TẠP CHÍ KHOA HỌC TÍNH TOÁN VÀ ĐIỀU KHIỂN TẬP IV (1988), SỐ 3 + 4 (19-26)

Trao đổi

CÔNG NGHỆ PHẦN MỀM VÀ LẬP TRÌNH QUÁ KHỨ - HIỆN TẠI - TƯƠNG LAI

GS Dines Bjorner
Trường Đại học Kỹ thuật Đan mạch

Lời tòa soạn

Giáo sư Dines Bjorner là một chuyên gia lớn trong lĩnh vực công nghệ phần mềm và lập trình. Bài nói chuyện của ông với giới tin học Nhật Bản giúp chúng ta hiểu được những nội dung và các kỹ thuật cơ bản trong hai lĩnh vực rộng lớn và rất quan trọng của tin học: Công nghệ phần mềm và lập trình.

Bài nói chuyện này có thể xem là một tài liệu tham khảo rất tốt cho những ai có ý định tìm hiểu sâu sắc về hai lĩnh vực nói trên.

Mở đầu

Rất cám ơn các bạn đã cho tôi cơ hội gặp gỡ và trình bày một vài ý tưởng hiện hành về công nghệ phần mềm (CNPM) và lập trình (LT). Những gì tôi sẽ nói với các bạn hôm nay là kết quả của nhiều năm làm việc, suy nghĩ và bàn luận. Nhiều người đã giúp tôi trong việc trình bày chính xác những ý tưởng của tôi. Tôi đặc biệt cám ơn các bạn đồng nghiệp của tôi trong các nhóm nghiên cứu 2.2 và 2.3 của IFIP.

Vì tôi tin vào đặc tả hình thức của phần mềm, tốt hơn là tôi cho các bạn đặc tả bài nói của tôi. Về cơ bản sẽ là bốn phận: (1) Trước hết là một phần mở đầu chung (các mục 1,2,3) trong đó tôi trình bày sự khác biệt giữa một bên là công nghệ thông tin và một bên là tin học xem như một quan niệm rộng hơn. (2) Tiếp đó (trong mục 4) tôi thu hẹp vấn đề vào việc trình bày sự khác biệt giữa CNPM và LT. (3) Sau một vài nhận xét về sự khác biệt đó, tôi sẽ đi vào một vấn đề ít nhiều kỹ thuật hay ít nhiều có bàn luận mang tính kỹ thuật (mục 5). (4) Trên cơ sở của ba điều đó, sự khác biệt giữa công nghệ thông tin và tin học, sự khác biệt giữa CNPM và LT và một thí dụ ít nhiều mang tính kỹ thuật, tôi sẽ nêu lên một số kết luận (mục 6).

1. Một vài định nghĩa

Một trong những vấn đề của ngành nghề non trẻ của chúng ta - có lẽ mới có khoảng 40 năm - là chưa thiết lập được thuật ngữ. Vì vậy tôi muốn đưa ra một số định nghĩa.

Trước hết, để tiện lập luận, tôi muốn nêu lên sự khác biệt giữa khoa học máy tính (computer science), khoa học tính toán (computing science), và công nghệ phần mềm (software engineering).

Theo tôi, khoa học máy tính là sự nghiên cứu các chương trình, còn khoa học tính toán là sự nghiên cứu lập trình. Dương nhiên cùng một người thường có cả hai năng lực, nhưng tôi tin rằng có một đường ranh giới. Trong việc nghiên cứu chương trình, ta nghiên cứu cái gì có thể tính được. Ta nghiên cứu để tính một cái gì đó có thể có khó khăn thế nào, và như vậy ta nghiên cứu các tính chất của chương trình. Trong việc nghiên cứu về lập trình, ta nghiên cứu xây dựng các chương trình như thế nào. Và bây giờ, CNPM là sự thực hành của chương trình và lập trình.

Bây giờ tôi muốn nói về sự khác biệt giữa người lập trình và người kỹ sư phần mềm. Người lập trình là người tuân thủ những qui tắc của khoa học tính toán, còn người kỹ sư phần mềm là người thừa nhận sự tồn tại của công nghệ và những mặt khớp nối (interface) giữa lập trình và các hệ thống. Lại nữa, thường rất hay xảy ra là cùng một người, lúc này là người lập trình nhưng lúc khác lại là người kỹ sư phần mềm.

Bây giờ tôi muốn giải thích sự khác biệt giữa quá khứ, hiện tại và tương lai. Trong quá khứ, có lẽ đã không có sự phân biệt giữa khoa học máy tính và khoa học tính toán, giữa khoa học chương trình và phương pháp luận của lập trình và đôi khi nảy sinh sự lẫn lộn trong việc phân biệt giữa người lập trình và người kỹ sư phần mềm. Ngày nay, một số nhà khoa học tính toán tin rằng không có CNPM. Theo tôi, người lập trình sẽ làm thành cầu nối giữa lý thuyết và thực hành, còn người kỹ sư phần mềm sẽ làm thành cầu nối giữa lập trình và công nghệ. Lý thuyết phát triển rất chậm. Công nghệ phát triển rất nhanh. Với công nghệ thay đổi, nhiều nguyên lý lập trình như nhau vẫn được áp dụng.

2. Tin học

Trước khi nói thêm về những sự khác biệt của quá khứ, hiện tại và tương lai, hãy cho tôi định nghĩa "Tin học" là gì. Tôi tin rằng những hệ thống tốt nhất của hiện tại và phần lớn những hệ thống của tương lai sẽ được xây dựng trên một số nguyên lý. Tôi sẽ thử liệt kê bốn trong số những nguyên lý đó. Còn có những nguyên lý khác nữa.

Chắc chắn là những hệ thống của chúng ta (1) sẽ được xây dựng phù hợp với các qui tắc của khoa học máy tính và khoa học tính toán, (2) nhưng các hệ thống của chúng ta cũng sẽ phô bầy những đặc điểm được gọi là tri thức, (3) chúng sẽ cho phép liên lạc và đối thoại giữa những con người với nhau thông qua các máy, và (4) chúng sẽ được phát triển bởi những tổ chức năng động và biến đổi.

Do đó việc chúng ta hiểu mỗi một trong những lĩnh vực đó và mối quan hệ giữa chúng là rất quan trọng. Trong các khoa học tính toán và kỹ thuật, ta quan tâm tới những đối tượng nào có thể tồn tại trong máy. Trong nhận thức, chúng ta quan tâm tới làm thế nào để phát hiện ra một cái gì đó là thông tin và làm thế nào để biểu diễn và thao tác thông tin, tri thức và các phương pháp như vậy. Còn về liên lạc, tôi không nói đó là sự liên lạc dữ liệu, mà là sự liên lạc giữa người với người thông qua các máy. Sự liên lạc dữ liệu (trao đổi dữ liệu) có thể là một cách để giải quyết các vấn đề về tính toán, nhận thức, liên lạc và tổ chức. Và như vậy có thể ta sẽ gọi khía cạnh thứ ba của ta là khía cạnh "ngôn ngữ". Ở đây ta quan tâm tới các hình mẫu của ngôn ngữ và các quy tắc cho việc đối thoại. Có nghĩa: cái gì là qui tắc cho việc thiết lập đối thoại giữa người với người và các quy tắc đảm bảo cái gì được truyền đạt sẽ được hiểu đúng ? Khi ta làm việc với tổ chức, ta quan tâm tới việc các tổ chức được cấu trúc như thế nào, những gì là thành phần của các tổ chức, ta phải sắp đặt các thành phần đó như thế nào, chúng tiến triển theo thời gian như thế nào. Như vậy trong tương lai, chúng ta sẽ thấy được sự hợp nhất ngày càng nhiều hơn của các khía cạnh nhân thức trong tính toán.

Theo một nghĩa, khi các bạn để tính toán và nhận thức cùng với nhau, các bạn có được trí tuệ nhân tạo và giờ đây, đang ở Tokyo, tôi thấy không cần nói gì với các bạn về vấn đề này. Tôi nghĩ rằng các bạn biết nhiều hơn tôi. Tôi tin rằng thành công của đề án máy tính thế hệ thứ năm nằm trong chính sự hợp thành của ba lĩnh vực: tính toán, nhận thức và liên lạc (thông báo).

Nhưng có lẽ ta còn cần thiết kế những hệ thống thích nghi tự động với các thay đổi trong các tổ chức sử dụng chúng. Tôi không biết là liệu điều đó có thể được giải quyết không nhưng ít nhất chúng ta cũng thử xem.

3. Công nghệ thông tin và tin học

Bây giờ tôi muốn nêu lên sự khác biệt giữa công nghệ thông tin và tin học. Công nghệ thông tin là một phần của tin học. Nó là một tập con của tin học.

Tôi tin rằng các hệ thống trong quá khứ đại diện cho công nghệ thông tin, và tôi sẽ đặc trưng những hệ đó như sau. Những hệ này gắn liền với công nghệ. Khi ta nói về những hệ như vậy, ta nói về VLSI, bộ nhớ bọt, các màn ảnh plasma, quang học sợi, v.v... Khi ta xây dựng những hệ như vậy, chính vì ta cần tự động các quá trình, ta cần tự động hóa các quá trình tầm thường, những quá trình không thật thú vị đối với con người, hay những quá trình trong công nghiệp nặng có thể làm cho người ta chán ngấy hoặc mệt mỏi, hoặc những quá trình đòi hỏi hàng nghìn người nếu như làm bằng tay.

Nhưng chúng ta phải thấy là, ít nhất là ở nước tôi , những hệ thống như vậy thường đòi hỏi những quy định nghiêm ngặt đối với người dùng tới mức những vấn đề riêng tư của cá nhân có thể bị đe doạ bởi những hệ như vậy. Vì vậy chúng tôi hy vọng trong tương lai những hệ đó sẽ được đưa thêm vào nhiều hơn những khái niệm của tin học.

Khi mà những hệ thống của công nghệ thông tin là hướng công nghệ, ta mong muốn các hệ thống tin học phải là hướng ý tưởng. Khi mà các hệ thống của công nghệ thông tin đã tự động hóa các quá trình, chúng ta muốn rằng các hệ thống tin học phải giúp đỡ trí tuệ con người. Khi mà những hệ thống của công nghệ thông tin đòi hỏi những hệ thống an toàn tính vi rất cao, chúng ta muốn các hệ tin học giúp ta tránh quá nhiều quan liêu.

Như vậy công nghệ thông tin quá khứ là gắn bó với công nghệ trong khi mà tin học tương lai sẽ hướng quan niệm. Công nghệ thông tin quá khứ tập trung vào những vấn đề kỹ thuật, tính tin cậy, dung sai lỗi, tính khoẻ, tính bảo trì được, tính đúng đắn, tính an toàn,...

Công nghệ thông tin quá khứ và hiện tại cùng tập trung vào các vấn đề thị trường như lớn hơn hay nhỏ hơn, nhanh hơn, rẻ hơn, còn chúng ta lại muốn các hệ tin học tương lai tập trung vào những vấn đề của con người, vào những mặt khớp nối người - người thông qua máy và những đối thoại ngôn ngữ tự nhiên.

Nói tóm lại chúng ta muốn rằng những hệ tin học tương lai tập trung vào những gì tốt hơn.

Trong công nghệ thông tin của quá khứ, các nhà vật lý và các kỹ sư điện là những ông chủ. Họ là người chỉ đạo sự phát triển, gọi là tiến bộ. Trong quá khứ, người lập trình và người kỹ sư phần mềm là người nô lệ. Họ phải tuân theo ý tưởng của cánh phần cứng.

Trong tương lai, tôi tin rằng những người lập trình cần phải trở nên thông thạo trong các lĩnh vực của tin học.

Đến đây tôi kết thúc phần chung của bài nói chuyện và bây giờ tôi chuyển sang một phần trong đó tôi nói về CNPM và lập trình là một lĩnh vực hẹp hơn.

4. Công nghệ phần mềm và lập trình

Ba năm trước đây, TS. Boehm cũng thuyết trình ở phòng này và ông đã nói về chu trình sống của phần mềm. Hôm nay tôi sẽ nhìn nó theo một quan điểm hơi khác. Đó là một bức tranh được làm đơn giản đi rất nhiều. Trong đời sống của các hệ mềm hay các hệ nói chung, chúng ta hiểu sự bắt đầu khi mà những yêu cầu của hệ thống được thiết lập, và sau đó ta đi vào sự phát triển của chúng và cuối cùng ta lắp đặt và cho chạy những hệ thống đó. Việc bảo trì những hệ thống đó bao gồm mọi hình thái.

Các nhóm người cũng nắm được các kỹ thật của tin học là quan trọng trong các giai đoạn đầu, những người lập trình ở các giai đoạn giữa, còn những người kỹ sư phần mềm ở các giai đoạn cuối.

Tôi lại nói về sự khác biệt giữa người lập trình và người kỹ sư phần mềm. Người lập trình quan tâm tới ngữ nghĩa, và xem chương trình và lập trình như những đối tượng hình thức. Người kỹ sư phần mềm có liên quan với thực hành, điều khiển và giám sát cả hai vấn đề sản xuất và chất lượng sản phẩm.

Nói rất vắn tắt, vậy những vấn đề về chất lượng của phần mềm là gì ? Tôi nhóm chúng thành hai loại: những vấn đề của sản phẩm. Chúng ta cố gắng để có được những phương pháp dùng có hiệu quả, dùng có dự đoán được, kinh tế và quản lý được, và chúng ta cố gắng để có được những phương pháp dẫn tới những hệ mềm đúng đắn, mạnh mẽ, tin cậy, có hiệu lực, bảo trì được, dung sai lỗi, và an toàn.

Xử lý hàng ngày những vấn đề đó là thuộc phạm vi của người kỹ sư phần mềm.

Đối với người lập trình,anh ta cần xem chương trình và lập trình như những đối tượng hình thức. Theo một cách rất sơ lược, lập trình bao gồm đặc tả, thiết kế và cài đặt và người lập trình cần xem đặc tả như những đối tượng hình thức, xem thiết kế như những đối tượng hình thức, xem cài đặt như những đối tượng hình thức, còn việc biến đổi đặc tả thành thiết kế, và thiết kế thành cài đặt cũng phải xem như những đối tượng hình thức.

5. Đồ thị của đề án

Bây giờ, trong phần ba của bài nói chuyện, tôi sẽ đi vào, xem xét chi tiết về một đề xuất để xem xét các chương trình và lập trình như những đối tượng hình thức.

Trong phần này tôi sẽ nói với các bạn về công việc nghiên cứu và phát triển mà giờ đây tôi đang thực hiện cùng các đồng nghiệp của tôi. Vì vậy tôi sẽ thay đổi cách nói và trình bày với các bạn một chủ đề mang tính kỹ thuật. Tôi muốn giới thiệu với các bạn khái niệm đồ thị của đề án. Một mặt chúng là những đồ thị có hướng không chu trình. Mặt khác chúng xác định những phương án (kế hoạch) cho sự phát triển phần mềm. Theo nghĩa đó, chúng làm thành một mô hình cho sự phát triển phần mềm.

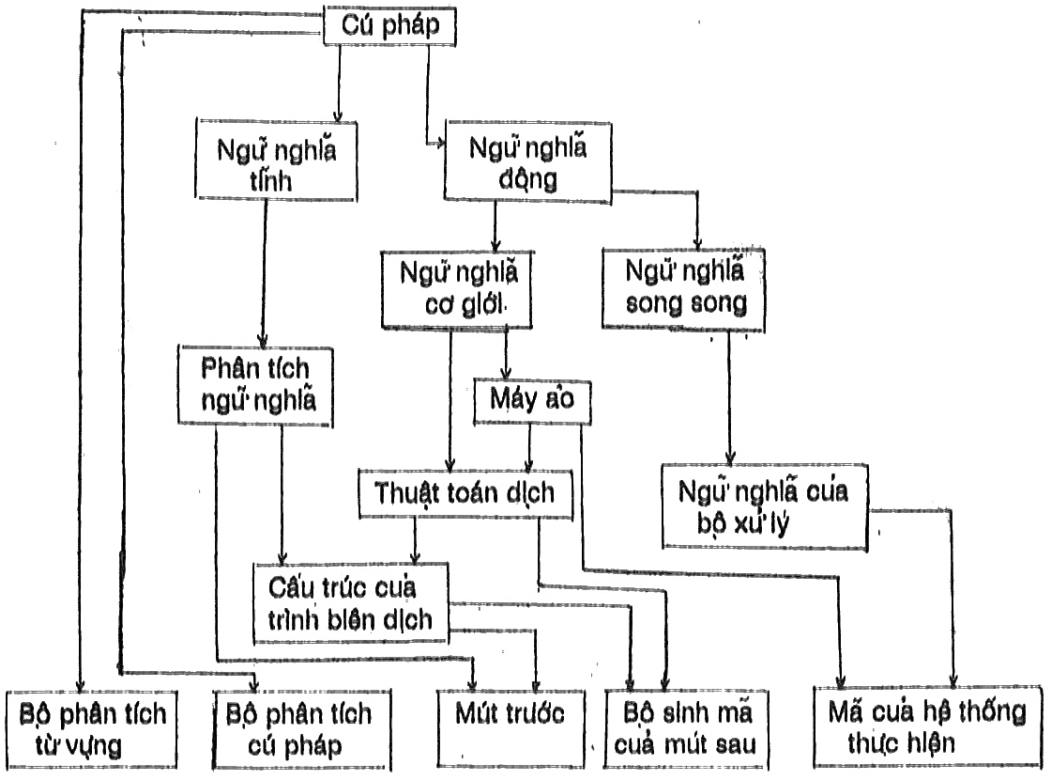
Các nút của đồ thị biểu diễn những hoạt động dẫn tới các tài liệu, và những tài liệu đó thuộc những lý thuyết nào đó. Các cung giữa các nút biểu diễn những quan hệ giữa các tài liệu. Đồ thị đặc biệt tôi giới thiệu với các bạn ở đây được dùng trong việc triển khai các chương trình biên dịch cho những ngôn ngữ rất phức tạp. Về cơ bản đây chính là đồ thị của đề án mà chúng tôi theo đuổi trong một trung tâm nghiên cứu và phát triển phần mềm ở Đan Mạch để sản xuất không những chỉ những chương trình dịch cho Ada mà còn cả những chương trình dịch cho CHILL.

Về cơ bản đồ thị ứng với ba phần: phần đặc tả hay phần định nghĩa hình thức, phần thiết kế và ở dưới đáy là phần mã.

Tôi sẽ giải thích một ít về sơ đồ này. Theo một nghĩa, ta thực hiện đồ thị đề án đó, và tới cuối của phần ba tôi sẽ chỉ cho các bạn một phác thảo của một máy thực hiện những đồ thị đề án như vậy. Nhưng các bạn nhớ rằng, đồ thị đề án này chỉ áp dụng được cho một lớp phần mềm, giống như các chương trình dịch. Các đồ thị đề án cho việc phát triển hệ điều hành trông khác hoàn toàn, và cũng khác với các đồ thị đề án cho việc phát triển của các hệ quản trị cơ sở dữ liệu.

Việc thực hiện những đề án đó bắt đầu với những nút không có lối vào và tiến hành theo một thứ tự tôpô - có thể thực hiện song song một số hoạt động.

Như vậy, vì các ngôn ngữ lập trình là một lĩnh vực được nhiều người hiểu biết và cũng vì bản thân tôi cũng thông thạo nhất với sự phát triển của các chương trình biên dịch, bây giờ tôi sẽ nói với các bạn chi tiết về thí dụ này.



Đồ thị đề án của việc triển khai trình biên dịch.

5.1. Lập trình

Trước hết, chúng tôi thiết lập cú pháp của ngôn ngữ lập trình, và độc lập với nhau, từ đó ta phát triển cả ngữ nghĩa tĩnh và động. Các định nghĩa của ngữ nghĩa tĩnh xác định tất cả những tính chất tĩnh của chương trình. Với một lớp ngôn ngữ nhất định, các ngôn ngữ tựa - Algol, tựa - Pascal, tựa - Ada, một số đặc điểm của ngôn ngữ có thể được kiểm tra tĩnh, chẳng hạn như các tên dùng đã được định nghĩa, các toán hạng thuộc kiểu mong đợi bởi các phép toán, cũng như có sự sánh hợp giữa các danh sách tham số và đối số của các thủ tục, và nhiều cái khác nữa.

Ngữ nghĩa động nói về các tính chất của chương trình khi chạy. Cuối cùng, các ngữ nghĩa tĩnh sẽ được xem xét, chăm lo bởi mút trước của chương trình biên dịch của bạn, và nếu bạn đang phát triển một trình biên dịch thì ngữ nghĩa hình thức sẽ dẫn tới mã được sản sinh ra bởi mút sau của trình biên dịch của bạn. Trong định nghĩa hình thức ta cố gắng thực hiện nó càng trừu tượng càng tốt để hiểu được bài toán (vấn đề) là gì, ngôn ngữ mà ta đang xây dựng trình biên dịch là gì. Như vậy với sự trừu xuất ta có thể chế ngự được độ phức tạp, và ta có thể hiểu cái mà ta phải cài đặt là gì.

Bây giờ ta phải tìm xem cài đặt cái đó như thế nào. Định nghĩa trừu tượng là quá trừu tượng để có thể dùng trực tiếp cho sự mã hóa, nên ta biến đổi ngữ nghĩa tĩnh trừu tượng thành một cái gì đó cụ thể hơn, và cũng tương tự như vậy đối với ngữ nghĩa động, ta biến đổi nó thành một cái gì đó cụ thể hơn cả đối với khía cạnh tuần tự cũng như các khía cạnh song song của ngôn ngữ.

Từ một ngữ nghĩa cơ giới ta có thể suy dẫn, có thể xây dựng một máy ảo để thực hiện những chương trình Ada hay khi không có cung từ nút ngữ nghĩa cơ giới tới nút ab ta có thể chấp nhận một máy đích hiện có. Từ ngữ nghĩa cơ giới nói với ta cấu trúc của chương trình ở thời gian thực hiện, và từ máy ảo ta có thể cùng phát triển thuật toán dịch. Ở những nút phân tích ngữ nghĩa và thuật toán dịch ta đã đặc tả cái mà chương trình biên dịch sẽ thực hiện mà không phải là như thế nào, nên bây giờ ta

đã sẵn sàng tìm xem phải thực hiện nó như thế nào. Như vậy, ở nút cấu trúc trình biên dịch, ta quyết định cấu trúc của chính trình biên dịch. Từ cú pháp ta có thể tự động tạo sinh bộ phân tích từ vựng và bộ phân tích cú pháp của chương trình dịch. Từ cái phải kiểm tra, từ mã nào phải sinh ra và sinh nó như thế nào ta mã mút cuối. Như vậy bốn phần: bộ phân tích từ vựng, bộ phân tích cú pháp, bộ kiểm tra kiểu (mút trước), và bộ sinh mã (mút sau) làm thành trình biên dịch và tương tự như vậy ta có thể phát triển thiết kế, và mã cho hệ thống thực hiện của máy đích.

Như vậy đó là một kế hoạch rất chi tiết để phát triển nhiều tài liệu theo một thứ tự nhất định.

5.2. Công nghệ phần mềm

Bây giờ ta bàn luận về kế hoạch đó theo quan điểm CNPM. Vào tháng 11 năm 1980 chúng tôi đã thiết lập xong đồ thị đề án này và chúng tôi hy vọng sẽ tốn vào đó 33 người trong 4 năm để sản sinh ra một trình biên dịch. Chúng tôi đã dự kiến số người và thời gian cho từng hoạt động một và đã bị chậm mất 33 phần trăm. Chúng tôi đã quá lạc quan và đã nghĩ rằng mỗi năm sẽ dùng 33 người, nhưng thực tế đã phải dùng 44 người.

Sáu người mất một năm để làm định nghĩa hình thức. Theo một nghĩa, trong tổng thời gian 4 năm, đã tốn mất một năm mà chưa có nghĩ gì về trình biên dịch.

Tiếp đó 12 người mất một năm, làm thiết kế "là cái gì" cho trình biên dịch, và hai người mất một năm cho hệ thống thực hiện. Sau hai năm làm việc cho chương trình dịch, chúng tôi vẫn chưa biết cấu trúc của trình biên dịch.

Tiếp đó 10 người mất khoảng nửa năm chỉ nghĩ về cấu trúc chi tiết của trình biên dịch.

Sau hai năm rưỡi chúng tôi chưa mã được một dòng nào của trình biên dịch. Thưa các bạn, đó là một đề án chương trình dịch, và sau hai năm rưỡi trời, chúng tôi vẫn chưa mã hóa được gì cả !

Tiếp đó 12 người mất một năm rưỡi vào việc mã hóa. Những người làm ngữ nghĩa tĩnh đã không làm phân tích ngữ nghĩa. Chúng tôi chuyển họ vòng quanh. Những người làm ngữ nghĩa động làm phân tích ngữ nghĩa. Những người mới đã tới ở mức ngữ nghĩa. Sự chậm trễ trong đề án không phải là do những người mới mà là do những thay đổi trong Ada. Ở trong giai đoạn mã hóa, chúng tôi đã dùng một loạt những người mới. Họ không biết gì về Ada, cũng như không biết gì về ngôn ngữ dịch, nhưng họ đã mã hóa được chương trình dịch vì họ đã theo đúng những đặc tả thiết kế cơ sở.

Mất 14.000 dòng để đặc tả định nghĩa hình thức, 59.000 dòng để định nghĩa thiết kế, và 220.000 dòng để mã chương trình dịch.

5.3. Khoa học máy tính lý thuyết

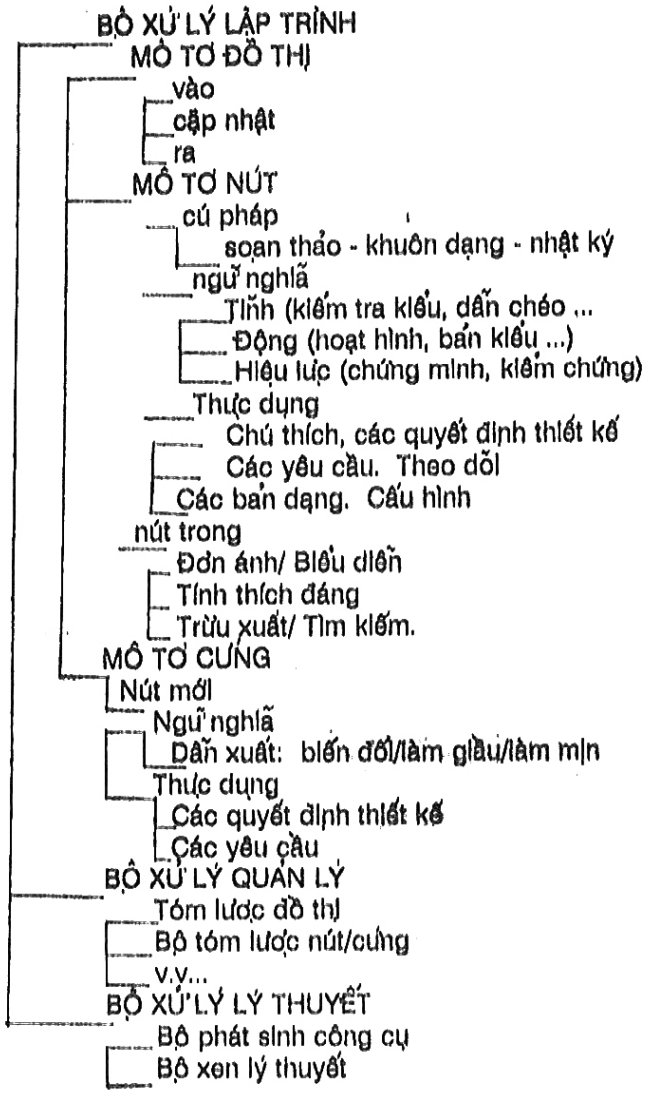
Tôi đã nói về đồ thị đề án đó theo quan điểm lập trình và theo quan điểm CNPM. Bây giờ tôi nói về nó theo một quan điểm lý thuyết hơn. Những tài liệu được sản sinh ra biểu thị những đối tượng của những lý thuyết nhất định. Chẳng hạn, cú pháp cụ thể là đối tượng trong lý thuyết các văn phạm và ngôn ngữ phi ngữ cảnh. Cú pháp trừu tượng là một đối tượng trong một miền Scott của những cái được gọi là khước từ (retracts).

Những nút khác biểu thị các đối tượng trong những lý thuyết khác. Chẳng hạn thuật toán dịch là một đối tượng trong lý thuyết văn phạm thuộc tính suy rộng và, như các bạn đã biết, bộ phận tích ngữ nghĩa là một đối tượng trong lý thuyết các máy hữu hạn trạng thái. Bộ phân tích cú pháp là đối tượng trong một lý thuyết các máy đẩy xuống, còn các chương trình, được viết cho các mút trước và sau theo một nghĩa là các đối tượng trong một hoare-logic của chứng minh các chương trình đúng.

5.4. Một hệ thống phát triển phần mềm.

Bây giờ ta muốn triển khai một công nghệ phần mềm tương lai để phát triển phần mềm. Ta muốn phát triển một hệ trợ giúp cho việc thực hiện các đồ thị đề án cho cả người lập trình, người kỹ sư phần mềm, nhà khoa học và nhà quản lý.

Điều mà các bạn cần nhớ là ta đang thực hiện những đồ thị đó và ta cần một hệ thống mềm - cứng thực hiện đồ thị này một cách tương tác cùng với những người lập trình, những người kỹ sư phần mềm và những nhà quản lý. Vì vậy ta cần một hệ mềm phát triển. Về cơ bản hệ này có hai phần chính: một phần cho việc thực hiện các đồ thị đề án và một cái gọi là máy lý thuyết.



Máy thứ nhất, máy đồ thị đề án, cơ bản gồm hai phần, trước hết là một môtơ để định nghĩa, thay đổi và in ra các đồ thị đề án. Trước khi ta bắt đầu phát triển, thứ nhất ta triển khai đồ thị đề án. Đồ thị đề án giống như một siêu trình (Meta - program), một chương trình để xây dựng những chương trình khác. Thứ hai là ta có phần chăm lo tới các nút thực hiện và các cung của đồ thị đề án. Giả sử là tôi đang thực hiện một nút, điều đó có nghĩa là gì ? Có nghĩa là cái gì đó vừa là cú pháp, ngữ nghĩa và thực dụng. Về cú pháp ta cần phải soạn thảo, làm khuôn dạng, bảo trì các bản dạng (versions), cấu hình, cần tới những thay đổi của nhật ký tới các tài liệu. Và sau đó chúng ta muốn xử lý những tài liệu đó theo ngữ nghĩa. Ta muốn máy của chúng ta hiểu ý nghĩa của những gì ta viết ra. Ta muốn có khả năng kiểm tra kiểu, thực hiện ngữ nghĩa tĩnh của nhiều tài liệu ta làm ra. Ta muốn có khả năng thực hiện một cách hình thức, hay một cách nào đó làm hoạt hình và tạo bản kiểu nhanh những đặc tả hay thiết kế đó và ta muốn phê chuẩn (xác nhận tính hiệu lực), chứng minh hãy kiểm chứng những tính chất của những tài liệu của ta.

Như vậy, theo cách đó thoạt tiên chúng ta đã tạo ra một tài liệu và sau đó xem xét hay phân tích tài liệu và vì vậy ta có thể nghĩ là bây giờ ta kết thúc với tài liệu đó và với nút đặc biệt đó và ta muốn sản sinh một nút mới. Hoặc là ta sản sinh nút mới này theo cách như với nút đầu tiên và sau khi đã sản sinh ra hai nút kế nhau với một cung giữa chúng, ta phải chứng minh những tính chất về mối quan hệ giữa chúng. Hoặc là ta có thể theo một cách nào đó suy dẫn ra nút mới từ nút cũ bằng những bước biến đổi, tinh chế hay làm giầu. Ở đây ta muốn máy hãy hệ thống hiểu được những phép toán trên tài liệu bảo toàn ý nghĩa. Nhưng dù thế nào đi nữa, những phép biến đổi mà ta thực hiện đều là kết quả của những quyết định thiết kế nhất định, và ta muốn lưu giữ những quyết định

thiết kế đó, và đó là một nội dung cú pháp. Và ta muốn giữ lại được những yêu cầu trước đó: rằng chúng thực sự được thể hiện trong thiết kế mới. Như vậy, phần lập trình, bộ xử lý lập trình của máy đồ thị đề án của ta đảm nhiệm việc tạo và thực hiện các đồ thị đề án.

Còn có một khía cạnh khác của máy đồ thị đề án, đó là khía cạnh quản lý: Việc quản lý cũng phải có khả năng kiểm tra và giám sát sự phát triển, ta đã đi xa tới đâu trong đồ thị, có bao nhiêu việc phải làm ở một nút hay một cung, ...

Chúng tôi đang phát triển hệ thống này không phải cho một ngôn ngữ đặc tả và thiết kế riêng biệt mà là cho cả một loạt. Chúng ta cần một loạt để quan tâm tới những tính chất khác nhau của hệ thống, những tính chất tất định và không tất định, những tính chất tương tranh và những tính chất phụ thuộc thời gian. Hiện nay, vào năm 1985 chưa có một ngôn ngữ, đặc tả nào cũng như chưa có một ngôn ngữ thiết kế nào quan tâm đều như nhau tới tất cả những khía cạnh đó. Ngày nay có một số lý thuyết, nhưng ngày mai có thể có những lý thuyết khác. Ta cần có khả năng đưa xen thêm những lý thuyết mới vàb máy của ta sao cho nó có thể hiểu được ý nghĩa tĩnh và động của các ngôn ngữ đặc tả mới và của những ngôn ngữ thiết kế mới.

5.5. Đề án Raise

Đó là một đề án rất lớn và tôi hy vọng rằng nó là điển hình. Tôi hy vọng chắc chắn rằng nó là điển hình của các công nghệ phần mềm trong tương lai, là chủ đề của bài nói chuyện của tôi hôm nay. Chúng tôi gọi đề án đó là RAISE hay một cách tiếp cận tới công nghệ phần mềm công nghiệp (Rigorous Approach to Industrial Software Engineering). Nó là một phần trong chương trình ESPRIT của khối cộng đồng kinh tế châu Âu. Chương trình này có sáu lĩnh vực và nó thuộc lĩnh vực thứ hai: công nghệ phần mềm. Hiện nó là đề án lớn nhất trong lĩnh vực này. Những đề án lớn có khoảng 100 người năm. Đây là đề án do bốn hãng cùng theo đuổi: Trung tâm nghiên cứu và phát triển phần mềm không kiếm lãi Danish Datametics Center; Một hãng máy tính mạnh Nordic Brown Bovery, đó là chi nhánh Đan Mạch của một hãng Thụy Sĩ; và hai hãng của Anh là STL và ICL. Đề án này hy vọng sẽ sản sinh ra được ba loại kết quả: những kết quả trí tuệ dưới dạng các phương pháp và các kỹ thuật; những kết quả vật chất dưới dạng môi trường phát triển phần mềm hay hệ trợ giúp phát triển phần mềm, bao gồm các hệ trợ giúp chuyên gia; và còn cả các kết quả trí tuệ dưới dạng các tài liệu giáo dục, đào tạo và sự vận chuyển công nghệ vào công nghiệp dưới dạng các giáo trình.

Đó là một đề án rất mạo hiểm và vì vậy chúng tôi đã phải mời một số cố vấn. Trong số các bạn ở đây, có thể có người biết Giáo sư Manfred Broy ở Đức, TS. Joseph A. Goguen o SRI California, GS. Ugo Montanari ở Ý, GS. Cliff B. Jones ở Manchester, TS. Gordon Plotkin ở Scotland, TS. J.R. Abrial ở Paris. Với sự giúp đỡ của họ, hy vọng là chúng tôi sẽ tránh được việc sa vào quá nhiều vấn đề.

6. Kết luận.

Bây giờ tôi trình bày phần thứ tư và cũng là phần cuối của bài nói chuyện của tôi. Trên cơ sở của ba phần trên, tôi muốn nêu lên một số kết luận. Ở mức trừu tượng, tôi đã nói về công nghệ thông tin và tin học. Ở một mức cụ thể hơn tôi đã nói về sự khác biệt giữa CNPM và lập trình và ở mức rất kỹ thuật tôi đã minh họa một trong số rất nhiều loại khác nhau có thể về các hệ trợ giúp phát triển phần mềm và do đó ta có thể đặt câu hỏi: điều đó ảnh hưởng như thế nào tới (1) CNPM, (2) Lập trình, (3) quản lý, (4) các hãng sản xuất máy tính và phần mềm, (5) các đại học ?

Các bạn nhớ cho là thường thì người kỹ sư phần mềm và người lập trình là cùng một người nhưng tôi tin rằng anh ta làm những công việc khác nhau ở những thời điểm khác nhau.

6.1. Công nghệ phần mềm.

Vậy thì những đòi hỏi của CNPM là gì ? Theo tôi là phải hiểu vai trò của CNPM giữa lập trình và các hệ thống. Chúng ta nói nhiều thứ tiếng. Tôi đã định nghĩa một số thuật ngữ với các bạn, có lẽ phần lớn chúng là mới. Tệ hại hơn nữa là chúng có thể mâu thuẫn (đối chọi) với những định nghĩa hiện có cho cùng những từ đó. Xem như một nghề, chúng tôi phải thiết lập, tuân thủ và phát triển tiếp một thuật ngữ mà nhờ vào đó chúng tôi có thể trao đổi như những người kỹ sư. Niềm tin của tôi là khi chúng ta hành động như những người kỹ sư phần mềm, chúng ta vẫn phải thừa nhận rằng chương trình và lập trình là những đối tượng hình thức. Thừa nhận như vậy có thể có khó khăn đối với người kỹ sư phần mềm, nhưng tôi nhìn thấy vai trò của họ là phát triển hay tạo ra những sách tra cứu phát triển kỹ thuật.

Như vậy, theo tôi những người kỹ sư phần mềm cơ bản là những người xây dựng công cụ.

6.2. Lập trình.

Những đòi hỏi đối với những người lập trình hay đối với lập trình là gì ? Theo tôi họ cần nắm vững nhiều kỹ thuật khác nhau. Thứ bảy vừa rồi tôi đi Shibuya, tới Tokyu Hands và đã mua hết 23.080 yên các thứ dụng cụ của Nhật - cưa, bào, dao, búa ... tất cả đều có hình thức rất đẹp và rất tiện dụng. Tôi chưa sử dụng ngay những dụng cụ đó. Tôi sẽ treo chúng lên tường để ngắm vẻ đẹp của chúng và tự nhủ mình rằng để đóng một thứ đồ gỗ đẹp ta cần tới nhiều, rất nhiều loại công cụ khác nhau.

Những người lập trình cần học đặc tả trừu tượng như thế nào, học thiết kế cái gì và cách nào như thế nào, và học biến đổi, làm mịn (tinh chế) và làm giầu các đặc tả thành thiết kế như thế nào. Họ cũng phải học chứng minh, kiểm chứng và phê chuẩn công việc của họ, không phải chỉ trong một ngôn ngữ mà trong nhiều loại ngôn ngữ khác nhau.

Những người lập trình đã thành công quá nhiều nên họ vẫn còn đang làm việc với những công cụ thời kỳ đồ đá, khác với nhưng đồng nghiệp phần cứng của họ đã đòi hỏi những công cụ rất tinh tế để thiết kế những mạch VLSI. Thưa các bạn, đã tới lúc mà những người lập trình đều có và đòi hỏi những công cụ cũng tinh tế và thậm chí còn tinh tế hơn so với những kỹ sư phần cứng. Tại sao họ đòi hỏi cái đó ? Vì rằng những đối tượng họ thiết kế còn phức tạp hơn nhiều những đối tượng phần cứng. Những người lập trình là những người xây dựng lý thuyết.

6.3. Quản lý.

Những đòi hỏi của quản lý là gì ? Để hiểu rằng lập trình là một khoa học, để không bao giờ tiến hành một đề án chừng nào chưa hiểu kỹ đồ thị của nó, để chỉ dùng những người quản trị và những người giám sát ở mức cao nhất, có khả năng tự làm lấy được. Ngày nay, nếu ta theo đúng ba quy tắc đó, sẽ không có phần mềm nào được sản xuất trong mười năm. Đối với quản lý còn có những yêu cầu khác. Những người quản lý là những người xây dựng tổ chức.

6.4. Công nghiệp và thương mại.

Tôi đã làm việc trong 13 năm ở một trong những hãng sản xuất máy tính lớn. Trong thế giới phương tây, hàng này chiếm hơn 60 % thị trường, cho nên tôi nghĩ là các bạn biết tôi đang nói về ai.

Trong phòng thí nghiệm kỳ diệu của Giáo sư Zemanek ở Vienna, chúng tôi đã cố gắng triển khai phương pháp riêng của chúng tôi. Với kinh nghiệm chua xót, tôi đang khuyên các nhà kỹ nghệ đừng có làm cái đó trừ khi họ có khả năng cùng một lúc gặt hái những thành quả của lao động học thuật, của những thành quả đã được triển lãm quốc tế và được tinh chế.

Các hãng lớn, tôi tin rằng cần phải biết rằng lập trình là một quá trình trí tuệ, rằng phần mềm là một sản phẩm trí tuệ, rằng những biện pháp để hiểu phần cứng là không áp dụng được.

6.5. Học viện.

Những đòi hỏi đối với các đại học là gì ? Trước hết tôi tin rằng phần lớn những dạy dỗ ở đại học là hỏng và đó là đề tài của một bài nói chuyện riêng biệt. Tôi đã thấy tốn quá nhiều thời gian về phía tôi trong công việc cùng với các nhóm đại học lao vào triển khai những bản kiểu (prototype). Họ cần phải tập trung nhiều hơn nữa vào lý thuyết.

6.6. Tóm tắt.

Bây giờ tôi đi tới một số nhận xét cuối cùng. Vì vậy để kết luận tôi muốn nói tóm tắt rằng: tôi tin rằng công nghệ thông tin là một thế giới diệu kỳ, nhưng đó là đại lượng vật chất, rằng khi chúng ta thêm vào công nghệ thông tin, như các bạn hiện đang làm ở Nhật, các khía cạnh trí thức, với các khía cạnh liên lạc hay ngôn ngữ, với các khía cạnh tổ chức và có lẽ cả với các khía cạnh sinh học, thì các bạn đi vào thế giới của chất lượng trí tuệ, đi vào tin học.

Theo tôi lập trình là kiến trúc của các ý tưởng trong đó CNPM là sự thực hành biến những ước mơ đó thành hiện thực.

Tôi tin rằng những nhà khoa học tính toán của tương lai cần phải hiểu quá khứ; không phải là máy tính hay khoa học tính toán trong quá khứ - nhưng cái đó không có lợi gì mấy - mà là những lý thuyết và triết học của khoa học trong quá khứ và tương lai.

Khi các bạn có những ý mời tôi đến đây và nói chuyện với các bạn, tôi được gợi ý nói về một cái gì đó của tương lai, và tôi đã tự nhủ là người ta không bao giờ mời một nhà khoa học nói chuyện về tương lai. Cần đòi hỏi một nhà khoa học nói về cái mà anh ta hiện đang làm vì rằng đó chính là tương lai.

Xin cám ơn các bạn.

Hồ Thuần dịch
Tháng 4/1988