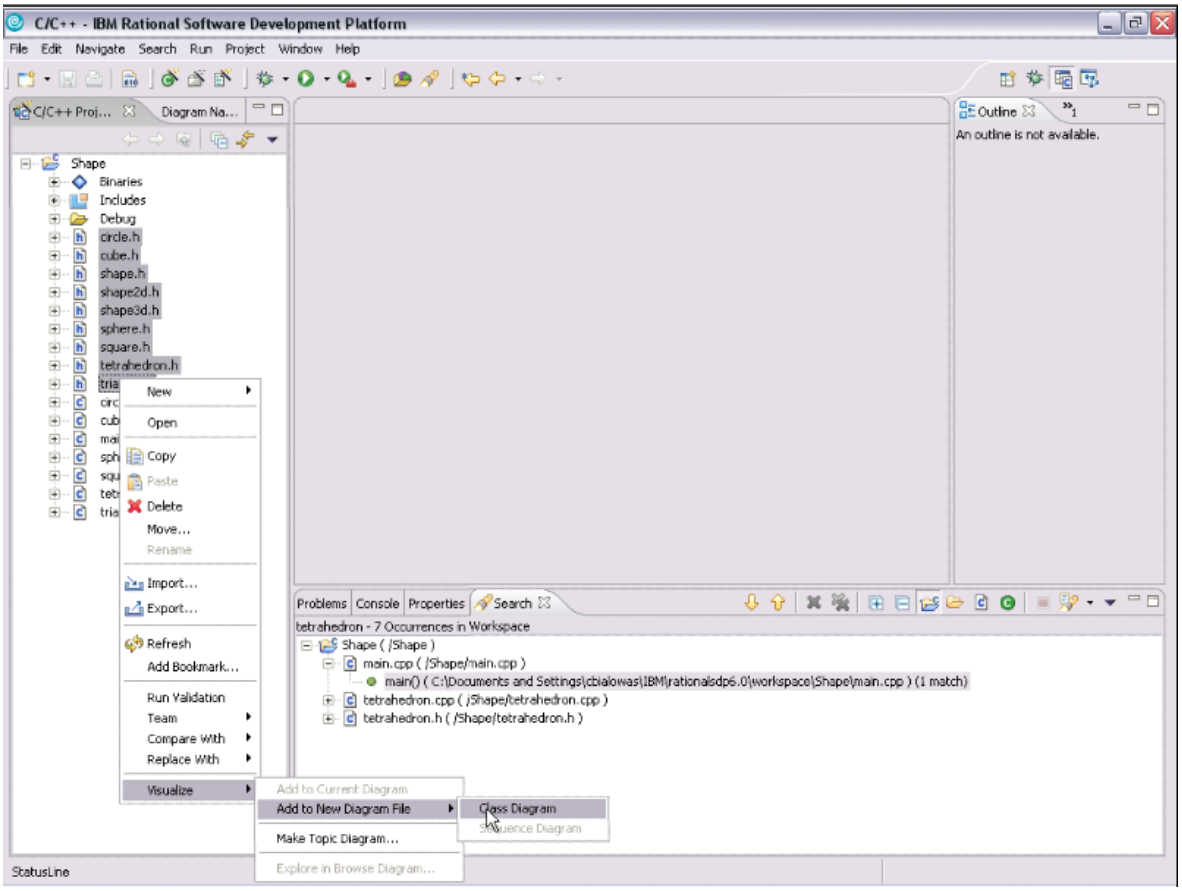
Một maketệp được tự động sinh ra trên cơ sở thiết lập. Chương trình có thể tự động được xây dựng nền bằng cách *Build project –Run in background*. Các Options có thể được chọn để cấu hình lại chương trình được như mong muốn bằng cách vào thư mục *Debug* chọn *maketệp* rồi chạy chương trình, Công cụ này cũng hỗ trợ điều khiển thực hiện chương trình và gỡ rối. Ví dụ: điều chỉnh lại biến, thay đổi các giá trị của biến, xem các thanh ghi và luồng dữ liệu.



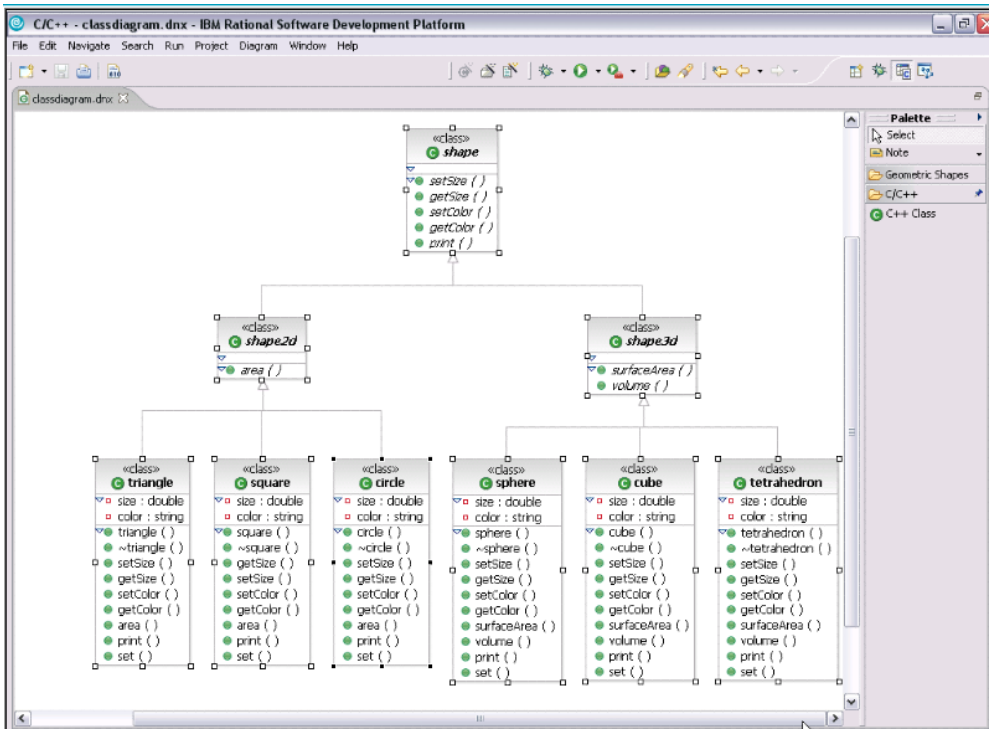
Hình 2-17 *Gỡ rối chương trình*

Khi gõ phím danh sách các thuộc tính và nội dung liên quan hỗ trợ. Việc tìm kiếm các khai báo và các thành phần của mã nguồn cũng được hỗ trợ.

Rational Software Architecture tích hợp bộ soạn thảo UML và soạn thảo mã nguồn. Do vậy, chúng ta có thể đồng thời sử dụng bộ soạn thảo UML để trực quan hóa và soạn các ứng dụng trên C/C++. Để làm việc này, ta bôi đen các tệp của C/C++ rồi click chuột phải chọn *Visualize ->Add a new Diagram tệp ->Class Diagram.*



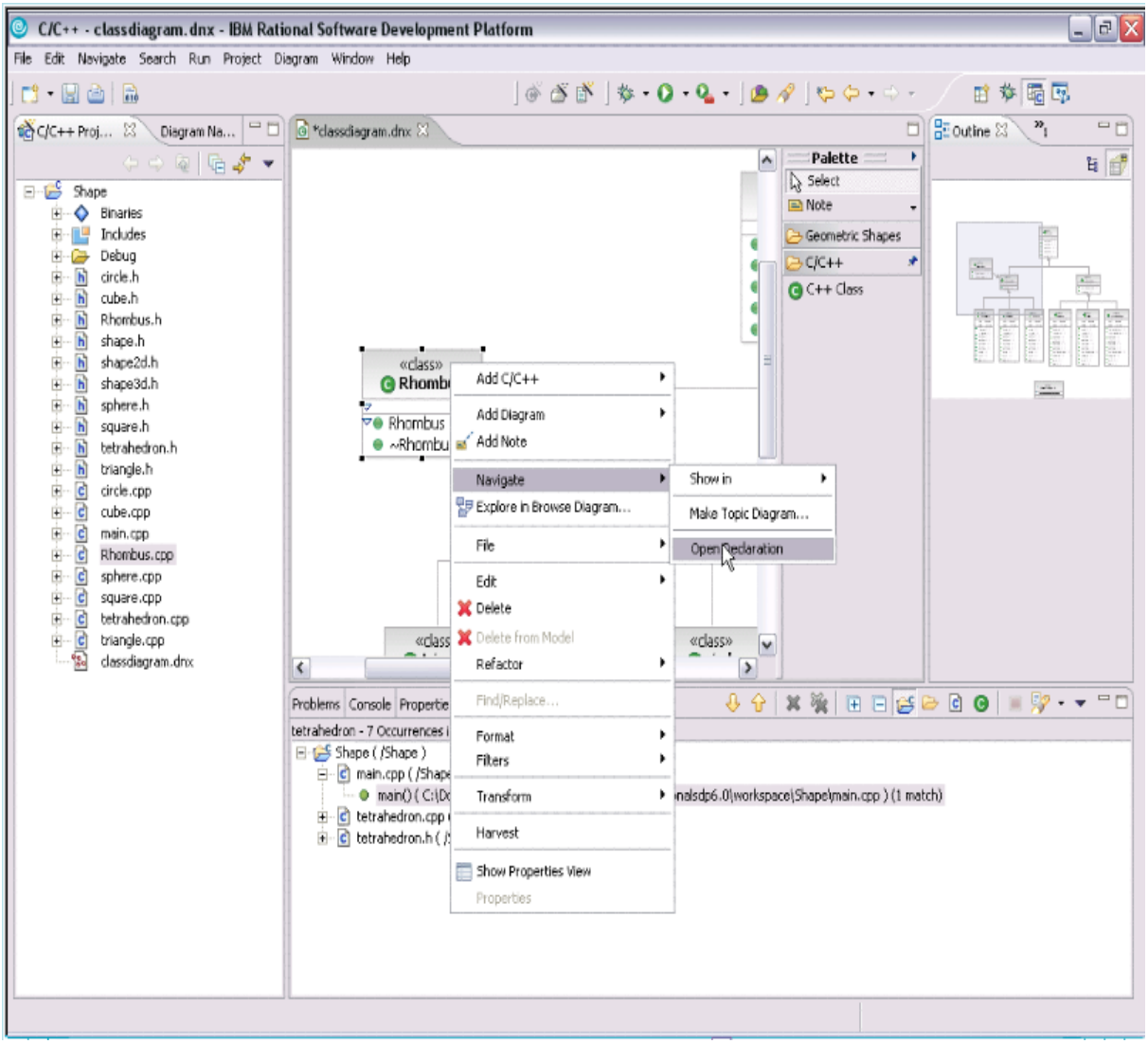
Hình 2-18 *Sử dụng bộ soạn thảo UML để trực quan hóa ứng dụng*



Hình 2-19 *Sử dụng bộ soạn thảo UML để soạn ứng dụng trên C/C++*

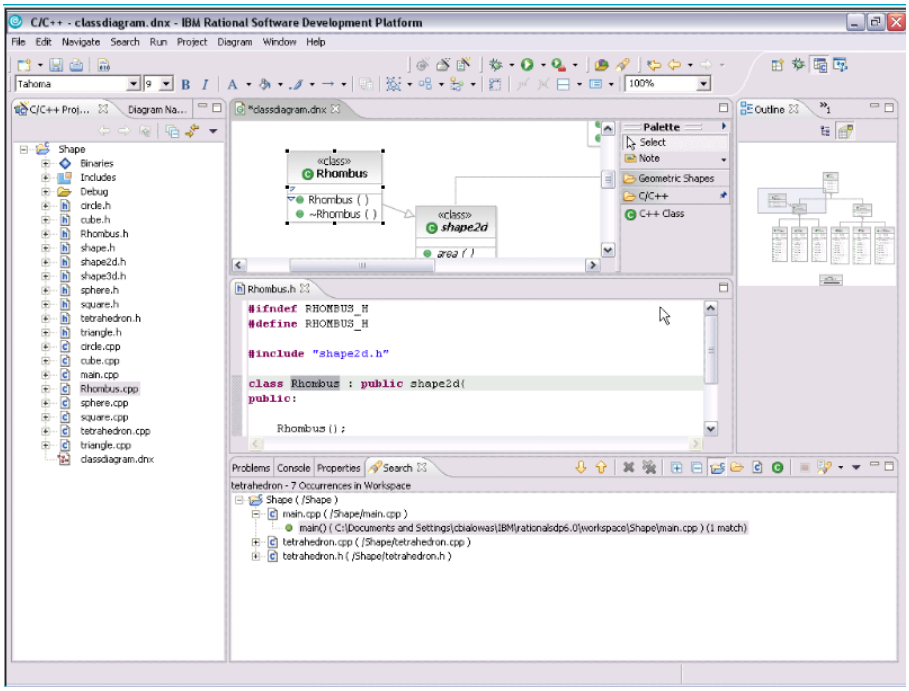
Chúng ta có thể thêm lớp mới vào ứng dụng bằng cách *add new class wizard*.

Chọn *C++ class*, một cửa sổ xuất hiện cho phép đặt tên, kiểu lớp và lớp kế thừa rồi nhấn *Finish* một lớp được tự động sinh mã tương ứng với việc thiết lập. Bộ soạn thảo mô hình cho phép ta điều khiển giữa UML và mã lệnh



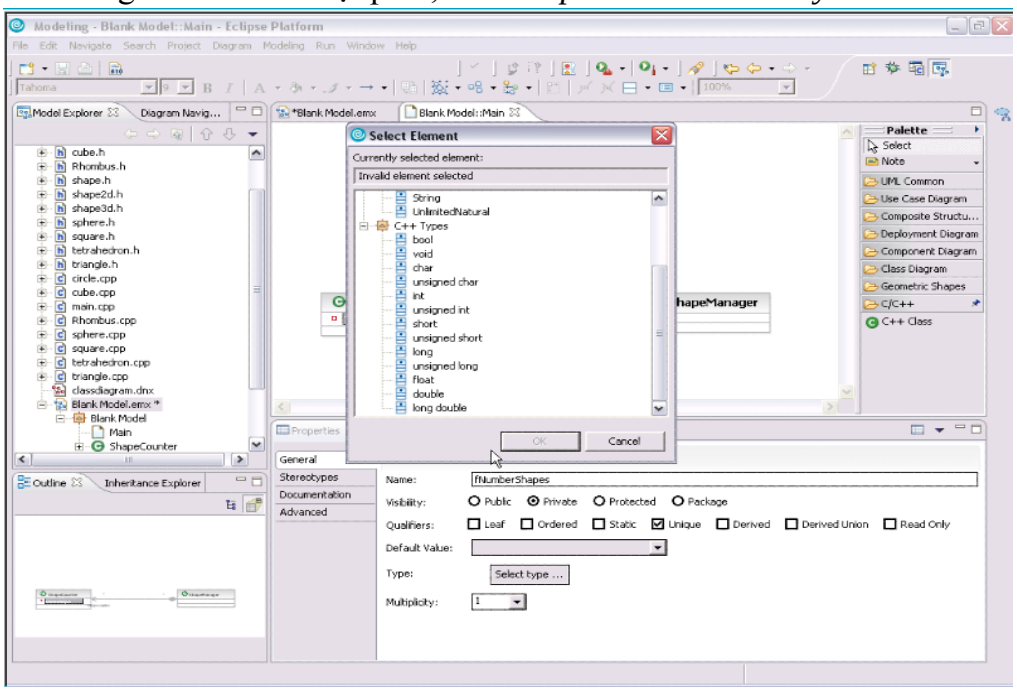
Hình 2-20 *Thêm lớp mới vào ứng dụng*

Khi soạn mã lệnh chương trình, thì trên mô hình UML tự động cập nhật. Ngược lại các mã nguồn của C++ cũng được sinh từ mô hình UML



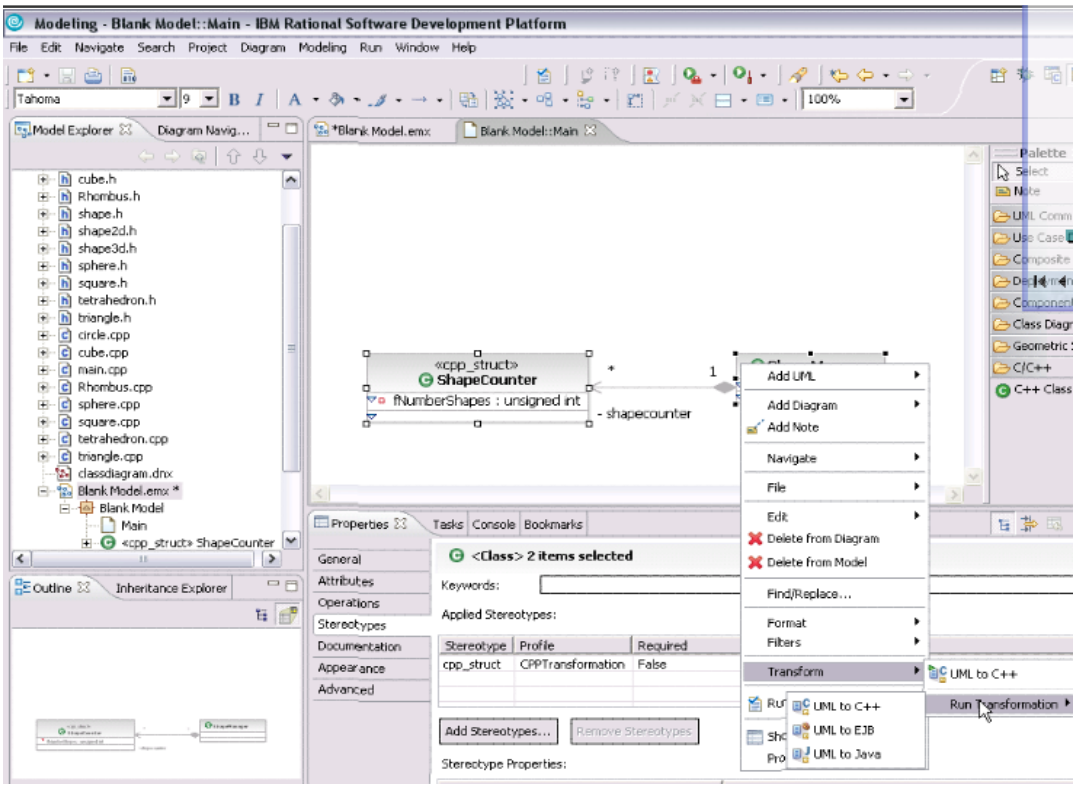
Hình 2-21 Mô hình UML tự động cập nhật

Mã nguồn C++ có thể sinh ra ở bất kỳ mô hình UML nào, tuy nhiên cần một hồ sơ cụ thể về dẫn hướng cho C++ để điều khiển quá trình sinh mã. Ta thực hiện điều này bằng cách add thêm các protệp, rồi chọn protệp mẫu tương ứng để chuyển đổi. Rational Software Architecture tích hợp các thư viện sẵn có để thiết lập các kiểu cụ thể của C/C++ bằng cách bấm chuột phải, chon *Import Model Library* .



Hình 2-22 Hồ sơ dẫn hướng cho C++ điều khiển quá trình sinh mã

Lựa chọn các thành phần của UML và các phép biến đổi C++ có liên quan để sinh mã vào dự án của mình. Cách làm: nhấn chuột phải vào thành phần của mô hình – rồi chọn *Transform - > Run transform - >UML to C++*, sau đó chọn sinh mã trên dự án đã tạo.



Hình 2-23 *Sinh mã vào dự án*

Mã được sinh là một thành phần trong dự án phát triển phần mềm.

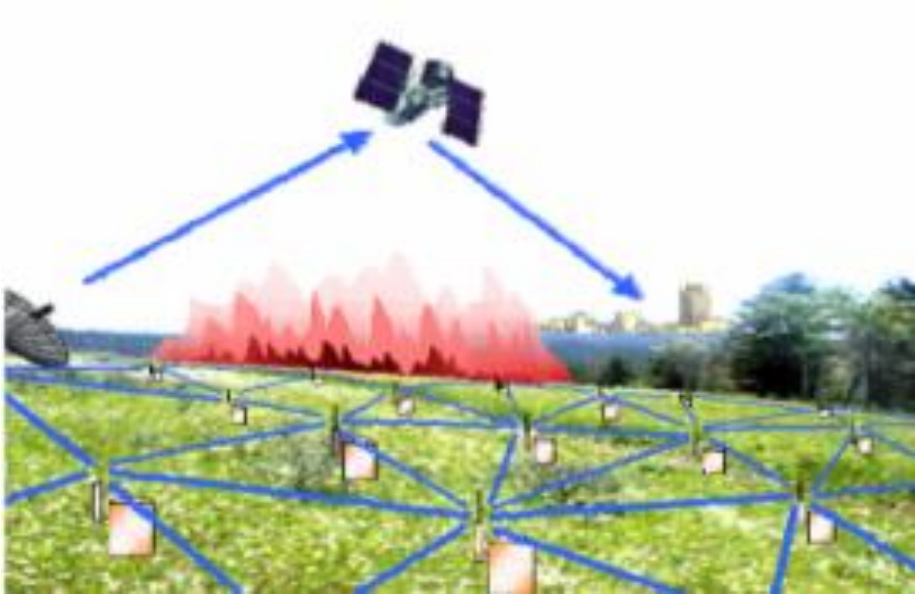
Bảo trì phần mềm là một công việc quan trọng đảm bảo cho hệ thống tồn tại và phát triển lâu dài, đỡ tốn nhiều công sức, thời gian và kinh phí. Ta có thể dùng kĩ nghệ đảo ngược của UML để thiết kế lại mô hình hệ thống phần mềm từ các thông tin mã nguồn được viết bằng một ngôn ngữ được chỉ ra. Việc tái kĩ nghệ lấy thông tin thu được và tái cấu trúc lại chương trình cho phép hệ thống đạt chất lượng cao hơn, thời gian ngắn hơn và đặc biệt có được tính ổn định cao.

Do UML là ngôn ngữ hỗ trợ mạnh cho phân tích và thiết kế hướng đối tượng, cho nên các hệ thống đó được xây dựng theo hướng đối tượng, thì việc tái thiết các hệ thống đó theo kĩ thuật đảo ngược của UML sẽ được thực hiện một cách thuận lợi. Tuy nhiên, còn nhiều vấn đề cần được thử nghiệm, đánh giá để có thể đề xuất một quy trình tái thiết kế tốt và những kinh nghiệm hữu ích khi tái thiết kế.

CHƯƠNG 3 TÁI KỸ NGHỆ TRONG HỆ THỐNG CẢNH BÁO THIÊN TAI

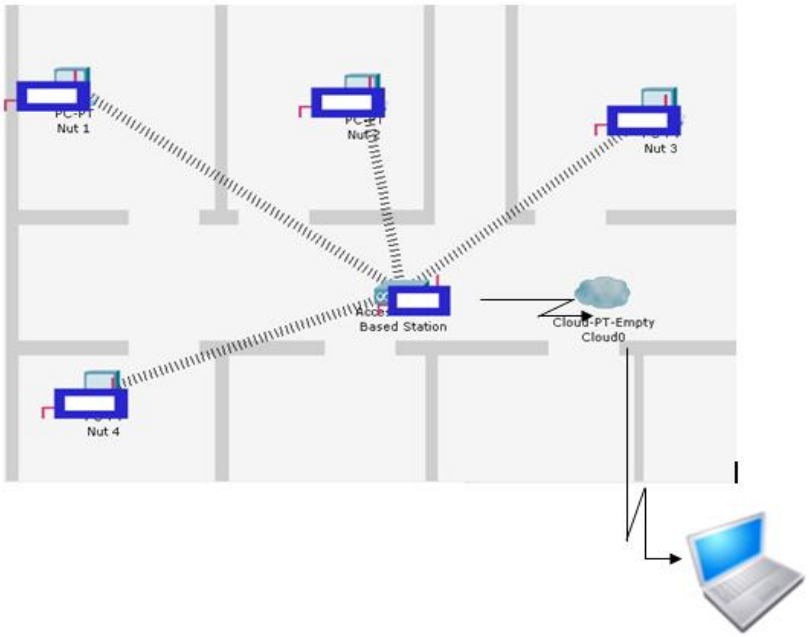
3.1. Cấu trúc hệ thống cảnh báo thiên tai

Hệ thống cảnh báo thiên tai sử dụng mạng cảm nhận không dây WSN là mạng mà trong đó các nút mạng có khả năng thực hiện chức năng mạng và chức năng cảm nhận. Nó là trường hợp riêng của mạng không dây Ad hoc vì ngoài chức năng mạng nó còn có chức năng cảm nhận môi trường. Mỗi nút mạng có chức năng như máy tính di động trong đó sử dụng hệ vi xử lý có kích thước rất nhỏ, (nhỏ đến mức người ta có thể chế tạo võng mạc nhân tạo bằng nhiều nút mạng cảm nhận). Mô hình cảnh báo cháy rừng được trang bị các nút mạng cảm nhận thông tin nhiệt độ từ môi trường, chuyển thông tin đó thành tín hiệu số rồi truyền về nút cơ sở - nút này được nối với máy tính phục vụ có thể tự động cập nhật lên Website sau mỗi chu kỳ thời gian hoặc được truyền về trung tâm xử lý thông qua internet hay vệ tinh.



Hình 3-11 *Hệ thống cảnh báo cháy rừng*

3.2. Hệ thống cảnh báo thiên tai



Hình 3-2 Hệ thống cảnh báo hiện có

Cấu trúc hệ thống cảnh báo gồm: bốn nút mạng và một trạm cơ sở, các nút truyền tới trạm cơ sở là truyền đơn bước, chúng thu thông tin nhiệt độ phát về nút cơ sở, nút cơ sở truyền về trung tâm xử lí, tại đó đặt ngưỡng về nhiệt độ (85 độ)để so sánh với thông tin thu về. Nếu thông tin thu về của các nút vượt quá ngưỡng thì cảnh báo cho vị trí của nút đó.

3.2.1. Mô tả hoạt động của hệ thống cảnh báo thiên tai

Hệ thống cảnh báo thiên tai hoạt động phụ thuộc vào các Sensors cảm nhận từ môi trường: như sensor cảm nhận nhiệt độ, cảm nhận ánh sáng, cảm nhận mức nước dâng lên, cảm nhận độ Hp, tốc độ gió vv.. Mỗi sensor chỉ cảm nhận được một thông số của môi trường. Thông tin cảm nhận được được chuyển thành tín hiệu số nhờ bộ ADC, trên mỗi nút mạng có gắn bộ thu phát tín hiệu.

Cấu trúc mạng cảm nhận không dây gồm nhiều nút mạng không dây liên kết với nhau thông qua bộ thu phát sóng vô tuyến để truyền tín hiệu mà nút mạng cảm nhận được từ môi trường truyền về trạm cơ sở mạng này kết nối với trung tâm xử lý.

Kiến trúc của một nút mạng cảm nhận không dây gần tương tự như kiến trúc của một máy tính thông thường, gồm các thành phần chính như: Bộ vi xử lý, bo mạch, bộ nhớ, bộ truyền thông, bộ cảm biến, và bộ khởi động.

Bộ vi xử lý (bộ xử lý dữ liệu: có nhiều loại): Xử lý dữ liệu thu thập từ môi trường, xử lý tín hiệu truyền nhận giữa các nút mạng do cấu tạo của nút mạng rất nhỏ nên bộ vi xử lý phải đáp ứng được kiến trúc nhỏ, tiêu thu điện năng ít...(ví dụ như bộ xử lý CC1010 của hãng Chipcon).

***Bo mạch*:** Bao gồm nguồn nuôi, các cổng giao tiếp và là nơi để tích hợp các thiết bị như : bộ cảm biến, bộ lưu trữ dữ liệu, bộ truyền thông…

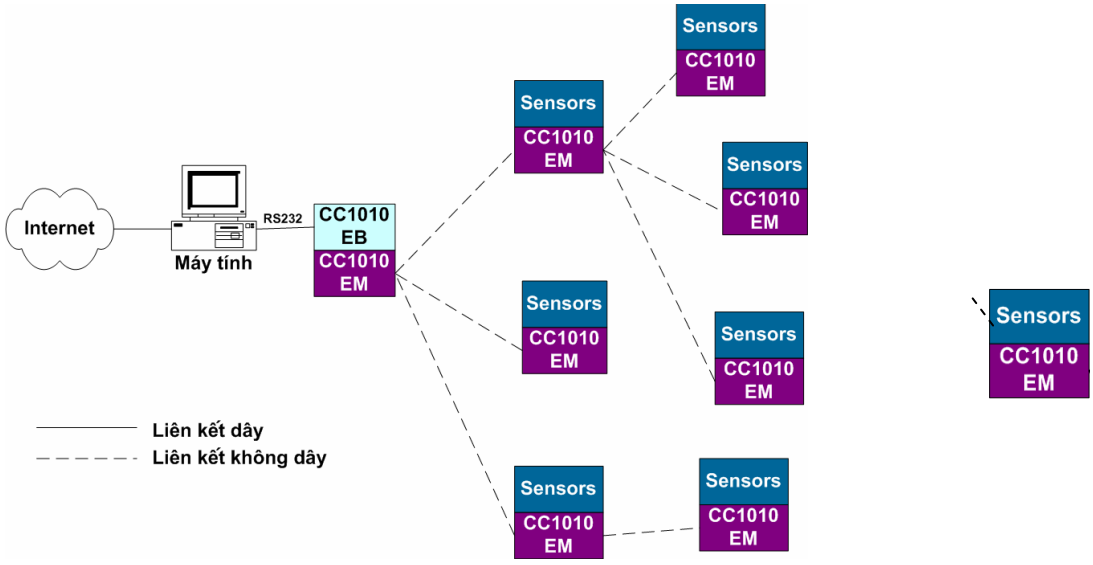
Bộ nhớ: **C**ác nút mạng cảm nhận không dây có những thành phần lưu trữ rất nhỏ. Chúng thường sử dụng bộ nhớ DRAM và Flash. Việc truyền thông chính là thành phần tiêu thụ năng lượng chính của mạng cảm nhận không dây. Vì vậy, nếu chúng ta mong muốn khả năng lưu trữ tại mỗi nút mạng sẽ tăng lên thì phải quản lý được năng lượng của mạng.

Bộ truyền thông: Gồm ăng ten (thu, phát sóng radio với bước sóng trong giải không cần cấp phép của ISM).

Mô hình truyền thông thường được đề cập trong mạng cảm nhận không dây hiện thời là việc truyền thông đa bước theo kiến trúc bó. Các kết quả hiện thời chỉ ra rằng việc truyền thông đa bước theo kiến trúc bó sẽ tiết kiệm được năng lượng chỉ có nút gốc sẽ tiêu thu điện năng khá lớn vì chính việc lắng nghe yêu cầu năng lượng ngang bằng với việc truyền thông tin.

Bộ cảm biến: Bộ cảm biến là các secsors, dùng để thu nhận các thông số bên ngoài môi trường. Có rất nhiều loại cảm biến như cảm biến quang, cơ, nhiệt....

Bộ khởi động: Nếu coi bộ cảm biến như là đôi mắt của mạng cảm nhận không dây thì bộ khởi động như là cơ bắp của nó.



Hình** 3-3 **Cấu trúc mạng cảm nhận không dây

Cấu trúc mỗi nút mạng gồm có một bộ vi xử lý họ Intel 8051, bộ nhớ flash 32kb (2048 + 128 Byte SRAM), có 4 bộ định thời, có 2 cổng UART- RTC, có 3 kênh ADC 10 Bit, giao diện lập trình SPI, bộ mã hóa DES tích hợp bên trong, có 26 chân vào ra chung, cần rất ít thành phần bên ngoài, có bộ thu phát sóng vô tuyến 300 – 1000 MHz, có bộ cảm nhận thông tin môi trường độ nhạy cao, nguồn nuôi từ 2.7 – 3.6V, tiêu thụ dòng thấp (9.1 mA trong chế độ thu), công suất phát có thể lập trình được (lên đến +10 dBm), tốc độ thu phát dữ liệu lên đến 76.8 kbit/s. Khi chương trình được nạp vào bộ nhớ của nó, nó sẽ hoạt động theo chương trình nằm trong bộ nhớ, hoạt động của nó thu nhận thông tin môi trường tùy thuộc bộ cảm nhận nào thì thu thông tin về môi trường theo bộ cảm nhận đó, hoặc thu thông tin từ một nút mạng khác rồi truyền thông tin đó về trạm khác hoặc trạm cơ sở.

3.2.2. Ưu điểm của hệ thống cảnh báo thiên tai

Hệ thống cảnh báo thiên tai sử dụng mạng cảm nhận không dây WSN có nhiều ưu điểm:

- Dễ triển khai lắp đặt ở các môi trường đặc biệt, vì không cần quan tâm tới việc lắp đặt nó tự tổ chức thành mạng truyền thông nhau.
- Chi phí chấp nhận được vì công nghệ tích hợp.

- Độ bao phủ tương đối.
- Thời gian đáp ứng nhanh, chính xác.
- Tính bảo mật cao
- Tốc độ lấy mẫu hiệu quả.

3.2.3. Nhược điểm của hệ thống cảnh báo thiên tai

- *Thiếu năng lượng hoạt động cho các nút*: Hệ thống sử dụng nguồn nuôi bằng PIN cho nên năng lượng sử dụng là hữu hạn. Mạng càng hoạt động nhiều mức độ tiêu hao năng lượng càng lớn. Khi mở rộng mạng, các nút mạng càng phải hoạt động nhiều hơn nữa, mạng sẽ tiêu tốn càng nhiều năng lượng. Do đó, thời gian hoạt động của các nút mạng sẽ giảm, dẫn đến thời gian hoạt động của toàn hệ thống giảm, như thế sẽ không đảm bảo hiệu quả cho hệ thống. Vậy, vấn đề đảm bảo năng lượng cho nút mạng và toàn mạng hoạt động lâu dài là vấn đề cần được quan tâm đối với các nhà thiết kế hệ thống.

- *Khả năng xử lý và bộ nhớ nhỏ*: Khả năng xử lý của bộ vi xử lý tuy đã đáp ứng được nhu cầu, tuy nhiên cũng chưa đủ mạnh nếu lượng thông tin xử lý lớn. Khả năng lưu trữ thông tin của các nút cũng vẫn chưa nhiều. Do vậy, dữ liệu từ các nút mạng chủ yếu được tập trung xử lý tại nút cơ sở. Việc này làm cho lưu lượng dữ liệu truyền trên mạng nhiều, dẫn đến mạng phải hoạt động thường xuyên hơn. Đây cũng chính là nguyên nhân làm cho năng lượng tại các nút mạng giảm đi, làm giảm thời gian sống của các nút mạng.

- *Giao thức truyền*: Mạng hoạt động dựa vào giao thức truyền bình đẳng và nhóm giao thức phân cấp. Trong nhóm giao thức bình đẳng các nút mạng bình đẳng và hoạt động độc lập với nhau, ngược lại trong nhóm giao thức phân cấp, một nút mạng sẽ giữ vai trò điều khiển hoạt động của các nút trong bán kính phủ sóng của nó. Trong nhóm giao thức bình đẳng, mỗi nút khi muốn truyền tin đều thực hiện việc truyền thống báo cho mọi nút còn lại trong nhóm. Giao thức này có nhược điểm lớn là khi một nút hoạt động sẽ kéo theo toàn nhóm phải hoạt động. Như vậy, sau hoạt động của mỗi nút sẽ làm cho năng lượng của toàn mạng giảm đi rất nhiều và tuổi thọ của mạng cũng sẽ bị rút ngắn, bên cạnh đó, hiệu quả truyền phát gói tin cũng không cao.

- *Do thay đổi về yêu cầu*: mở rộng kích thước của hệ thống, số lượng nút mạng.

3.2.3.1. Lựa chọn giải pháp tái kĩ nghệ

Giải pháp tìm đường ngắn nhất hay giải pháp xây dựng giao thức định tuyến cho mạng đều không nằm ngoài mục đích muốn giảm thiểu chi phí năng lượng cho các nút mạng để kéo dài tuổi thọ của chúng. Để thực hiện điều này, ta có thể tham khảo một số cách sau:

- *Thiết kế mới*: Thiết kế và cài đặt mới phần mềm theo tư tưởng "đan mới nhanh hơn dặm lại": Trong trường hợp này chưa hẳn đã đúng, vì giải pháp tạo mới này sẽ phức tạp và rất tốn kém. Nó liên quan đến việc tạo mới phần cứng, các thư viện của phần cứng. Giải pháp này không những khó khăn, tốn kém mà còn thực sự không thiết thực, không hiệu quả.

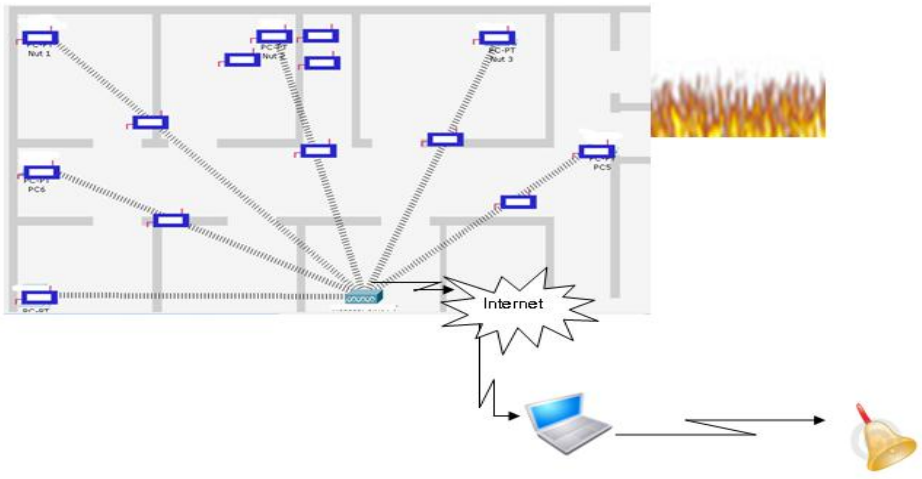
- *Thiết kế bổ xung thêm chức năng:* Cách làm này dễ làm cho hệ thống trở nên phức tạp và bị chồng chéo khiến cho việc bảo trì sau này gặp nhiều khó khăn.

- *Sử dụng lại:* Đây là giải pháp lấy các thành phần của một số sản phẩm để phát triển sản phẩm khác với tính năng khác. Trong trường hợp này, phần mềm nhúng thường có sự gắn kết chặt chẽ với phần cứng. Do đó, giải pháp này là không khả thi.

- *Tái kỹ nghệ:* Đây là quá trình xem xét tìm hiểu một hệ thống với mục đích thực hiện một dạng mới (cải tiến) cho hệ thống.

Trong các giải pháp đưa ra ở trên ta thấy, chỉ có giải pháp tái kĩ nghệ là khả thi nhất, vì giải pháp này là phù hợp nhất, vừa giải quyết được yêu cầu đặt ra, vừa kinh tế, ít tốn kém nhất, hiệu quả cao nhất.

3.3. Tái kĩ nghệ hệ thống cảnh báo thiên tai



Hình 3-4 Yêu cầu hệ thống mới

Như phần trên, chúng ta thấy rằng hệ thống cảnh báo thiên tai sử dụng mạng cảm nhận không dây WSN đã phát huy nhiều tính năng tốt, bên cạnh đó cũng có nhiều nhược điểm. Tái kĩ nghệ cho hệ thống cảnh báo thiên tai nhằm khắc phục những nhược điểm đó:

-Tăng phạm vi hoạt động của các nút mạng do: Độ phức tạp của phương thức truyền thay đổi, chương trình cũ không thích hợp.

-Tăng khoảng cách: Khoảng cách cũ từ nút mạng đến trạm cơ sở <=100m nên không thể thu phát được tín hiệu xa, vì thế ta phải mở rộng khoảng cách này từ đơn chặng chuyển thành đa chặng..

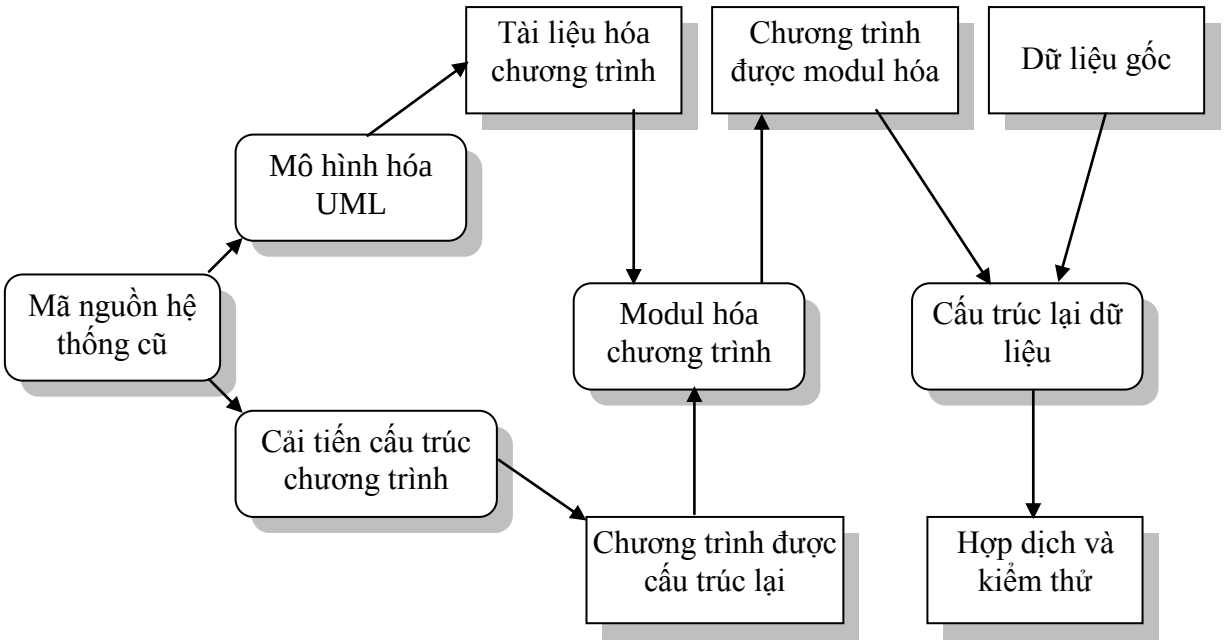
-Kéo dài tuổi thọ của các nút mạng: Hệ cảm nhận không dây sử dụng nguồn nuôi bằng pin cho nên vấn đề bảo đảm năng lượng cho nút mạng và toàn mạng hoạt động lâu dài là vấn đề cần quan tâm.

Ngoài ra, còn có các kĩ thuật thay đổi về định tuyến và xâm nhập môi trường cho từng nút mạng:

- *Tìm đường ngắn nhất*: Để khắc phục nhược điểm năng lực xử lý và dung lượng bộ nhớ ta có thể nâng cao năng lực xử lý và cung lượng bộ nhớ. Tuy nhiên, ta không chủ động làm được việc này mà bị phụ thuộc vào nhà cung cấp phần cứng. Về mặt phần mềm, ta vẫn có thể có giải pháp để khắc phục được một phần nhược điểm trên. Để giảm việc tiêu thụ năng lượng do dữ liệu phải truyền về nút cơ sở để xử lý, mỗi nút mạng phải duy trì một bảng định tuyến để hướng gói tin đi con đường tối ưu nhất, ít tốn năng lượng nhất.
- *Xây dựng giao thức định tuyến cho mạng*: Ngoài việc cần nâng cao hiệu quả chuyển phát các gói tin, phải cân nhắc tới vấn đề tiêu thụ năng lượng. Thời gian sống của mạng WSN phụ thuộc năng lượng tiêu thụ tại mỗi nút mạng (mà chủ yếu là do quá trình truyền thông). Do vậy, cần phải xây dựng giao thức định tuyến cho mạng. Để thay đổi giao thức định tuyến ta có thể đưa *tham số thời gian thiết lập* để không bị phát quảng bá lặp theo chu trình, và thay đổi một số yếu tố khác về vấn đề tiết kiệm năng lượng.

3.4. Tiến trình tái kỹ nghệ hệ thống cảnh báo thiên tai

3.4.1. Sơ đồ tiến trình



Hình 3-5 *Quy trình tái kỹ nghệ hệ thống cảnh báo thiên tai*

3.4.2. Các bước thực hiện

Bước 1. Từ mã nguồn hệ thống cũ đưa về dạng mô hình.

Hệ thống cảnh báo thiên tai: cảm nhận thông tin từ môi trường thông qua sensor, thông tin thu được (ví dụ thông tin nhiệt độ môi trường) phát truyền về trạm cơ sở.

Bước 2. Từ mô hình trực quan cấu trúc lại chương trình

Chương trình nên được tái cấu trúc một cách tự động hoặc bằng thủ công để bỏ đi những thành phần thừa, hoặc cấu trúc sai. Các điều kiện nên được đơn giản hóa để dễ hiểu. Trong bước này môdul hóa lại chương trình và xây dựng tài liệu thiết kế.

Bước 3. Tái kỹ nghệ dữ liệu

Phân tích và tổ chức lại cấu trúc dữ liệu trong chương trình, thêm thông số thời gian trong pha thiết lập; thay đổi lại thông tin thời gian, giá trị tần số

thu phát và quảng bá, thiết lập chế độ ngủ Hibernate khi nút mạng ở trạng thái đợi,...

Bước 4. Hợp dịch và nạp lại cho nút mạng để vận hành hệ thống

Phân biệt sự khác nhau của mã nguồn, thiết lập bộ từ mẫu dịch giữa UML và mã nguồn. Hệ thống được tái kỹ nghệ nên được dịch mã tự động. Sử dụng công cụ nạp ROM, kiểm thử sau mỗi công đoạn.

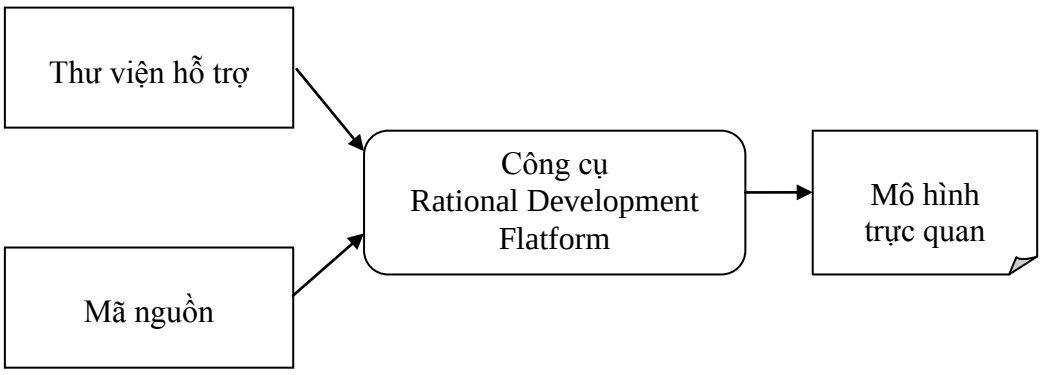
Bước 5. Đánh giá kết quả sau bước thay đổi

Vận hành hệ thống mới, kiểm thử toàn hệ thống. Thực hiện các phép đo, đánh giá, so sánh kết quả trước và sau khi thay đổi.

Các bước của quá trình trên được chi tiết hóa thông qua việc ứng dụng tiến trình RUP về phát triển hệ thống của Rational Architurect Software trong việc thay đổi thiết kế và tài liệu hóa cho hệ thống cảnh báo. Quy trình tái kỹ nghệ hệ thống cảnh báo được mô tả như hình 4.1.

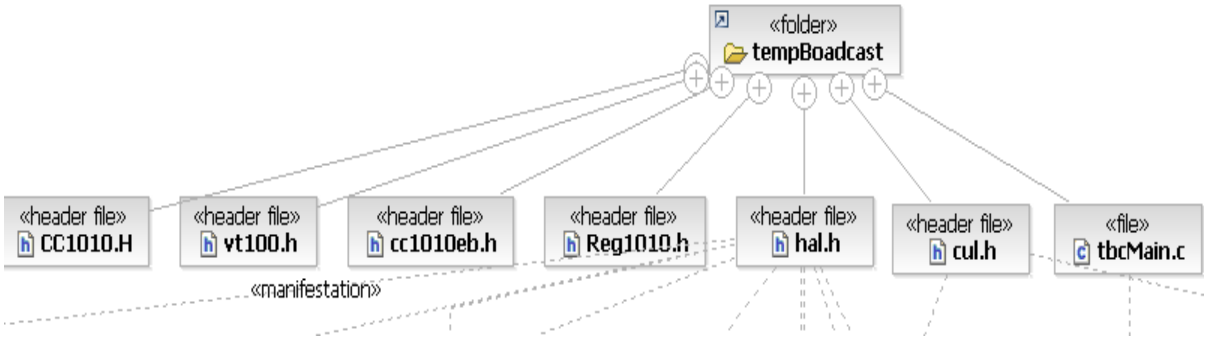
3.4.2.1. Từ mã nguồn của hệ thống chuyển sang mô hình trực quan

Sử dụng mô hình hóa trực quan giúp ta mô tả hệ thống một cách dễ dàng, việc chuyển hóa này nên được thực hiện một cách tự động thông qua công cụ của Rational. IBM Rational Software Architecture đã được đề cập chương trước. Công cụ này cho phép phân tích, thiết kế, phát triển, thử nghiệm và triển khai lại hệ thống cảnh báo thiên tai một cách dễ dàng. Các nút mạng cảm nhận không dây của hệ thống được lập chương trình, nạp vào chíp vi điều khiển CC1010. Để lập được chương trình cần có sự hỗ trợ của các thư viện thành phần cung cấp bởi hãng Chipcon sử dụng vào ra cổng và thanh ghi một cách dễ dàng.



Hình 3-6 *Từ mã nguồn hệ thống chuyển sang mô hình trực quan*

Theo quy trình phát triển RUP thì hệ thống cần được phần tích, thiết kế và kiến trúc thành phần trên mô hình. Bản công cụ phát triển phần mềm IBM Rational Development Flatform cho phép dễ dàng đưa mã nguồn của phần mềm đã có sẵn vào mô hình một cách tự động để có thể thay đổi hoặc thiết kế lại và kiến trúc lại hệ thống. Một điều rất tuyệt của công cụ này là ngay trên mô hình có thể sinh mã và sửa mã theo. Điều này giúp ta cho ta có cái nhìn tổng quan về thiết kế, dễ dàng đọc hiểu và phát triển chương trình.



Hình 3-7 *Các thành phần của chương trình được chuyển về mô hình*

Thao tác Xây dựng mô hình trong UML model

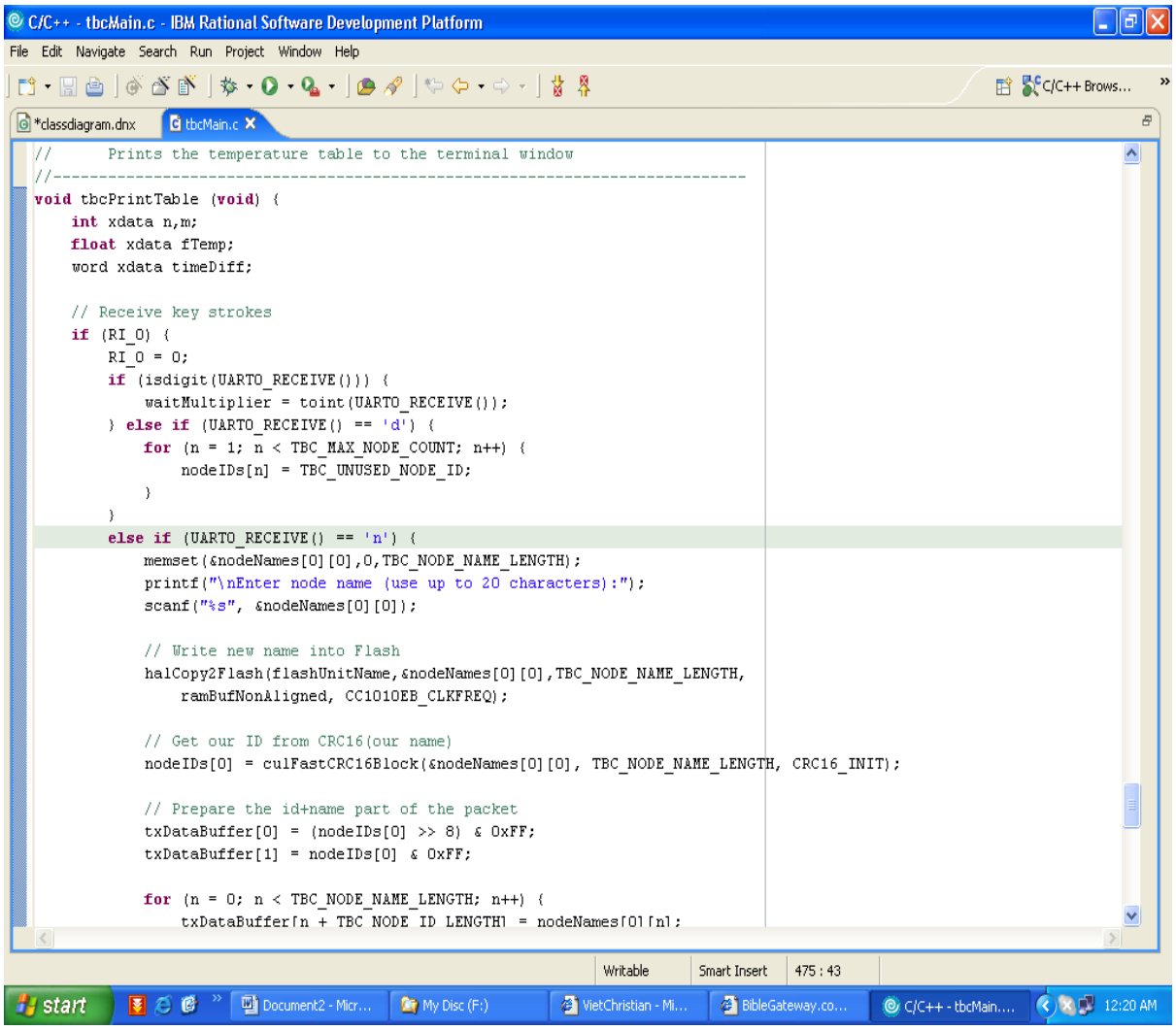
1. Từ Rational Software Architect, tạo mô hình.

   Click File --> New --> UML model.

2. Chấp nhận giá trị ngầm định, click *Finish.*

3. Trong của sổ *Model Explorer view*, bấm chuột phải chọn *new UML model* (biểu tượng này là:); rồi click *Add UML --> Class*

4. Muốn thay đổi đoạn mã nào trong các thủ tục ta chỉ việc click đúp chuột và thủ tục đó để cửa sổ màn hình mã lệnh xuất hiện (hình 4.4).

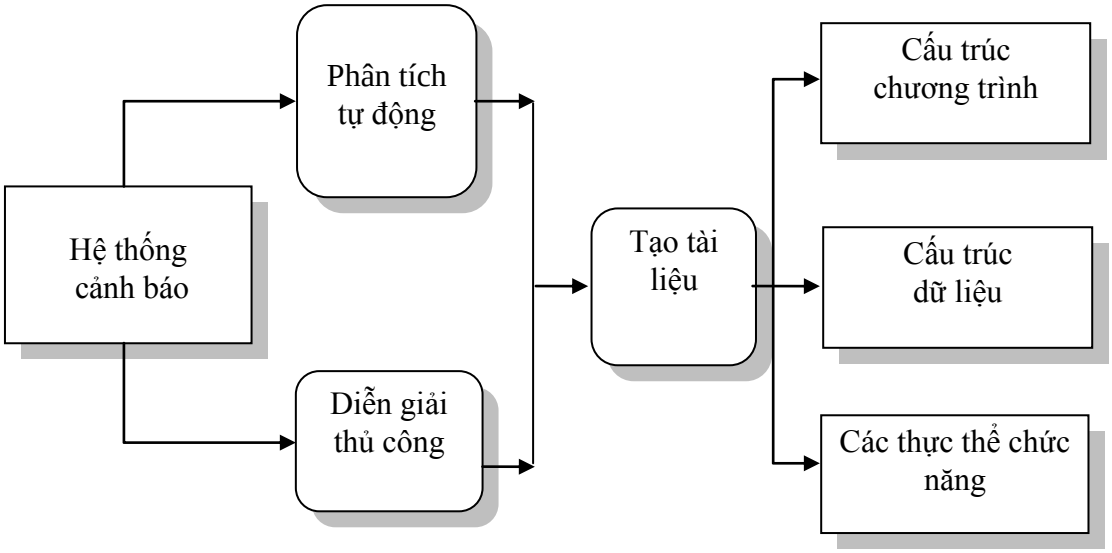


Hình 3-8 Cửa sổ màn hình soạn thảo mã nguồn

3.4. 2.2. Từ mô hình trực quan cấu trúc lại chương trình

a. Kỹ nghệ ngược

Kỹ nghệ ngược thường là sự tái hiện kỹ nghệ trước đó, một hệ thống được kỹ nghệ ngược sẽ là đầu vào cho tiến trình đặc tả các yêu cầu, hỗ trợ quá trình bảo trì sau này. Để phân tích được một phần mềm phải dựa trên quan điểm hiểu thiết kế và đặc tả của nó, có thể là một phần của tiến trình tái kỹ nghệ, cũng có thể là quá trình cụ thể hóa lại một hệ thống cho việc cài đặt lại. Nhờ đó xây dựng được bộ dữ liệu chương trình và sinh dữ liệu từ nó.



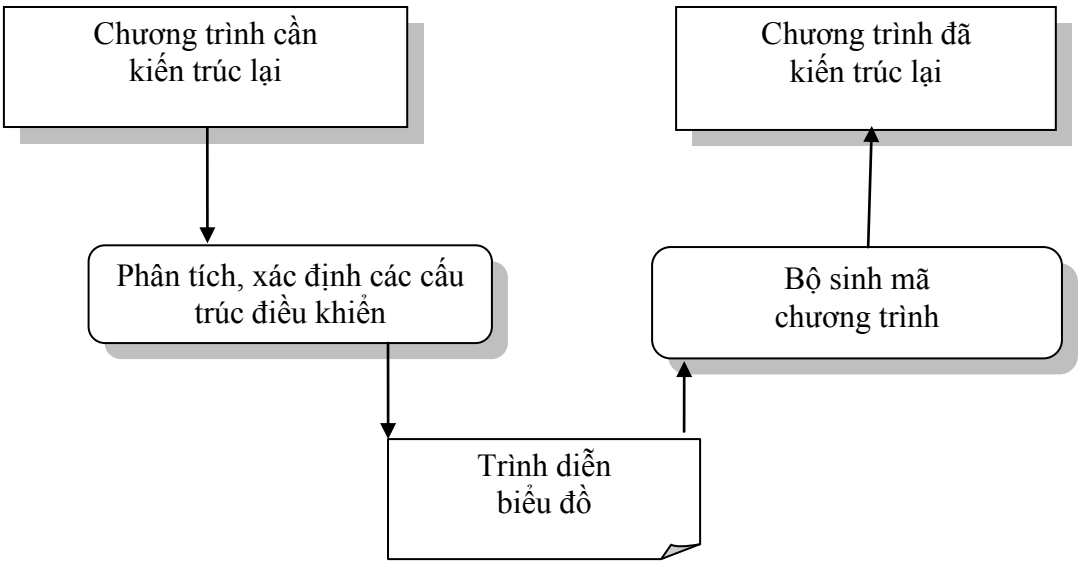
Hình 3-9 *Sơ đồ tiến trình cấu trúc chương trình*

Việc áp dụng quy trình tái kỹ nghệ cho hệ thống cảnh báo thiên tai với mạng cảm nhận không dây có đề cập vấn đề cấu trúc lại chương trình, dữ liệu và cải thiện một số chức năng chương trình để phù hợp với yêu cầu mở rộng hệ thống và tiết kiệm năng lượng cho mạng, đồng thời giúp hiểu hệ thống và dễ bảo trì sau này.

Tiết kiệm năng lượng cho nút mạng cảm nhận không dây nhằm kéo dài thời gian sống của nút mạng, cụ thể là: tiết kiệm năng lượng dựa trên hoạt động truyền nhận không dây, dựa trên việc thiết lập chế độ làm việc của nó, ngoài ra phụ thuộc yếu tố tiêu thụ điện năng của thiết bị phần cứng. Việc thay đổi các chế độ và cách chuyển đổi giữa các chế độ của hệ vi xử lý CC1010 thông qua tần số đã giúp cho việc thay đổi chế độ làm việc của nút mạng WSN một cách linh hoạt trong phần mềm nhúng. Khi chọn chế độ được thực hiện bằng phần mềm và khi có sự chuyển đổi chế độ làm việc nhờ đưa vào thông số thời gian cho nút mạng trong quá trình thiết lập hợp lý sẽ mang lại tiết kiệm năng lượng đáng kể.

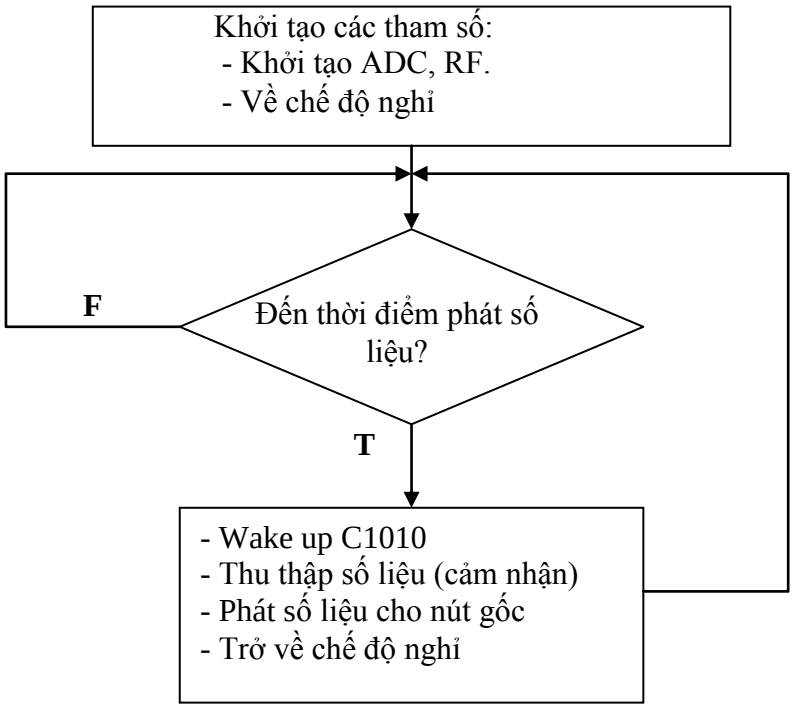
b. Cấu trúc lại chương trình

Cải thiện cấu trúc chương trình là quá trình sửa lại các cấu trúc điều khiển sai trong vòng lặp và đơn giản hóa lại điều kiện Ví dụ: Điều kiện phức tạp: ***if not*** *(A > B* ***and*** *(C < D* ***or not*** *(E > F)))*.... Điều kiện được đơn giản hóa: ***if*** *(A <= B* ***and*** *(C>= D* ***or*** *E > F)*...



Hình 3-10 *Tiến trình kiến trúc lại chương trình*

Trong quá trình tổ chức lại chương trình, hệ thống được chỉnh sửa với hai thuật toán: thuật toán của chương trình thu nhận nhiện độ và thuật toán định tuyến giá tối thiểu. Do vi điều khiển bị ràng buộc về mặt tài nguyên nên đòi hỏi chương trình đưa vào vi điều khiển phải ngắn gọn, tiết kiệm bộ nhớ song vẫn đảm bảo cho việc viết chương trình nhanh, bảo trì và nâng cấp dễ dàng. Thuật toán của chương trình thu nhận nhiệt độ có thể mô tả như sau:



Hình 3-11 *Thuật toán làm việc thu nhận nút mạng cảm nhận*

Ý nghĩa của các bước trong lưu đồ thuật toán:

- ***Bước 1***: Khởi tạo:
  - Khởi tạo ADC:

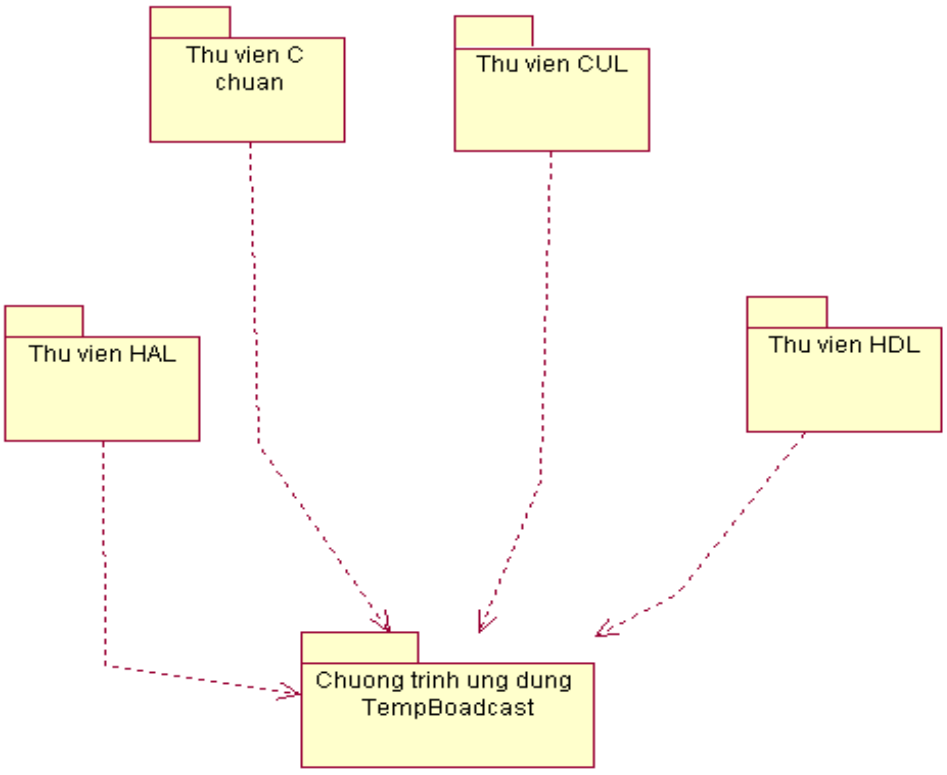
    + Đặt bộ biến đổi ADC về chế độ đơn (10 bit).

    + Đặt điện áp tham chiếu là 1.25V.
  - Khởi tạo RF:
    - Thiết lập một trong các tần số RF: 433MHz, 868MHz, 915MHz.
    - Cách điều chế tín hiệu: mã hoá Manchester
    - Công suất phát: 4 dBm
    - Xác định tốc độ truyền dữ liệu: 2.4kb/s
  - Về chế độ nghỉ:
    - Thiết lập giá trị của bit PCON.IDLE = 01h.
- ***Bước 2:*** Thời điểm phát số liệu là thời điểm xung nhịp của đồng hồ thời gian thực – RTC đã đếm được 15s sau khi nút mạng chuyển về chế độ nghỉ.
  - Nếu True : nút mạng sẽ chuyển sang trạng thái của bước 3.
  - Nếu False : nút mạng quay trở lại trạng thái nghỉ.
- ***Bước 3***:
  - Wake up C1010: Nút mạng thức dậy và chuyển sang chế độ hoạt động, giá trị của bit PCON.IDLE thay đổi.
  - Thu thập số liệu (cảm nhận): Cảm biến sẽ thực hiện chức năng cảm nhận, thu thập thông tin, sau đó đưa tín hiệu ở dạng tương tự về cho vi điều khiển. Tại vi điều khiển, ADC sẽ chuyển tín hiệu sang dạng số rồi đọc giá trị vừa chuyển đổi.
  - Phát số liệu cho nút gốc: Bộ thu phát RF bật TX để thực hiện truyền số liệu cảm nhận được về nút gốc.
  - Trở về chế độ nghỉ: Thiết lập giá trị cho bit PCON.IDLE = 1

3.4.2.3. Modul hóa tiến trình

Tiến trình tổ chức lại một chương trình trong đó các thành phần liên quan được tập hợp lại với nhau thành một modul. Việc kiểm tra và tổ chức lại chương trình thường được thực hiện thủ công. Chương trình của hệ thống cảnh báo nên được cấu trúc lại như sau:

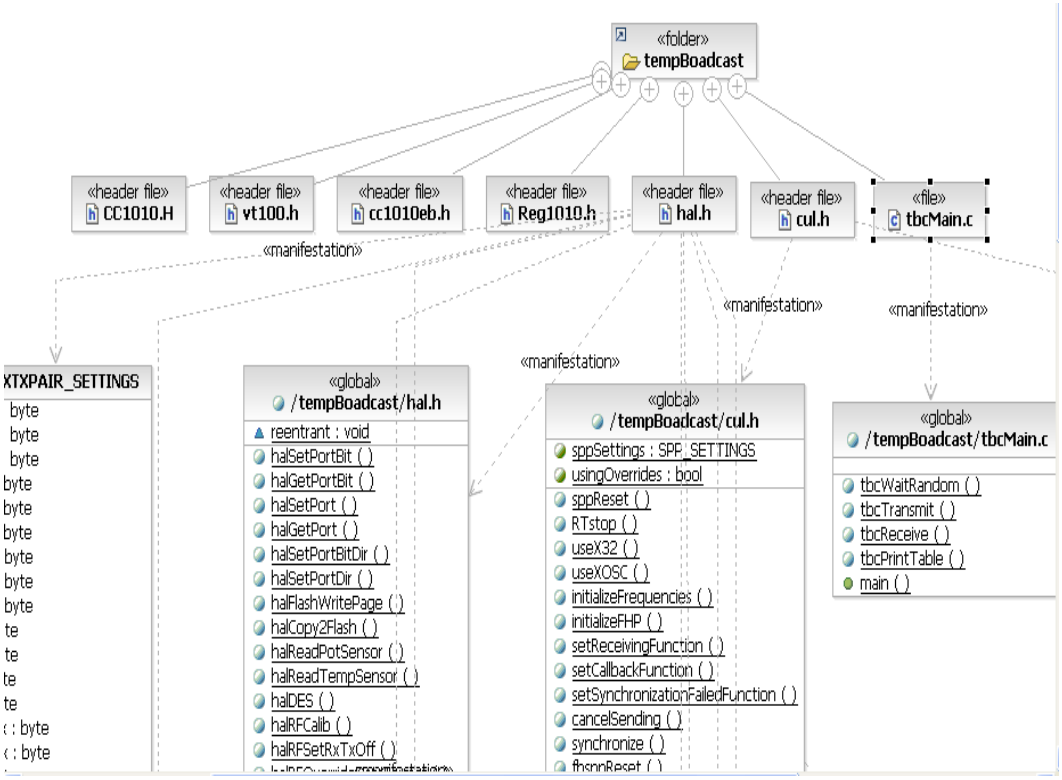
- Các thư viện hệ thống được tập hợp thành Modul systems
- Thư viện chuẩn của C/C++ thành một modul thư viện C/C++
- Tập hợp các hàm có sử dụng việc định nghĩa phần cứng Hardware Definition Files thành modul HDF
- Tập hợp các hàm có sử dụng giao thức truyền nhận Hardware Abstraction Library thành modul HAL
- Tập hợp các hàm có sử dụng vào ra thiết lập thanh ghi cho vi điều khiển CC1010 của Chipcon Utility Library thành modul CUL.



Hình 3-12 *Sơ đồ modul hóa cấu trúc chương trình*

Ngoài ra còn có các thư viện hỗ trợ:

- *CC1010.h*: là thư viện hỗ trợ hệ xử lý vào ra cho chíp hệ vi điều khiển CC1010.
- *vt1010.h*: là thư viện nhằm điều chỉnh, điều khiển các quá trình ta cần thu thập các thông tin, đo đạc, theo dõi sự biến thiên của các biến trạng thái của quá trình.
- *C1010eb.h*: hỗ trợ quy trình nạp chương trình phần mền vào bộ nhớ của vi điều khiển.
- *Reg1010.h*: là thư viện thao tác truy cập với thanh ghi của CC1010.



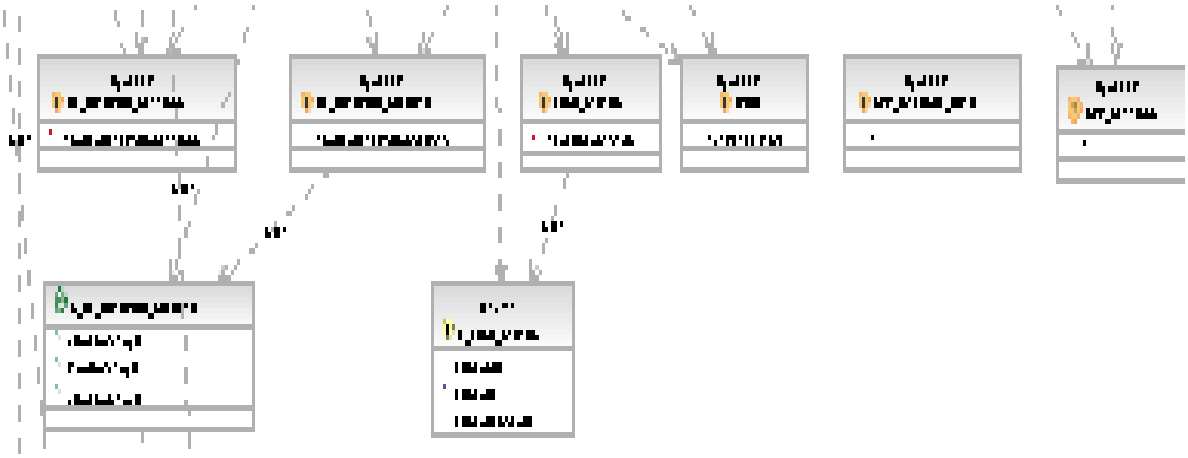
Hình 3-13 *Biểu đồ các lớp thành phần trong chương trình*

Các bước chuyển đổi mô hình sang mã của C/C++ như sau:

1. Từ cửa sổ khung nhìn *Model Explorer view*, bấm chuột phải vào mô hình UML model, rồi chọn *Transform -> Execute Transform -> UML to C/C++*.
2. Trong cửa sổ thực hiện *Run Transform*, chọn *Target page*, click *New Project* để tạo ra một dự án đích cho C++.
3. Tại cửa sổ *New Project wizard*, đặt tên cho dự án *NewTempBoadcast* rồi kích *Finish*.
4. Nhấn vào *Run* trong trang *Run transform*.

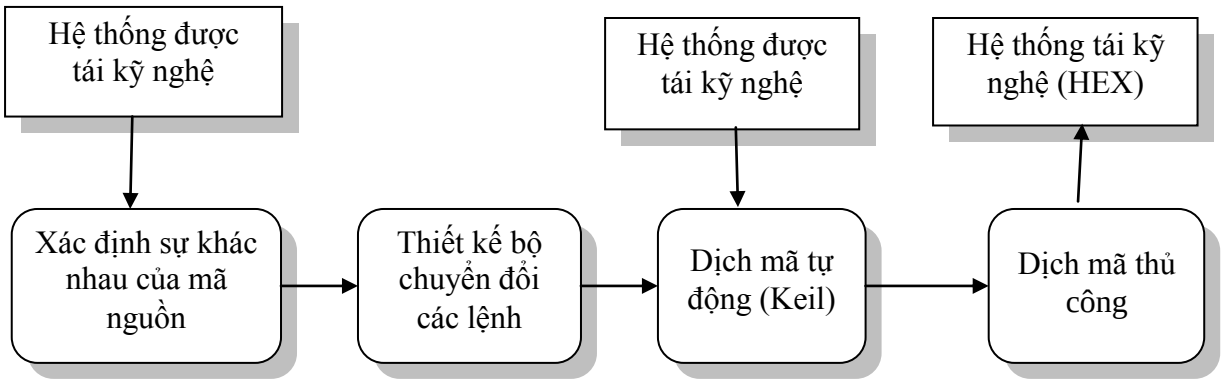
3.4.2.4. Tái kỹ nghệ dữ liệu

Tái kỹ nghệ dữ liệu là phân tích và tổ chức lại cấu trúc dữ liệu trong chương trình, phù hợp với yêu cầu hiện tại của hệ thống, có thể là một phần của quá trình xử lý việc di chuyển dữ liệu từ một hệ thống tới hệ dữ liệu hoặc từ hệ dữ liệu này tới hệ dữ liệu khác. Mục đích là tạo ra môi trường quản trị dữ liệu, phù hợp yêu cầu của hệ thống. Các phương pháp tái kỹ nghệ dữ liệu như: phương pháp dữ liệu sạch, xóa đi những bản ghi trùng lặp, xóa những thông tin thừa và các định dạng không chuẩn, phương pháp mở rộng dữ liệu, xóa giới hạn xử lý, thay đổi độ dài trường hoặc thay đổi giới hạn bảng, vv...



Hình 3-14 *Sơ đồ cấu trúc dữ liệu địa chỉ bộ nhớ và thanh ghi*

3.4.2.5. Tiến trình dịch chương trình



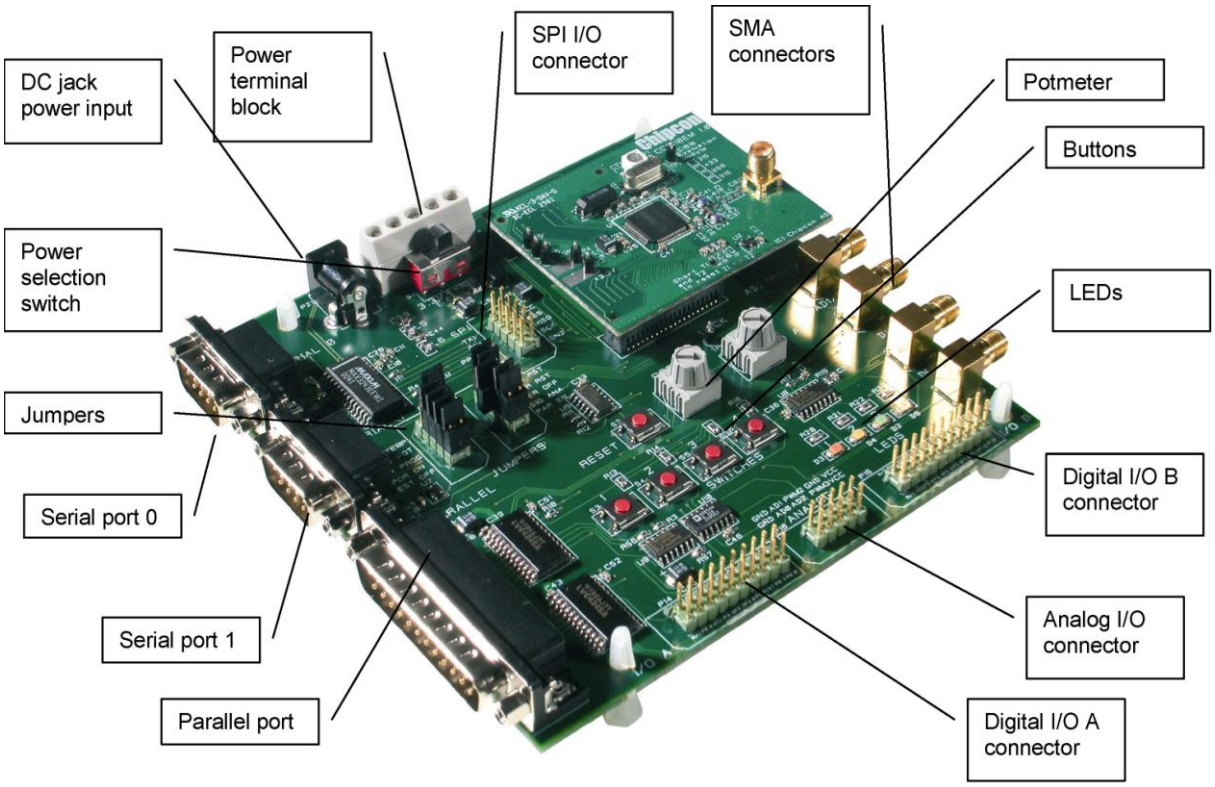
Hình 3-15 *Tiến trình dịch chương trình*

Trong thư mục *NewTempBoadcast* chứa các tệp mã nguồn và header của TempBoadcast đã được thay đổi. Nhưng tệp này có mã C++ mà ta có thể xem, sửa hay thay đổi các giá trị ngay trên mô hình bằng cách nhấn đúp chuột vào từng lớp để sửa mã nguồn. Sau đó sử dụng Keil để dịch dự án này sang tệp HEX rồi thực hiện qua trình nạp lại (Ép flash rom) cho bộ nhớ của CC1010 của từng nút mạng một.

3.5. Quy trình nạp phần mền cho từng nút mạng và vận hành hệ thống

Sau khi phần mềm được thiết kế lại, sử dụng công cụ Keil để dịch dự án này sang tệp HEX rồi thực hiện quá trình nạp lại (Ép flash rom) vào bộ nhớ của CC1010 cho từng nút mạng. Các bước của tiến trình được thực hiện như sau:

- ***Bước 1*:** Nối bản mạch MB với PC. Chương trình nhúng sẽ được nạp cho nút mạng thông qua bản mạch này

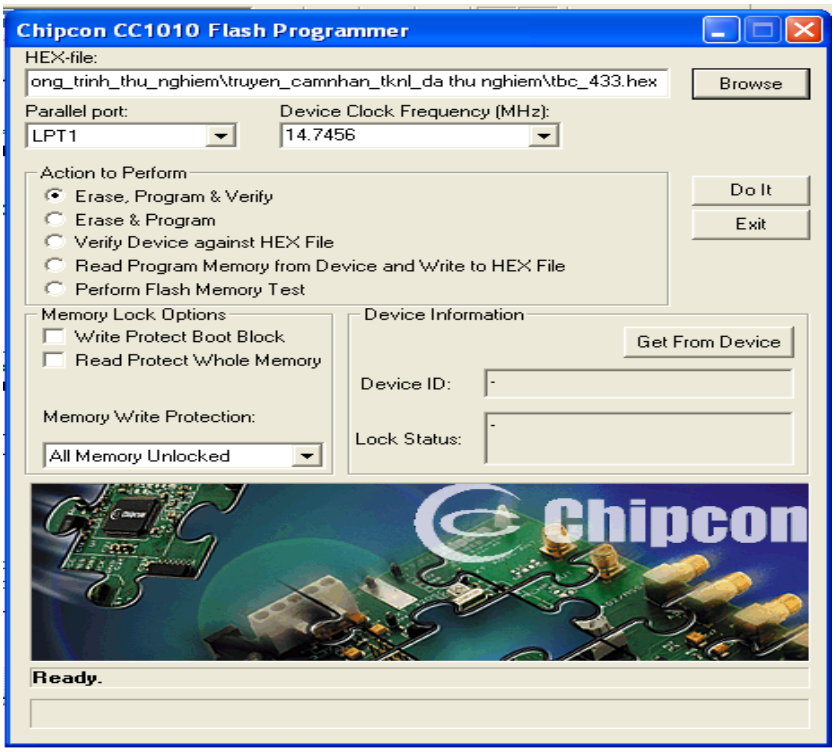


Hình 3-16 *Bo mạch tool để nạp phần mềm cho nút mạng*

- ***Bước 2*:** Gắn nút mạng vào bản mạch đã nối với PC

Hình 3-17 *Cách kết nối vào máy tính*

- ***Bước 3*:** Dùng trình biên dịch, dịch chương trình trên C/C++ sang tệp Hex để sử dụng trình dịch Keil Vision 2.0.
- ***Bước 4*:** Bật nguồn pin của bản mạch gắn nút mạng, mở chương trình Chipcon CC1010 Flash Programmer để nạp tệp *.hex* vừa dịch ở bước 3 cho nút mạng.



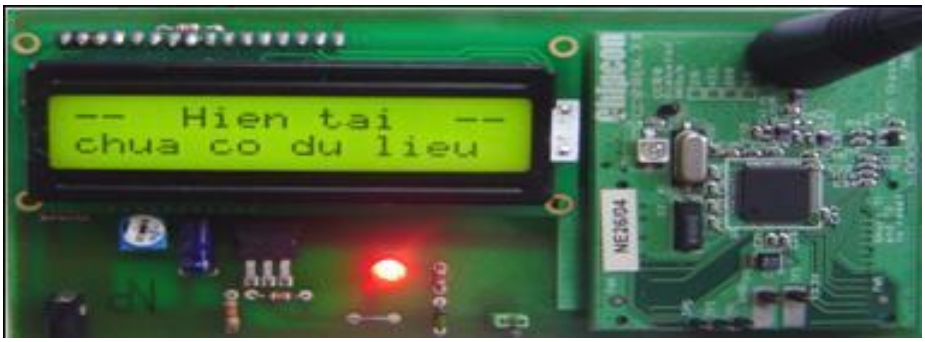
Hình 3-18 *Chương trình nạp phần mềm cho nút mạng*

- ***Bước 5*: Tháo nút mạng ra khỏi bản mạch, gắn với nguồn pin 3.5V rồi lắp đặt lại vị trí cần thiết. Trước khi lắp lại nên tiến hành kiểm tra hoạt động của nút mạng xem có tốt không. Nếu không tốt thì phải nạp lại và kiểm thử.**
- ***Bước 6*: Cuối cùng là vận hành cả hệ thống.**

3.6. Kết quả đạt được và một số đánh giá

Tái kỹ nghệ áp dụng trong quy trình phát triển phần mềm RUP giúp chúng ta tổ chức quản lý toàn bộ tiến trình phát triển phần mềm. Nhờ quá trình tái kỹ nghệ mà các nhà kiến trúc, nhà phân tích, thiết kế và nhà phát triển phần mền dễ dàng sử dụng đồng bộ trao đổi các công đoạn với nhau, tuân thủ theo tiến trình chung, quản lý được các thay đổi, quản trị được các yêu cầu, dễ bảo trì và cải thiện được chất lượng phần mềm.

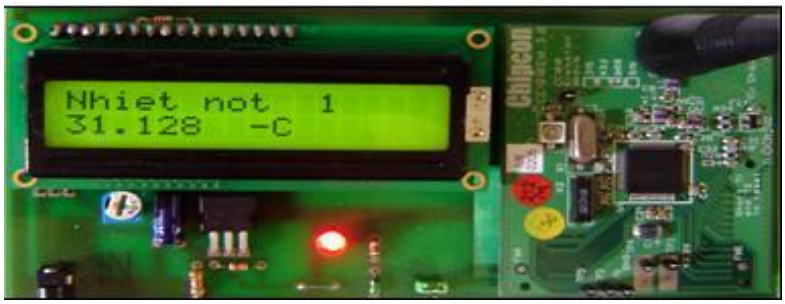
Với hệ thống trên, sau khi vận hành cho kết quả, ta kiểm thử các nút ở các chế độ khác nhau bằng cách gắn thiết bị hiển thị cho nút mạng như hình 4.23, khi chưa bật nút gốc, nghĩa là nút gốc vẫn hoạt động nhưng không thu được tín hiệu của 2 nút mạng. Màn hình sẽ hiển thị như sau:



Hình 3-19 *Kết quả kiểm tra nút số 1 khi nut gốc chưa hoạt động*

3.6.1. Cấp nguồn cho cả nút gốc và các nút mạng

Khi cấp nguồn, nút gốc sẽ thu được thông tin của nút mạng và tiến hành kiểm tra địa chỉ, nếu đúng là địa chỉ cần đọc, nó sẽ cho hiển thị dữ liệu lên LCD như sau:



Hình 3-20 *Truyền đơn bước khi nút gốc hoạt động*

Truyền đa bước nút mạng truyền về nút cơ sở thông qua các nút trung gian

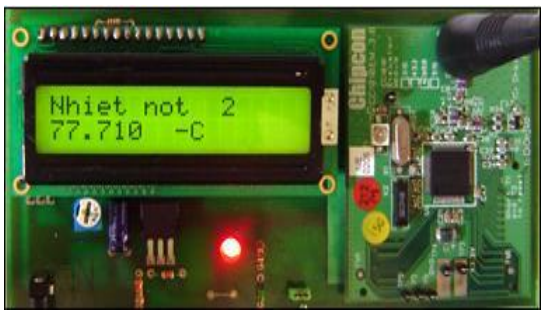
H. a

H. b

H. c

Hình 3-21 *Truyền đa bước mức nhiệt độ an toàn*

H.a



H.b

H. c

Hình* 3-22** ***Nhiệt độ vượt quá mức ngưỡng

3.6.2. Đánh giá kết quả qua các phép đo

Bảng dưới đây cho kết quả đo với chương trình có tiết kiệm năng lượng nhờ chuyển đổi chế độ làm việc, tần số RF là 433MHz, kết quả thu được là:

Lần đo	Dòng điện tiêu thụ (mA)					
	Chế độ nghỉ		Cảm nhận		Truyền	
	Số liệu mới	Số liệu cũ	Số liệu mới	Số liệu cũ	Số liệu mới	Số liệu cũ
1	0.1	9.6	21	23.6	18	18
2	0.2	10	23.6	24	17.8	17.3
3	0.2	10	23.6	24	17.9	18.5
4	0.2	10	23.5	24	17.8	18
5	0.1	10	22.8	24	19	18
Trung bình	0.16±0.048	9.8±0.056	22.9± 0.8	24± 0.8	18.1± 0.36	18.1± 0.36

Bảng 3-23 *Kết quả sau khi vận hành thử nghiệm*

3.6.3. Nhận xét

Từ bảng kết quả trên ta nhận thấy, chương trình đã thực hiện được tiết kiệm năng lượng rất rõ ràng. Dòng điện tiêu thụ tại chế độ nghỉ chỉ bằng khoảng 1% dòng điện tiêu thụ tại chế độ tích cực. Vì vậy, nếu thời gian nút mạng ở trong chế độ nghỉ kéo dài sẽ tiết kiệm năng lượng rất nhiều. Trong chương trình này, thời gian nút mạng nghỉ được lấy là 15s. Tuy nhiên, tuỳ theo ứng dụng thực tế yêu cầu thường xuyên hay định kỳ cung cấp thông tin mà giá trị này có thể tăng lên hoặc giảm đi. Ta có thể nhận thấy, với những mạng chỉ cần cung cấp thông tin một cách định kỳ sẽ tốn ít năng lượng hơn. Căn cứ vào nhu cầu thực tế sử dụng ta có thể can thiệp vào thời gian nút mạng nghỉ để có thể tiết kiệm năng lượng nhất.

Với kiến trúc chương trình mới này đã tiết kiệm tiêu thụ năng lượng nút mạng sẽ thay đổi chế độ liên tục, vì vậy sẽ khó theo dõi kết quả đo. Để có thể thấy rõ hiệu quả

tiết kiệm năng lượng, ta bỏ hàm chuyển đổi chế độ làm việc: SelectClockMode(char iMode) và chức năng truyền dữ liệu về nút gốc, kết quả đo được khi mạng chỉ cảm nhận là:

Tần số RF	Dòng điện tiêu thụ (mA)					
	Lần 1	Lần 2	Lần 3	Lần 4	Lần 5	Trung bình
433MHz	21.2	21	21.1	21.2	21	21.1±0.1
915MHz	23.1	23	23.3	23.1	23	23.1±0.1

Bảng 3-24 *Bảng kết quả thử nghiệm khi thay đổi không có tiết kiệm năng lượng*

Khi tần số truyền nhận tăng lên, dòng điện tiêu thụ cũng lớn hơn. So sánh cột giá trị dòng điện tiêu thụ khi cảm nhận ở bảng 4.3 với bảng 4.4 ta thấy: cùng ở tần số 433MHz nhưng khi có chuyển đổi chế độ làm việc dòng điện tiêu thụ sẽ lớn hơn. Như vậy, rõ ràng giữa các quá trình chuyển đổi chế độ làm việc cũng tiêu hao 1 phần năng lượng. Tuy nhiên, phần năng lượng do nó tiêu hao là không đáng kể so với phần năng lượng mà nó tiết kiệm được. Vì vậy, giải pháp chuyển đổi chế độ làm việc vẫn được coi là giải pháp tiết kiệm năng lượng.

KẾT LUẬN

Đề tài này trình bày các vấn đề về tái kỹ nghệ phần mềm và vận dụng chúng cho việc phát triển phần mềm. Quá trình tái kỹ nghệ phần mềm đề cập tới các tác vụ nhằm tổ chức lại, hay thay đổi lại hệ thống phần mềm để làm cho việc bảo trì chúng dễ dàng hơn, hệ thống hoạt động hiệu quả hơn. Tiến trình tái kỹ nghệ có liên quan đến hoạt động: *dịch mã nguồn, kỹ nghệ đảo ngược, cải thiện cấu trúc chương trình, môdul hóa chương trình và tái kỹ nghệ dữ liệu.*

Để áp dụng quy trình tái kỹ nghệ cho một hệ thống cụ thể, đề tài cũng tìm hiểu quy trình IBM Rational Unified Process dành cho phát triển hệ thống phần mềm hướng đối tượng cùng các công cụ trợ giúp cho tiến trình tái kỹ nghệ và phát triển hệ nhúng.

Đề tài tập trung vào việc ứng dụng quy trình tái kỹ nghệ phần mềm và các công cụ trợ giúp cho hệ thống cảnh báo thiên tai (cảnh báo cháy rừng) với mạng cảm nhận không dây WSN nhằm nâng cao chất lượng và hiệu quả của nó.

Kết quả đạt được của hệ thống mới nhờ tái kỹ nghệ cho thấy: hệ thống hoạt động trở lại tốt, với một số thay đổi như: việc tiêu thụ dòng điện trên nút mạng ở ba chế độ khác nhau rất nhiều, dòng điện tiêu thụ ở chế độ nghỉ chỉ bằng khoảng 1% ở chế độ cảm nhận hoặc chế độ truyền, nghĩa là năng lượng tiêu thụ trong chế độ nghỉ giảm khoảng 100 lần so với năng lượng tiêu thụ trong chế độ tích cực. Đồng thời, khi năng lượng của nút mạng được duy trì lâu sẽ làm cho việc chọn đường nhanh chóng, dễ dàng hơn, tăng tốc độ của mạng. Nhược điểm của quy trình này là phụ thuộc phần lớn vào công cụ hỗ trợ triển khai.

Hướng tiếp theo của đề tài là mở rộng ứng dụng quy trình tái kỹ nghệ và sử dụng lại trên những hệ phần mềm cụ thể khác: hệ thống quản lý thông tin, phần mềm tiện ích, phần mềm hệ thống với các loại dự án nhằm cải tạo các phần mềm cũ để phù hợp với sự thay đổi của yêu cầu hiện tại.

TÀI LIỆU THAM KHẢO

[1]. Nguyễn Văn Vỵ, *Phân tích và thiết kế các hệ thống thông tin hiện đại hướng cấu trúc và hướng đối tượng*, NXB Thống kê, 2002

[2]. Nguyễn Văn Vỵ, *Phân tích và thiết kế các hệ thống thông tin hiện đại hướng cấu trúc và hướng đối tượng*, NXB Thống kê, 2002.

[3]. PSG.TS. Nguyễn Văn Vỵ, *Phân tích thiết kế hệ thống phần mềm theo hướng đối tượng*, Bài giảng, ĐH Công nghệ, 2004.

[4]. PGS.TS Vương Đạo Vy, *Mạng và truyền dữ liệu*, ĐHQG, 2005

[5]. Roger S. Pressman, *Software Engineering a practitioner's Approach,* McGraw-Hill, 860 pp, 2001.

[6]. E. J. Chikofsky and J. H. Cross, *"Reverse Engineering and Design Recovery: A Taxonomy,"* IEEE Software, vol. 7, pp. 13–17, January 1990.

[7]. Chipcon, *CC1010 Datasheet*