ĐẠI HỌC BÁCH KHOA HÀ NỘI

Báo cáo môn học Trí tuệ nhân tạo

Đề tài: Quy trình phát triển phần mềm nhúng

Người thực hiện

Trịnh Hoàng Hà - 20101461
Tạ Văn Trường - 20102404
Nguyễn Duy Thành - 20102737

Nhóm:

Nhóm 40 - K55

Người hướng dẫn

Lương Mạnh Bá

Viện công nghệ thông tin và truyền thông

Bộ môn Công nghệ phần mềm

Ngày 21 tháng 11 năm 2012

Mục lục

Danh sách hình vẽ

Danh sách bảng

Abbreviations

DFD	**D**ata **F**low **D**iagram: Đồ thị luồng dữ liệu
ROOM	**R**eal-time **O**bject **O**riented Modeling: Mô hình hướng đối tượng thời gian thực
UML	**U**nified **M**odeling **L**anguage: Ngôn ngữ mô hình hoá thống nhất

Chương 1

Hệ thống nhúng và phần mềm nhúng

1.1 Hệ thống nhúng

1.1.1 Định nghĩa

Hệ thống nhúng(embedded system) là một thuật ngữ để chỉ một hệ thống có khả năng tự trị được nhúng vào trong một môi trường hay một hệ thống mẹ. Đó là các hệ thống tích hợp cả phần cứng và phần mềm phục vụ các bài toán chuyên dụng trong nhiều lĩnh vực công nghiệp, tự động hóa điều khiển, quan trắc và truyền tin...Đặc điểm của các hệ thống nhúng là hoạt động ổn định và có tính năng tự động hóa cao.

Hệ thống nhúng thường được thiết kế để thực hiện một chức năng chuyên dụng, thường nó có khả năng tự hành và được thiết kế tích hợp vào một hệ thống lớn hơn để thực hiện một chức năng nào đó. Khác với các máy tính đa chức năng, một hệ thống nhúng chỉ thực hiện một hoặc một vài chức năng nhất định, thường đi kèm với những yêu cầu cụ thể và bao gồm một số thiết bị máy móc và phần cứng chuyên dụng mà ta không tìm thấy trong một máy tính đa chức năng nói chung.

Vì hệ thống chỉ được xây dựng cho một số nhiệm vụ nhất định nên cá nhà thiết kế có thể tối ưu hóa nó nhằm giảm thiểu kích thước và chi phí sản xuất. Các hệ thống nhúng thường được sản xuất hàng loạt với số lượng lớn. Hệ thống nhúng rất đa dạng và phong phú về chủng loại. Đó có thể là những thiết bị cầm tay nhỏ gọn như đồng hồ kĩ thuật số, máy chơi nhạc MP3 hoặc những sản phẩm lớn như cột đèn giao thông, bộ kiểm soát trong nhà máy... Xét về độ phức tạp, hệ thống nhúng có thể rất đơn giản với một vi điều khiển hoặc rất phức tạp với nhiều đơn vị, các thiết bị ngoại vi và mạng lưới được nằm gọn trong một lớp vỏ máy lớn. Các thiết bị như máy tính cầm tay cũng có một số đặc điểm tương tự với hệ thống nhúng. Hệ thống nhúng bao gồm các thiết bị phần cứng và phần mềm, hầu hết đều phải thỏa mãn yêu cầu hoạt động theo thời gian thực(

real- time). Tùy theo tính chất và yêu cầu, mức độ đáp ứng của hệ thống có thể rất nhanh hoặc có thể chấp nhận một mức độ chậm chễ nhất định.

1.1.2 Độ tin cậy của hệ thống nhúng

Các hệ thống nhúng thường nằm trong các cỗ máy được kì vọng là sẽ chạy hàng năm trời liên tục mà không gặp lỗi hoặc có thể khôi phục khi gặp lỗi. Vì thế các phần mềm và hệ thống nhúng được phát triển và kiểm thử một cách cẩn thận hơn là phần mềm cho máy tính cá nhân.

Ngoài ra, các thiết bị rời không đáng tin cậy như ổ đĩa, công tắc hoặc nút bấm thường được hạn chế sử dụng.

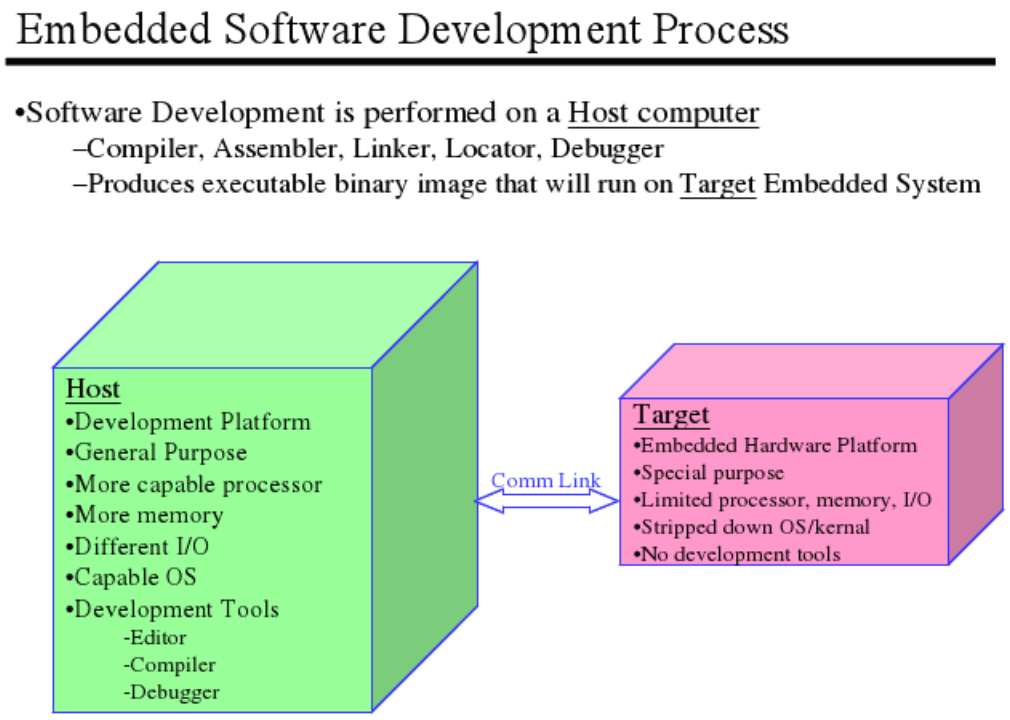
Một số vấn đề cụ thể về độ tin cậy như:

- Hệ thống không thể ngừng để sửa chữa một cách an toàn như các hệ thống không gian, hệ thống dây cáp dưới biển, các đèn hiệu dẫn đường...Giải pháp đưa ra là chuyển sang sử dụng các hệ thống con dự trữ hoặc các phần mềm cung cấp một phần chức năng.
- Hệ thống phải được chạy liên tục vì tính an toàn, ví dụ như các thiết bị dẫn đường máy bay, thiết bị kiểm soát độ an toàn trong các nhà máy hóa chất,... Giải pháp đưa ra là lựa chọn backup hệ thống.
- Nếu hệ thống ngừng hoạt động sẽ gây tổn thất rất nhiều tiền của ví dụ như các dịch vụ buôn bán tự động, hệ thống chuyển tiền, hệ thống kiểm soát trong các nhà máy...

1.2 Phần mềm nhúng

Là phần mềm trong các hệ thống nhúng. Phần mềm nhúng có thể là những chương trình đơn giản chạy trực tiếp trên nền phần cứng hoặc là những chương trình, ứng dụng chạy trên nền một hệ điều hành nhúng. Phần mềm nhúng thường chạy với số tài nguyên phần cứng hạn chế: không có bàn phím, màn hình hoặc có nhưng với kích thước nhỏ, bộ nhớ hạn chế.

Phần mềm nhúng thường được lập trình trên máy tính cá nhân của lập trình viên, được biên dịch với một trình biên dịch và một môi trường phát triển, máy tính dùng để lập trình được gọi là host. Sau đó chương trình được nạp lên thiết bị và chạy, thiết bị mà chương trình được nạp lên gọi la target.Với mỗi target khác nhau sẽ có cấu trúc vi điểu khiển khác nhau, và sử dụng hệ điều hành nhúng khác nhau, do vậy tùy từng loại sẽ có các cách thức lập trình tương ứng.



HÌNH 1.1: Host và target trong phát triển phần mềm nhúng

C là một trong những ngôn ngữ lập trình nhúng phổ biến nhất hiện nay. C có một số ưu điểm nổi bật tiêu biểu như khá nhỏ và dễ dàng cho việc học, các chương trình biên dịch C thường khá sẵn cho hầu hết các bộ xử lý đang sử dụng hiện nay, và có rất nhiều người đã biết và làm chủ được ngôn ngữ này.

So với phần mềm thông thường, phần mềm nhúng có những điểm khác biệt sau:

- Phần mềm và phần cứng được phát triển cùng một lúc,
- Máy tính dùng để phát triển phần mềm (host) thường xuyên khác với target,
- Phần cứng cho target thường xuyên không có sẵn cho đến gần kết thúc dự án,
- Thương không rõ một lỗi nào đó là lỗi phần cứng hay lỗi phần mềm,
- Lỗi phần cứng thường được “chữa” bằng phần mềm,

- Số tiền bỏ ra cho phát triển phần mềm gấp hàng trăm lần so với phần cứng, do đó phần mềm thường được yêu cầu làm những việc mà nó nên được thực hiện bởi phần cứng,
- Có thể có ràng buộc về thời gian thực, xử lý song song, và vấn đề an toàn,
- Không có giao diện người-máy truyền thống, vì vậy các hoạt động của máy tính được giấu khỏi người dùng
- Thường có các ràng buộc về tài nguyên (bộ nhớ) hay mức tiêu thụ năng lượng.

Với những đặc điểm này, việc phát triển phần mềm nhúng trở nên khó khăn hơn.

Chương 2

Quy trình phát triển phần mềm

2.1 Định nghĩa

Toàn bộ quy trình quản lý phát triển phần mềm gắn với khái niệm vòng đời phần mềm, được mô hình hóa với những kỹ thuật và phương pháp luận trở thành các chủ đề khác nhau trong Công nghệ phần mềm.

Vòng đời của phần mềm là thời kì tính từ khi phần mêm được bắt đầu sinh ra cho đến khi chết đi (từ lúc hình thành đáp ứng yêu cầu, vận hành, bảo dưỡng cho đến khi loại bỏ không dùng nữa),

Chuẩn ISO/IEC 12207:2008 là một chuẩn quốc tế về quy trình phát triển phần mềm, định nghĩa tất cả các công việc cần thiết cho việc phát triển và bảo trì phần mềm [1].

Chuẩn ISO/IEC 12207:2008 quy định 43 quá trình (process), trong đó có các pha chính là:

- phân tích/xác định yêu cầu người dùng,
- thiết kế,
- mã hoá,
- kiểm thử,
- bảo trì.

Mô hình phát triển phần mềm sắp xếp các quá trình phát triển phần mềm theo một cách nào đó. Có nhiều mô hình phát triển phần mềm khác nhau, mỗi cái đều có ưu và nhược điểm riêng. Có thể kể đến:

- Mô hình thác nước (Waterfall model),
- Mô hình xoắn ốc (Spiral model),
- Mô hình lập trình linh hoạt (agile development),
- Mô hình chế thử (prototype model),
- ...

2.2 Các quá trình trong phát triển phần mềm

2.2.1 Phân tích - xác định yêu cầu

2.2.2 Thiết kế

2.3 Một số mô hình phát triển phần mềm

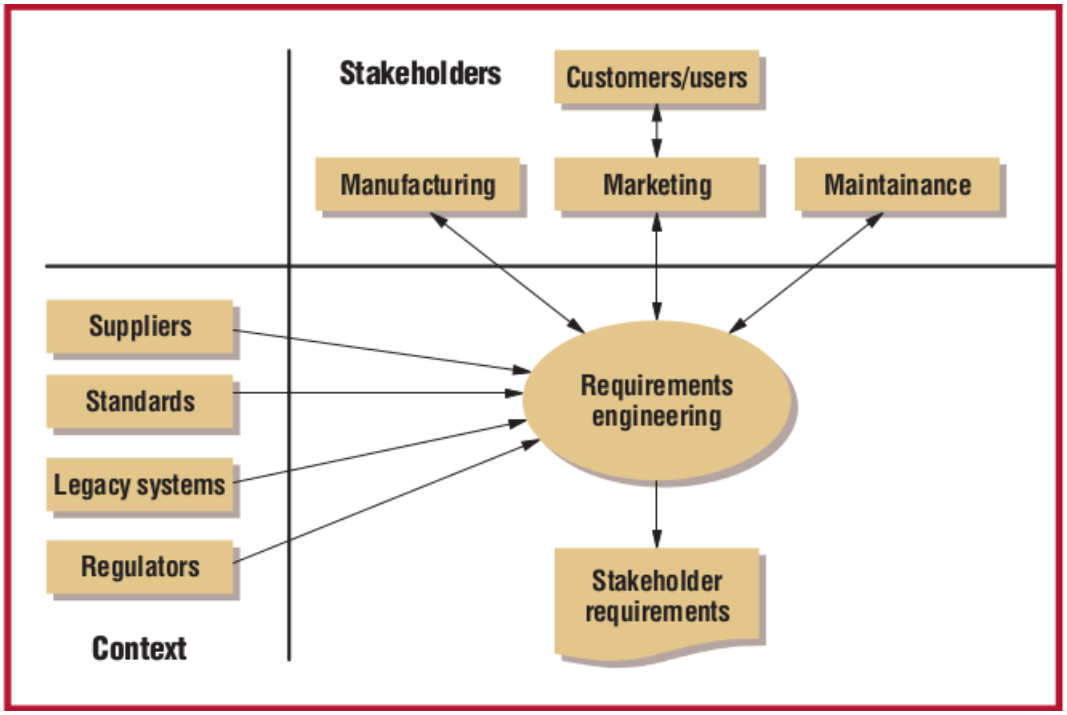
2.3.1 Mô hình thác nước

Chương 3

Quy trình phát triển phần mềm nhúng

3.1 Xác định yêu cầu người dùng

Việc phát triển một hệ thống nhúng thường có nhiều người yêu cầu, bao gồm: nhà sản xuất phần cứng, người bảo trì, marketing, khách hàng (hình 3.1)[2].



HÌNH 3.1: Các bên yêu cầu (stakeholder) trong một dự án hệ thống nhúng

Bước đầu, khách hàng xác định các yêu cầu chức năng (functional) và các yêu cầu phi chức năng (non-functional). Tuỳ thuộc vào loại sản phẩm mà khách hàng sẽ đàm phán qua bộ phận marketing hay trực tiếp với các nhà phát triển.

Kết quả của bước này sẽ là một bản đặc tả các yêu cầu, mô tả về hệ thống sao cho tất cả các bên yêu cầu đều có thể hiểu được. Tài liệu này được coi như hợp đồng với nhà phát triển (phần mềm nhúng).

Đặc tả yêu cầu Với phần mềm nhúng, các đặc tả phi chức năng (non-functional specification) rất quan trọng (do các ràng buộc về tài nguyên hạn chế, đáp ứng ở thời gian thực, mức tiêu thụ năng lượng, ...). Mà đặc tả phi chức năng thường rất khó phát biểu chính xác, do đó không thể dùng các phương pháp đặc tả yêu cầu như bình thường (như sử dụng các mẫu (template) của các trình soạn thảo văn bản).

Với những dự án có sử dụng các biểu đồ để đặc tả, các biểu đồ này hầu hết ở dạng tự do hoặc các biểu đồ tương tự với UML, DFD (Dataflow diagrams - biểu đồ luồng dữ liệu), ... Do không có một sự thống nhất về cú pháp của các biểu đồ, các thành viên dự án có thể hiểu sai ý nghĩa của nó.

Đặc tả hình thức (formal specification) thường ít được sử dụng do tính phức tạp của hệ thống nhúng cũng như các yêu cầu của nó. Việc sử dụng đặc tả hình thức sẽ làm sự hợp tác giữa các thành viên trong dự án trở nên khó khăn hơn vì hầu hết các thành viên đều không hoàn toàn hiểu nó. Với khách hàng thì vấn đề này còn khó khăn hơn nữa.

3.2 Thiết kế

3.2.1 Thiết kế hệ thống

Ở bước này, tổ chức tổng quát của chương trình sẽ được phát triển, bao gồm:

- Task: các tác vụ,
- Comumunications: sự liên lạc giữa các tác vụ với nhau cũng như giữa hệt thống và thể giới bên ngoài,
- Timing: sự phân bố thời gian trên toàn hệ thống, đảm bảo mỗi tác vụ hoạt động trong đúng khoảng thời gian để hoàn thành chức năng của nó,
- Priority: độ ưu tiên, để nhường quyền thực hiện cho một số tác vụ nào đó được coi là quan trọng cho hệ thống vào một thời điểm bất kỳ.

Đây là những yêu cầu cơ bản cho ứng dụng đa tác vụ.

Để bắt đầu thiết kế mức hệ thống, người thiết kế cần phải hiểu rõ chương trình cần phải hoàn thành những gì. Hay phải hiểu rõ được các yêu cầu của phần mềm.

3.2.1.1 Xác định các tác vụ

Đây là quá trình hợp các yêu cầu chức năng của phần mềm lại và hợp thành một số ít các tác vụ. Mỗi tác vụ sẽ là một module có thể được thực hiện riêng biệt, có Comumunications, Timing, Priority riêng của nó. Do đó, các chức năng bên trong module phải tương thích với nhau, hay ít nhất là có thể thực hiện hiện mà không ảnh hưởng đến chức năng khác.

Các chức năng được coi là tương thích với nhau nếu thoả mãn một trong các tiêu chí sau [3]:

- Thực hiện tuần tự,
- Thực hiện đồng bộ với nhau,
- Cùng quản lý một thiết bị ngoại vì,
- Quản lý dữ liệu chung,
- Loại trừ lẫn nhau trong lúc thực hiện,
- Đều là sự mở rộng của một chức năng khác.

Các chức năng thoả mãn một trong các tiêu chí sau đây thì được gọi là không tương thích với nhau. Không nên cho các chức năng này vào cùng một module.

- Thực hiện không đồng bộ,
- Thực hiện ở tốc độ (rate) khác nhau,
- Có mức độ ưu tiên khác nhau

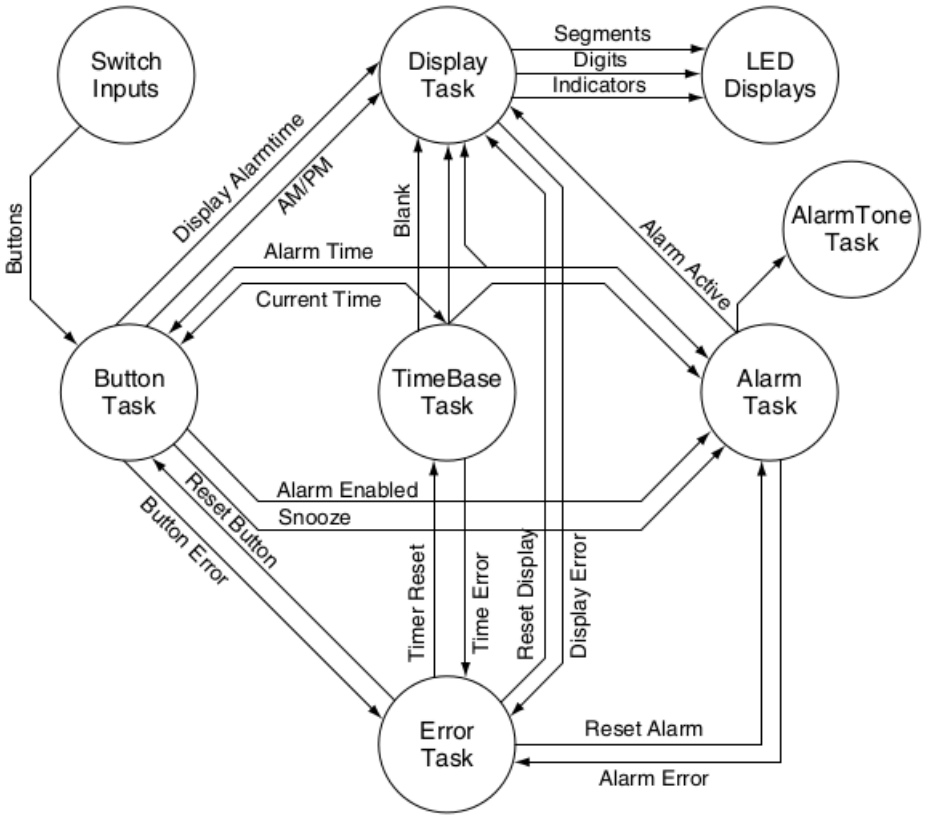
3.2.1.2 Giao tiếp

Bước tiếp theo của thiết kế hệ thống là xác định các giao tiếp giữa các tác vụ với nhau và với thế giới bên ngoài (thông qua các thiết bị ngoại vi) của hệ thống. Qua bước này, chúng ta có thể:

1. ước lượng được bộ nhớ cần thiết cho hệ thống,

2. Xác định tất cả các biến trong chương trình mà không thuộc vào một tác vụ cụ thể nào,

Người ta thường sử dụng biểu đồ luồng dữ liệu (DFD) để diễn tác các giao tiếp.



HÌNH 3.2: Biểu đồ luồng dữ liệu cho đồng hồ báo thức

Hình 3.2 là sơ đồ luồng dữ liệu biểu diễn giao tiếp giữa các tác vụ của một đồng hồ báo thức. Trong đó hình tròn biểu diễn cho một tác vụ, các đường thẳng chỉ ra giao tiếp giữa chúng.

3.2.2 Thiết kế chương trình

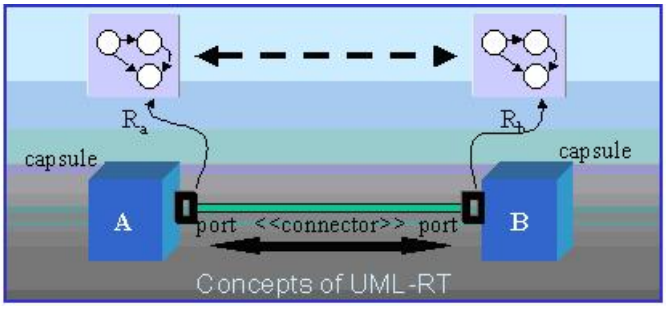
Thiết kế hướng đối tượng / hướng thủ tục Các phương pháp thiết kế này sử dụng các `thủ tục` (đối với thiết kế hướng đối tượng là `phương thức`) là các thành phần cơ bản của chương trình. Các thủ tục (phương thức) này nhận các tham số, thực hiện một số hữu hạn các phép tính và trả về giá trị.

Phương pháp hướng đối tượng hợp nhất các thủ tục và dữ liệu để tạo thành các `đối tượng (object)`. Các đối tượng này là thụ động, bởi vì nó cần sự tác động của bên ngoài để thực hiện các phương thức của nó.

Tuy nhiên thế giới thực là chủ động, giống với các tiến trình hơn là các đối tượng, nó hoạt động trên các luồng riêng, tương tác với các tiến trình khác bằng các thông điệp.

Như vậy, mặc dù thiết kế hướng đối tượng được sử dụng rất nhiều trong việc xây dựng các hệ thống phần mềm lớn, nó lại không thích hợp cho việc giải quyết các vấn đề của việc thiết kế phần mềm nhúng.

Người ta đã mở rộng mô hình thiết kế hướng đối tượng thành `mô hình hướng đối tượng thời gian thực (Realtime Object-Oriented Model - ROOM)` [4]. ROOM hỗ trợ UML-RT, một mở rộng của UML chuẩn dành cho các hệ thống thời gian thực.



HÌNH 3.3: UML-RT

Hình 3.3 minh hoạ cho biểu đồ UML-RT. Trong đó A, B là các `capsule` - đối tượng tích cực. Các capsule đều có các `port` để kết nối với nhau. Các capsule sẽ truyền thông điệp qua các port này.

3.3 Mã hoá chương trình

Ngôn ngữ lập trình Do thường gặp hạn chế về tài nguyên cũng như yêu cầu về hiệu năng, C và Assembly là hai ngôn ngữ thường được sử dụng trong lập trình phần mềm nhúng. Chương trình viết bằng 2 ngôn ngữ này thường nhỏ, nhanh và có khả năng thao tác trực tiếp với phần cứng. Trong đó Assembly thường được dùng cho những nơi cần yêu cầu cao về hiệu năng, kích thước chương trình hay là phụ thuộc phần cứng (VD: driver).

Mặc dù ngôn ngữ C không trực tiếp hỗ trợ lập trình hướng đối tượng nhưng vẫn có thể sử dụng các kỹ thuật hướng đối tượng bằng các cấu trúc có sẵn của C:

- Sử dụng struct thay cho class,
- Sử dụng hàm với đối số đầu kiểu của struct thay cho phương thức,
- Sử dụng con trỏ void để có được khả năng đa hình,
- ...

Ngoài ra, cũng có thể dùng C++ cho phát triển phần mềm nhúng. C++ kế thừa được những ưu điểm của C (mềm dẻo, hiệu năng) cộng thểm nhiều tính năng khác làm cho việc lập trình đơn giản hơn (template, STL, ...). Tuy vậy cần chú ý đến kích thước chương trình khi sử dụng các tính năng này. Ngoài ra một số tính năng như exception cũng có thể không được hỗ trợ trên một số hệ thống.

Một số vấn đề khi mã hoá chương trình phần mềm nhúng

Sự phụ thuộc nền tảng Một chương trình có thể chạy đúng với phần cứng này, HDH này nhưng chưa chắc đã chạy đúng trên phần cứng, HDH khác. Đó là do một số yếu tố trong chương trình khác nhau ở những nền tảng khác nhau. Ví dụ:

- Kích thước kiểu dữ liệu nguyên (`int, long, short`, các kiểu con trỏ): ngôn ngữ C/C++ không quy định về kích thước chính xác của các kiểu dữ liệu này. VD: HDH 32bit thường quy định kích thước con trỏ là 32bit, HDH 64bit thường quy định kích thước con trỏ là 64bit. Do vậy cần chú ý khi sử dụng các kiểu dữ liệu này, không nên mặc nhiên coi rằng một kiểu dữ liệu nào đó có một kích thước xác định. Nếu muốn kiểu nguyên với kích thước cụ thể thì sử dụng các kiểu có sẵn như `int32_t, uint16_t, ....`
- Sự căn chỉnh byte (byte alignment) và thứ tự byte (byte order): Với mỗi hệ thống khác nhau, yêu cầu về byte alignment và byte order có thể khác nhau. VD: trên MIPS, truy cập vào một biến nguyên 32bit có địa chỉ không chia hết cho 4 sẽ sinh ra lỗi SIGBUS.
- Tính toán số thực dấu phẩy động: nhiều vi xử lý hiện nay không hỗ trợ tính toán trên dấu phẩy động. Khi đưa chương trình lên nền tảng này cần cân nhắc việc cài đặt tính năng này bằng phần mềm hoặc sử dụng một thuật toán khác không sử dụng đến tính toán số thực dấu phẩy động.

Hạn chế về tài nguyên Đây luôn là vấn đề xuyên suốt cả quá trình phát triển phần mềm nhúng. Trong gian đoạn mã hoá chương trình này, cần tuân thủ chặt chẽ các nguyên tắc để sự dụng tiết kiệm tài nguyên nhất có thể mà vẫn đảm bảo được hiệu năng.

Công cụ

Trình biên dịch (compiler) Trình biên dịch được sử dụng trong lập trình nhúng là cross compiler, tức là trình dịch chạy trên host nhưng lại sinh mà cho target. Với mỗi target khác nhau sẽ có một cross compiler riêng, thường do các nhà sản xuất vi xử lý cung cấp. Ngoài ra còn có những trình biên dịch có thể sinh mã cho nhiều target khác nhau, như GCC hỗ trợ hơn 20 loại vi xử lý [5].

Trình dịch hợp ngữ (Assembler) Chạy trên host. Có nhiệm vụ dịch mã hợp ngữ được sinh ra bởi compiler thành mã máy.

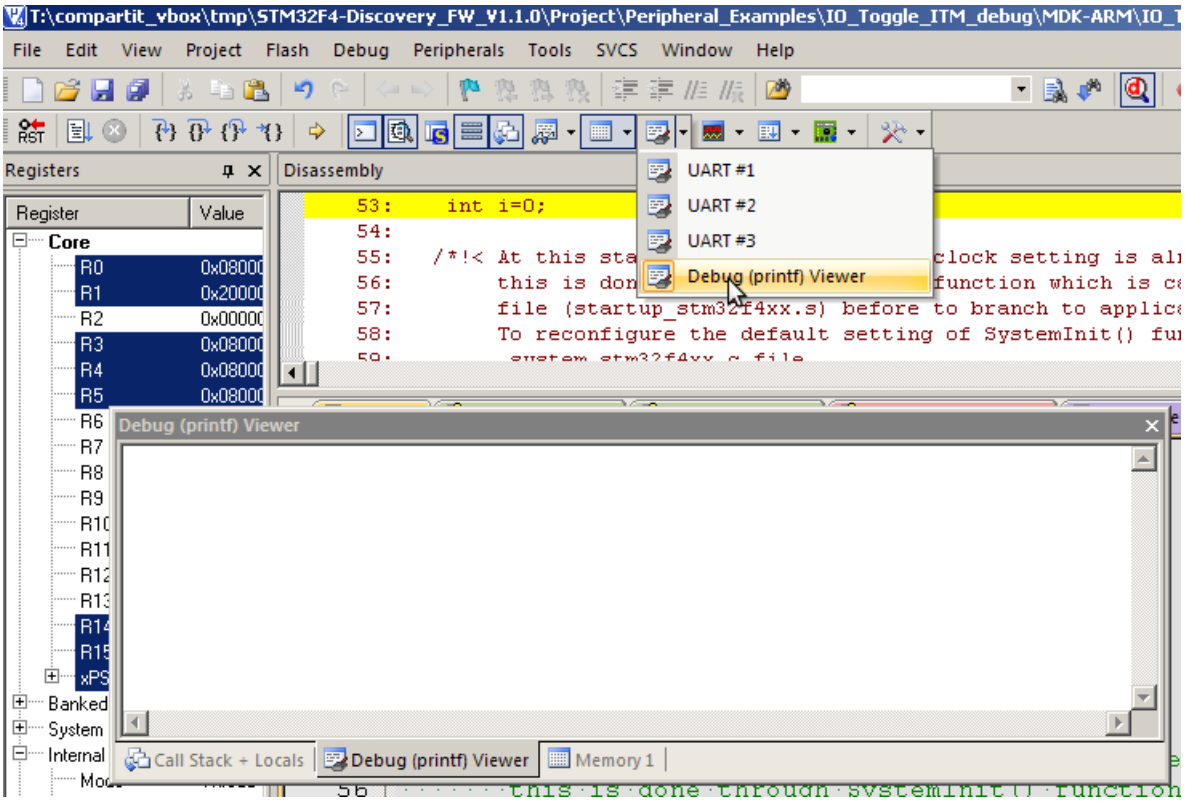
Trình liên kết (Linker) Liên kết các file mã máy (nhị phân) đã được biên dịch cùng với các thư viện cần thiết thành chương trình hoàn chỉnh. Cũng chạy trên host.

Chương trình giả lập (emulator) Giả lập target trên host. Dùng để debug hoặc test chương trình mà không cần phải sử dụng phần cứng thật (target). Emulator thường sử dụng phương pháp thông dịch mã nhị phân của target thành mã nhị phân dành cho máy host để máy host có thể hiểu và thực thi được chương trình. Do đó chương trình chạy trên emulator thường chậm hơn so với máy thực.

HÌNH 3.4: Giả lập ARM trên PC x86

3.3.1 Môi trường phát triển tích hợp

Chương trình hợp nhất tất cả các công cụ phát triển (compiler, assembler, linker, emulator, . . .).



HÌNH 3.5: Keil - IDE phát triển phần mềm trên các vi xử lý họ ARM

3.4 Kiểm thử

Kiểm thử phần mềm nhúng có nhiều điểm giống nhau so với kiểm thử phần mềm ứng dụng thông thường. Tuy nhiên, có một vài điểm khác nhau giữa hai loại kiểm thử. Kiểm thử phần mềm nhúng thường phải truy cập đến những công cụ kiểm thử dựa trên phần cứng, cái thường không được sự dụng trong phát triển phần mềm ứng dụng.

Kiểm thử cho điện thoại di động đương nhiên sẽ khác so với kiểm thử một bộ phận điều khiển của xe hơi. Mỗi công việc đều đòi hỏi định lượng pháp kiểm thử cho từng hệ thống riêng biệt. Không có một phương pháp chung nào cho tất cả các hệ thống nhúng.

3.4.1 Các giai đoạn kiểm thử

1. Kiểm thử module
2. Kiểm thử tích hợp

3. Kiểm thử hệ thống (phần mềm)

4. Kiểm thử tích hợp phần cứng - phần mềm

Như vậy, so với kiểm thử thông thường thì kiểm thử phần mềm nhúng có thêm giai đoạn kiểm thử tích hợp phần cứng - phần mềm.

3.4.2 Một số vấn đề trong kiểm thử phần mềm nhúng

- Kiểm thử các hệ thống thời gian thực khó bởi vì chúng phản ứng với các sự kiện và dữ liệu không đồng bộ. Trên thực tế gần như không thể kiểm tra một cách toàn diện các trường hợp đầu vào của một hệ thống nhúng.
- Không thể sử dụng các phương pháp kiểm thử thông thường vào một số hệ thống (vì dụ: không thể đặt breakpoint trong quá trình thực thi hệ thống điều khiển máy bay khi nó đang ở độ cao hàng chục km!). Nếu những hệ thống này hoạt động sai, gần như không thể lặp lại chuỗi sự kiện dẫn đến lỗi để tìm cách chẩn đoán.
- Nhiều hệ thống phụ thuộc vào trạng thái, có nghĩa là phản ứng đối với một sự kiện không chỉ phụ thuộc vào sự kiện đó mà còn phụ thuộc vào những gì đã xảy ra trước đó. Nói cách khác, hai sự kiện giống nhau có thể dẫn đến kết quả khác nhau.

3.5 Bảo trì

Về cơ bản, bảo trì phần mềm nhúng không khác với bảo trì phần mềm thông thường. Chúng đều có các bước chính như nhau. Tuy nhiên, do các đặc tính của phần mềm nhúng mà có một số yếu tố phát sinh [6].

Một số vấn đề trong bảo trì phần mềm nhúng

- Các yêu cầu không ổn định: Với phần mềm nhúng, các yêu cầu được thêm vào và thay đổi thường xuyên. Sự thay đổi thường không phải lúc nào cũng có thể thông báo được đến tất cả các bên. Do đó việc bảo trì trở nên rất phức tạp.
- Sự thay đổi về công nghệ: Công nghệ phần cứng thường phát triển nhanh hơn phần mềm.

- Cần phải đào tạo: Với một vài lĩnh vực, quá trình bảo trì phức tạp bởi rất khó để hiểu được chương trình phần mềm.
- Môi trường giả lập vs thiết bị thực (target): Thông thường, những người phát triển phần mềm không phải ai cũng có quyền tiếp xúc với thiết bị thực, việc phát triển / bảo trì phần mềm trở nên phụ thuộc vào môi trường giả lập.
- Ràng buộc về phần cứng

Tài liệu tham khảo

[1] Wikipedia. Iso/iec 12207, 2012. Available at http://en.wikipedia.org/wiki/ISO/IEC_12207.

[2] Bas Graaf, Marco Lormans, and Hans Toetenel. Embedded software engineering: The state of the practice. *IEEE*, 2003.

[3] Keith E. Curtis. Criteria for compatible software functions. In *Embedded System: World class Design*. Newnes, 2007.

[4] Wikipedia. Objectime, 2012. Available at http://en.wikipedia.org/wiki/ObjecTime.

[5] Wikipedia. Gnu compiler collection, 2012. Available at http://en.wikipedia.org/wiki/GNU_Compiler_Collection#Architectures.

[6] Mikael Lindvall, Seija Komi-Sirviö, Patricia Costa, and Carolyn Seaman. Embedded software maintenance. Technical report, Fraunhofer Center for Experimental Software Engineering Maryland and The University of Maryland, 2003.