ĐẠI HỌC BÁCH KHOA HÀ NỘI

Báo cáo

TÌM HIỂU VỀ MỘT SỐ MÔ THỨC LẬP TRÌNH

Môn học:	Kỹ thuật lập trình
Giảng viên:	Vũ Đức Vượng
Sinh viên thực hiện:	Nguyễn Ngọc Huyền
Mã số sinh viên:	20131821
Lớp:	CNTT-TT 2.03 K58

Hà Nội, tháng 7 năm 2015

Mục lục

Lời nói đầu

Mô thức lập trình là một mô hình tổng quát được dùng cho một lớp các ngôn ngữ lập trình có cùng những đặc trưng cơ bản. Đó là cách tiếp cận tổng quát, là hướng giải quyết hoặc nguyên lý lập trình có thể được sử dụng khi thực hiện chương trình. Một mô thức lập trình có thể được triển khai cho những vấn đề xác định, cho ra kết quả tối ưu hơn so với các mô thức khác. Đồng thời, một mô thức có thể được sử dụng để giải quyết những vấn đề nhất định tốt hơn các vấn đề khác.

Một số ngôn ngữ được thiết kế để hỗ trợ duy nhất một mô thức (Smalltalk hỗ trợ lập trình hướng đối tượng, Haskell hỗ trợ lập trình hàm), trong khi đó một ngôn ngữ lập trình có thể hỗ trợ nhiều mô thức (multi-paradigm), như Common Lisp, Java, C#, C++.... Ví dụ, chương trình được viết bằng ngôn ngữ C++ hoặc Java có thể là lập trình thủ tục thuần túy, lập trình hướng đối tượng thuần túy hoặc có thể chứa những thành phần của cả hai mô thức trên.

Nhiều mô thức lập trình được ra đời, trong đó có bốn mô thức lập trình cơ bản là:

- Lập trình thủ tục - Imperative paradigm
- Lập trình hàm - Functional paradigm
- Lập trình lo-gic - Logical paradigm
- Lập trình hướng đối tượng - Object-oriented paradigm

Bên cạnh đó có thể kể đến:

- Lập trình trực quan – Visual paradigm
- Lập trình song song – Parallel paradigm
- Lập trình tương tranh – Concurrent programming
- Lập trình phân tán – Distributed programming
- Lập trình cực hạn – Extreme programming

Phần 1: SƠ LƯỢC VỀ MỘT SỐ MÔ THỨC LẬP TRÌNH

1. Lập trình hàm – Functional paradigm

Khái niệm

Lập trình hàm (Functional Paradigm) là một mô thức lập trình xem việc tính toán là sự đánh giá các hàm toán học và tránh sử dụng trạng thái và các dữ liệu biến đổi.

Lập trình hàm được xuất phát từ phép tính lamda, một hệ thống hình thức được phát triển vào những năm 1930 để nghiên cứu định nghĩa hàm số , ứng dụng của hàm số , và đệ quy. Do lập trình hàm có nguồn gốc từ toán học thuần túy đó là lý thuyết hàm , vì vậy so với lập trình thủ tục (Imperative Paradigm) thì lập trình hàm đơn giản , rõ ràng và không gây ra hiệu ứng phụ như vẫn gặp trong lập trình thủ tục.

Các ngôn ngữ lập trình hàm được sử dụng chủ yếu là : *MathLab, Lisp, Haskell, F#.*

Các đặc trưng cơ bản

- Về mặt nguyên lý và ý tưởng : Toán học và lý thuyết hàm
- Các giá trị tạo được là không thể biến đổi *non-mutable*
- Không thể thay đổi các yếu tố của giá trị hợp thành
- Giống như phương thuốc, có thể tạo một phiên bản của các giá trị hợp thành : một giá trị trung gian
- Trừu tượng 1 biểu thức đơn thành 1 hàm và hàm có thể tính toán như là 1 biểu thức
- Các hàm là những giá trị đầu tiên
- Hàm là dữ liệu hoàn chỉnh, giống như số, danh sách, ...
- Thích hợp với xu hướng tính toán theo yêu cầu
- Mở ra những khả năng mới.

Lập trình hàm xem chương trình là một tập hợp các hàm nhận vào đối số và trả về giá trị. Hàm trong lập trình hàm rất giống với hàm trong toán học vì nó không làm thay đổi trạng thái của chương trình. Về mặt kĩ thuật cài đặt bên dưới, khi giá trị được gán vào một vùng nhớ thì được đánh dấu là đã sử dụng và không ghi đè lên nữa. Để tạo ra k ết quả trả về, các hàm sao chép giá trị rồi chỉnh sửa trên các bản sao đó, không làm ảnh hưởng

đến giá trị ban đầu, rồi trả về bản sao đã được chỉnh sửa. Các đại lượng không còn được hàm nào tham chiếu đến sẽ tự động bị hủy để giải phóng bộ nhớ (đây là ý tưởng của bộ thu gom rác trong Java và NET).

Cơ sở toán học cho lập trình hàm rất tường minh, cung cấp giải pháp logic và ngắn gọn cho các vấn đề tính toán. Tuy nhiên do nó không linh hoạt thay đổi trạng thái và chuyên sử dụng đề quy khiến người ta ưa chuộng các mô thức khác hơn để xử lý các thao tác tính toán thông dụng.

2. Lập trình logic – Logical paradigm

Khái niệm

Lập trình logic (Logical Paradigm) là một mô thức lập trình mà theo đó các bài toán được đưa về bài toán suy diễn logic toán học sau khi đã mô tả (phát biểu) các dữ kiện liên quan bằng mệnh đề logic.

Trong lập trình logic, ta có thể sử dụng các vị từ để định nghĩa các khái niệm của tất cả các môn khoa học khác.Lập trình logic phù hợp với các lĩnh vực liên quan đến việc rút ra những kiến thức từ sự kiện và quan hệ cơ bản –lĩnh vực trí tuệ nhân tạo và lập trình này hoàn toàn khác so với các lập trình khác , nó không gắn liền với một lĩnh vực tính toán chung nào.

Ngôn ngôn ngữ lập trình logic được sử dụng chủ yếu đó là : *Prolog*.

Các đặc trưng cơ bản

- Về nguyên tắc và ý tưởng : Tự động kiểm chứng trong trí tuệ nhân tạo
- Dựa trên các chân lý- tiên đề, các logic toán học,các quy luật suy diễn , và các truy vấn.
- Chương trình thực hiện từ việc tìm kiếm có hệ thống trong 1 tập các sự kiện, sử dụng 1 tập các luật để đưa ra kết luận.

Cụ thể hơn, khi mô tả bài toán dưới dạng mô thức lập trình logic, ta có thể yêu cầu hệ thống tìm kiếm các lời giải liên quan đến các khai báo đó. Bài toàn cần giải được xem

là “mục tiêu” mà hệ thống phải chứng minh trên cơ sở các tri thức đã được khai báo. Như thế, toàn bộ các kỹ hiệu của ngôn ngữ lập trình suy về một công thức đặc biệt:

- Phát sinh từ một yêu cầu

- Nhằm chứng minh một mục tiêu. Để trả lời cho câu hỏi đó hệ thống xem nó như là một đích và cố chứng minh đích đó bằng cách tạo những suy diễn trên cơ sở các tri thức đã khai báo.

Một ngôn ngữ logc có thể được dùng trong gia đoạn đặc tả yêu cầu của quy trình xây dựng một sản phẩm phần mềm. Hơn thế nữa, mô thức lập trình logic cho phép biểu diễn hầu hết các khái niệm và các định lý trong các bộ môn khoa học.

3. Lập trình trực quan – Visual paradigm

Khái niệm

Lập trình trực quan (Visual Paradigm) là mô thức lập trình cho phép người sử dụng tạo ra các chương trình bằng công cụ đồ họa. So với các ngôn ngữ lập trình bằng chế độ văn bản thì lập trình trực quan dễ sử dụng hơn do không phải nhớ các câu lệnh.

Lập trình trực quan là sự kết hợp giữa đồ họa máy tính, ngôn ngữ lập trình và tương tác người máy.

Các ngôn ngữ lập trình tr ực quan thường được sử dụng là :*Visual Basic, Visual FoxPro, Visual C, Delphi*…

Các đặc trưng cơ bản

- Về nguyên tắc và ý tưởng : giao tiếp với máy bằng chế độ đồ họa.

- Cho phép xây dựng chương trình theo hướng sự kiện (Event-Driven Programming), một chương trình ứng dụng được viết theo kiểu này đáp ứng dựa theo tình huống xảy ra lúc thực hiện chương trình. Tình huống này bao gồm người sử dụng ấn một phím tương ứng, chọn lựa một nút lệnh hoặc gọi một lệnh từ ứng dụng khác chạy song song cùng lúc.

- Tự tạo khung giao diện, ứng dụng thông qua các thông tác trên màn hình dựa vào các đối tượng.

- Rất ít khi phải tự viết lệnh , tổ chức chương trình một cách rắc rối mà chỉ cần khai báo việc gì cần làm khi tình huống xuất hiện.

- Máy tính dựa vào phần thiết kế và khai báo của lập trình viên để tự động tạo lập chương trình.

- Khó thực hiện khi có một tình huống bất ngờ xuất hiện.

- Người lập trình trực tiếp tạo ra các khung giao diện (interface), ứng dụng thông qua các thao tác trên màn hình dựa vào các đối tượng (object) như hộp thoại, nút điều khiển, mang các thuộc tính riêng biệt như màu sắc, font chữ…

4. Lập trình song song – Parallel paradigm

Khái niệm

Hiện nay, để giải quyết các bài toán lớn người ta thường nghĩ đến việc sử dụng các siêu máy tính hoặc việc kết hợp nhiều máy tính với nhau để tính toán. Tuy nhiên, với phương pháp lập trình cổ điển thì không thể nào phát triển được chương trình có thể tận dụng được sức mạnh của các hệ thống đó. Đó chính là lý do lập trình song song ra đời.

Tính toán song song là việc chia một công việc ra thành các công việc nhỏ và cho các công việc này thực hiện đồng thời với nahu bởi các hệ thống có nhiều bộ vi xử lý hay bộ xử lý đa nhân nhằm giảm thời gian thực hiện công việc đó xuống. Việc lập trình để tách ra các công việc nhỏ và sắp xếp để xử lý song song thì được gọi là lập trình song song.

Lập trình song song là một công việc rất phức tạp so với lập trình tuần tự thông thường, người phát triển phải thực hiện một quá trình “song song hóa”, biến đổi các chương trình tuần tự thành chương trình song song có khả năng tận dụng tối đa sức mạnh của hệ thống.

Lập trình song song bao gồm các bước :

- Phân hủy một thuật toán hay các dữ liệu đầu vào.
- Phân chia nhiệm vụ cho các bộ phận để làm việc trong bộ vi xử lý cùng một lúc.
- Phối hợp và trao đổi giữa các bộ vi xử lý.

Các ngôn ngữ lập trình song song thường được sử dụng là : *CUDA,Ada*.

Các đặc trưng cơ bản

- Về mặt nguyên lý và ý tưởng : Tăng tốc độ xử lý của CPU hay siêu máy tính.
- Cải thiện hiệu suất và hiệu quả của chương trình,
- Các chương trình được chia thành nhiều phần và được thực hiện đồng thời

- Lập trình song song là lập trình để chạy được đa luống với các dòng CPU đa nhân hiện nay (core i3, core i5, core i7,…)

Bộ vi xử lý có nhiều nhân sẽ tăng tốc chương trình song song tuy nhiên không có nghĩa là nó sẽ tăng lên 100% trên một nhân (core) được thêm vào. Thậm chí chương trình song song không hề tăng hiệu suất lên trong một số trường hợp. Vì vậy lập trình viên phải biết quyết định khi nào sử dụng lập trình song song bằngcách sử dụng các công cụ đo lường để xác định được tốc độ thực thi của chương trình.

5. Lập trình tương tranh – Concurrent programming

Khái niệm

Lập trình tương tranh (Concurrent Programming) là một mô thức lập trình trong đó chương trình thiết kế sao cho các quá trình tính toán , tương tác với máy tính có thể được thực hiện cùng một lúc.

Lập trình tương tranh có thể được thực hiện liên tục trên một bộ vi xử lý duy nhất bởi sự đan xen của các bước thực hiện trong quá trình xử lý tính toán hoặc thực hiện song song bằng cách gắn mỗi phép tính toán cho một bộ vi xử lý khác ở trong hệ thống.

Một trong những điểm khó khăn khi xây dựng các chương trình tương tranh đó là phải đảm bảo được đúng trình tự tương tác giữa các phép toán của các quy trình khác nhau .Có một số phương pháp được sử dụng để xây dựng được chương trình tương tranh , ví dụ như thực hiện xử lý tính toán như là một quy trình của hệ thông , hoặc thực hiện các quá trình tính toán như một tập hợp các chủ đề trong một hệ thống các quy trình điều hành đơn lẻ.

Trong lập trình tương tranh, một số dòng lệnh có thểđược thựchiện đồng thời.Mỗi dòng lệnh là một chương trình tuần tự, ngoại trừ một thực tế là các dònglệnh có thể giao tiếp và gây trở ngại lẫn nhau. Mỗi một dòng lênh là một luồng (thread), các chương trình tuần tự còn gọi là chương trình đơn luồng, còn chươngtrình viết theo phương pháp tương tranh là chương trình đa luồng.

Các ngôn ngữ lập trình tương tranh thường được sử dụng là : *Java, C#, Erlang , Limbo*.

Các đặc trưng cơ bản

- Về nguyên tắc và ý tưởng : thực hiện đồng thời nhiều chương trình cùng lúc.

- Cấu trúc của lập trình tương tranh có thể bao gồm đa luồng , hỗ trợ tính toán phân tán , truyền thông và chia sẻ tài nguyên.

- Hoàn thành được một số lượng nhiều công việc cùng một lúc.

- Người dùng có thể tương tác với các ứng dụng khi tiến trình đang thực thi.

- Những tiến trình chạy thời gian dài làm trì hoãn các tiến trình ngắn.

- Các chương trình phức tạp được thực hiện tốt hơn trên các bộ xử l{ đaluồng.

- Những tiến trình đòi hỏi điều kiện tiên quyết có thể tạm dừng và đợi đếnkhi đáp ứng được để tiếp tục xử lý

6- Lập trình phân tán – Distributed programming

Khái niệm

Lập trình phân tán (Distributed Programming) là tạo ra các ứng dụng hoặc hệ thống mà ứng dụng và hệ thống đó được sắp đặt ở nhiều máy khác nhau theo một nguyên tắc do người thiết kế chỉ ra.

Lập trình phân tán là một ngôn ngữ lập trình chú trọng vào thiết kế các hệ thống phân tán có các tính chất : chịu lỗi, xuyên dụng. mở rộng được , mở.Lập trình phân tán là kết quả của việc dùng máy tính để tạo nên các mạng máy tính.

Lập trình phân tán là một dạng của lập trình song song (tính toán song song). Lập trình song song tạo ra mối liên hệ giữa máy tính và các đơn vị tính toán, khiến chúng hoạt động đồng thời đối với một vấn đề cụ thể (dự báo thời tiết chẳnghạn). Các đơn vị tính toán có thể đặt rất gần nhau hoặc tách rời nhau. Khi các đơnvị tính toán được đặt tách rời, ta gọi đó là lập trình phân tán. Với mô hình lậptrình này, các đơn vị tính toán thường rất khác nhau, cũng như sự khác nhau giữa các hệ điều hành và thiết lập mạng máy tính. Những yếu tố đó khiến cho việc lậptrình các tính toán của máy tính trở nên tương đối phức tạp và khó khăn.Lập trình mạng phân tán, thường có 2 khái niệm chính: peer to peer và client -server... peer to peer là lập trình ngang hàng giữa 2 máy tính... còn lập trình client-server là lập trình chophép n máy client kết nối tớim máy server -đây cũng làmô hình chúng ta sẽ gặp nhiều trong thực tế.

Các đặc trưng cơ bản

Có thể chia lập trình mạng thành 3 cấp độ, căn cứ theo mức độ thân thiện với nhà phát triển và khả năng triển khai:

- Mức thấp nhất là lập trình sử dụng socket-có thể truyền từng byte, từng streamlên trên mạng, ứng dụng này thường gặp trong các ứng dụng mạng cỡ nhỏ (vd:trong 2 máy tính, trong 1 mạng LAN...).

- Mức cao hơn là lập trình sử dụng, triển khai ứng dụng theo mô hình dịch vụ (service) - quản lí mạng theo mô hình: kết nối, tạo tham chiếu client trên server vàtrả về, gọi hàm và truyền dữ liệu thông qua proxy của đối tượng trả về từ server(vd: RMI trên Java hay Remoting trên.NET...), mô hình này thường gặp trong cácứng dụng mạng cỡ trung bình và lớn, đòi hỏi tính trình bày cao (vd: các ứng dụng mạng đòi hỏi kết nối nhiều-liên tục và sử dụng diện rộng như các dịch vụchat, game online...), ưu điểm chính của mô hình này là tính performance cao,bảo mật, nhưng nhược điểm là tính đa nền (multiplatform) và tính khả chuyển chưa cao (server và client phải cùng 1 công nghệ phát triển.)

- Mức cao nhất là triển khai ứng dụng theo mô hình triển khai trên web, điển hình nhất ta có thể thấy làcác web application và web service, mới nhất giờ có công nghệ WCF của M$ (tổng hợp tất cả trong 1)... mô hình này cho phép triển khai trên diện rộng, phục vụ lượng khách hàng lớn... nhưng cũng có nhược điểm là tính biểu diễn chưa cao, bù lai, nó có khả năng mềm dẻo (flexibility) cao (server và client không cần sử dụng chung một công nghệ). Chúng ta thường thấynhững ứng dụng loại này trong các ứng dụng được cung cấp từ một nhà cung cấpnào đó (vd: các số liệu chứng khoán, thời tiết …).

7. Lập trình cực hạn – Extreme programming

Khái niệm

Extreme Programing là mô thức lập trình nổi tiếng nhất của phương thức Agile (Agile Method). Nó được hình thành bởi một tập các thực nghiệm đơn giản nhưng phụthuộc lẫn nhau. Những thực nghiệm này làm việc với nhau để hình thành một dạngtổng quát hơn các thành phần của nó.

Lập trình cực hạn (eXtreme Programming viết tắt là XP) là một trong những nhóm các phương pháp phát triển phần mềm một cách linh hoạt. XP sử dụng các nhóm làm việc

kết hợp gồm những người lập trình , khách hàng và các nhà quản trị để phát triển phần mềm. Một chương trình chạy được là thước đo đầu tiên của tiến trình theo XP.

XP có thể phát triển và tồn tại được là do sự hiểu biết ngày một tiến bộ về các vấn đề đang được giải quyết và cũng là công cụ sẵn có cho phép ta thay đổi được cái giá của sự thay đổi (cost-of-change). XP giữ cho cái giá phải trả này ở mức thấp do vậy sẽ thúc đẩy môi trường sản xuất phần mềm.

Các đặc trưng cơ bản

- Liên lạc: Một XP team lớn mạnh dựa trên các kiến thức, sự hiểu biết bài toán và hiểu biết phần mềm được chia sẻ . Các phương pháp giải quyết vấn đề được trao đổi trực tiếp . Những thứ cản trở đến công việc đều được loại bỏ .

- Đơn giản hoá**:** công việc cần giảm tối thiểu độ phức tạp để đạt được hiệu quả cao nhất

- Thông tin phản hồi thường các đội làm dự án và khách hàng của họ không nhận ra những vấn đề rắc rối cho tới khi sắp bàn giao sản phẩm. Nhưng các XP teams thường xuyên lấy feedback – trong quá trình làm việc, test, bàn giao sản phẩm … Khi đó sẽ điều khiển được các vấn đề phát sinh.

- Thế mạnh**:** các đội làm phần mềm thành công cần phải kiểm soát được ngay cả khi xuất hiện các lỗi . XP đưa ra 12 phương án thực hành, và điểm mạnh của XP chính là đã kết hợp được các phương án này lại . Mỗi một phương án tuy đơn giản nhưng rất cần thiết phải nắm vững, sẽ góp phần làm giảm bớt đáng kể cái giá của sự thay đổi .

Phần 2: **MÔ THỨC LẬP TRÌNH TRỰC QUAN (Visual Paradigm)**

Sự phát triển không ngừng của Tin học gắn liến với sự phát triên của các loại ngôn ngữ lập trình.Có rất nhiều loại ngôn ngữ lập trình xuất hiện nhưng hầu hết các loại ngôn ngữ lập trình đều tương tác với máy tính ở chế độ văn bản (text), chỉ có một số ít ngôn ngữ lập trình tương tác với máy tính ở chế độ đồ họa , điển hình là ngôn ngữ lập trình trực quan. Sau đây , em xin trình bày , phân tích về loại ngôn ngữ này.

1. Lịch sử ra đời của ngôn ngữ lập trình trực quan

Ngôn ngữ lập trình trực quan xuất hiện đầu tiên vào năm 1963 , đó chính là hệ thống Sketchpad của Sutherland. Hệ thông cho phép người dùng làm việc với một bút vẽ tạo ra hình ảnh trong không gian 2 chiều (2D) đơn giản như đường thẳng , đường tròn , các xử lý như sao chép và các phép biến đổi hình học khác trên các hình vẽ này.

Cột mốc tiếp theo đánh giá sự phát triển của ngôn ngữ này là năm 1975 với việc công bố luận án tiến sĩ của David Canfield Smith với tiêu đề : “Pygmalion: A Creative Programming Environment”. Công việc của Smith đánh dấu sự khởi đầu của một loạt các nghiên cứu trong lĩnh vực này cho đến ngày hôm nay**.** Những kết quả của Smith cùng nhiều thành tựu nghiên cứu khác sau này đã góp phần hình thành nên tiêu chuẩn về lý thuyết biểu tượng (icon theory).

2. Phân loại các ngôn ngữ lập trình trực quan

Lập trình trực quan phân thành các nhóm chính sau đây:

- Các ngôn ngữ trực quan thuần túy
- Những hệ thống lai giữa trực quan và văn bản
- Những hệ thống lập trình bằng ví dụ
- Những hệ thống rằng buộc đối tượng
- Những hệ thống được xây dựng dựa trên các biểu mẫu

Việc phân loại trên không loại trừ lẫn nhau. Do đó, có nhiều ngôn ngữ có thể thuộc nhóm này nhưng cũng có thể thuộc nhóm khác.

Trong các nhóm trên , nhóm các ngôn ngữ trực quan thuần túy là quan trọng nhất. Các ngôn ngữ này được đặc trưng bởi việc dựa hoàn toàn trên các kỹ thuật trực quan thông qua tiến trình lập trình. Người lập trình vận dụng các biểu tượng hoặc các sự biểu diễn đồ hoạ khác để khởi tạo một chương trình rồi tiếp theo sẽ gỡ lỗi và thực thi trong cùng một môi trường trực quan. Chương trình là được biên dịch trực tiếp từ sự trình bày trực quan của nó và không bao giờ dịch đến một ngôn ngữ trung gian dựa trên chế độ văn bản (interim textbased language). Những ví dụ của các hệ thống thuần tuý trực quan như VIPR, Prograph và PICT.

3. Lý thuyết các ngôn ngữ lập trình

Để có thể đi sâu tìm hiểu các vấn đề trong lập trình trực quan , chúng ta nên làm quen với các định nghĩa sau:

- Icon (biểu tượng, hình tượng): Một hình tượng với sự trình bày kép của một phần logic (ý nghĩa) và một phần vật lý (hình ảnh).

- Iconic system (hệ thống có tính biểu tượng): một tập hợp có cấu trúc của những biểu tượng có liên quan.

- Iconic sẹntence (câu biểu tượng/trực quan): Một sự sắp xếp không gian của các biểu tượng từ hệ thống biểu tượng.

- Visual language (Ngôn ngữ trực quan):Một tập các câu trực quan được cấu trúc với cú pháp và ngữ nghĩa cho trước.

- Syntactics analysis (phân tích cú pháp): Một sự phân tích của một câu trực quan để xác lập cấu trúc cơ sở.

- Semantic analysis (phân tích ngữ nghĩa): Một sự phân tích của một câu trực quan để xác định ngữ nghĩa cơ sở.

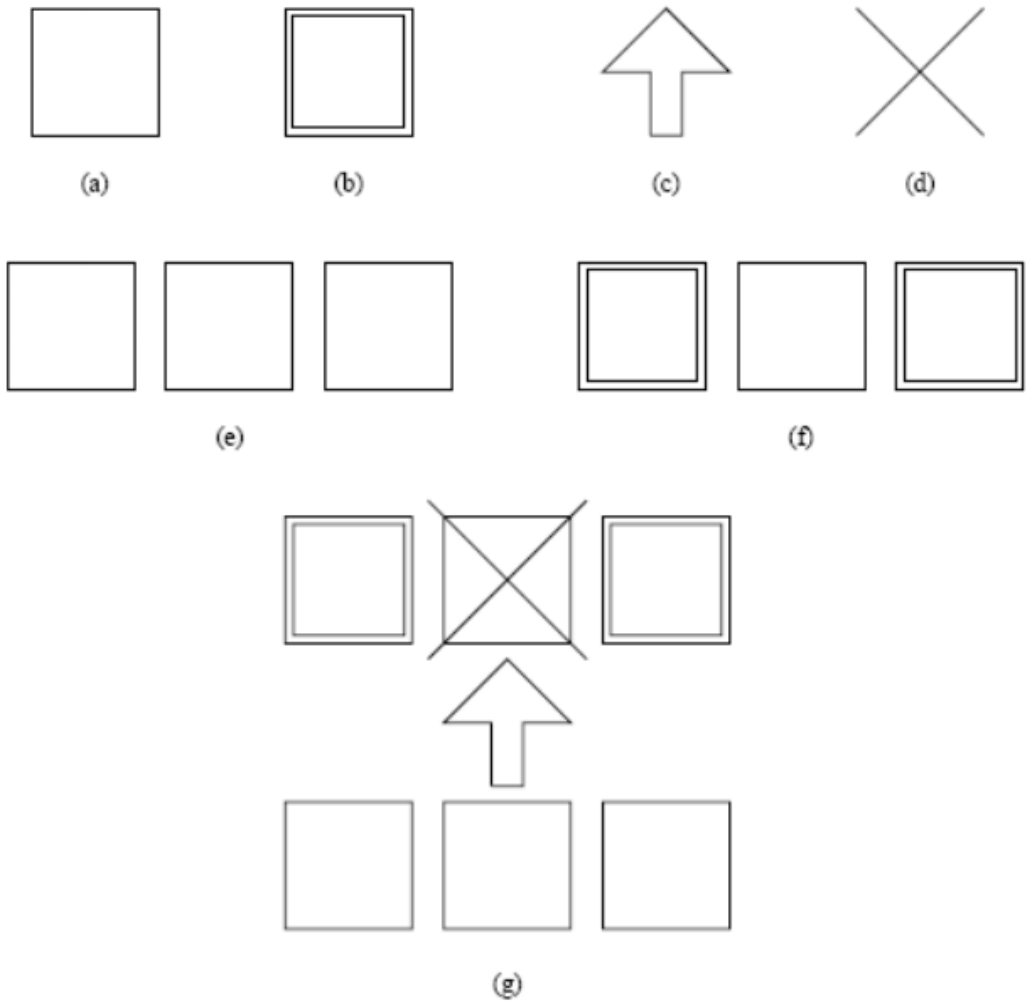
a. Hình thức của ngôn ngữ lập trình trực quan

Một sự sắp xếp không gian của các biểu tượng sẽ tạo thành một câu trực quan. Nó là bả nsao hai chiều của sự sắp xếp một chiều của các dấu hiệu trong các ngôn ngữ lập trình qui ước (trong chế độ text). Trong những ngôn ngữ này, một chương trình là được diễn giải như một xâu ký tự trong đó ký hiệu kết thúc là móc nối đến mẫu một câu mà cấu trúc và ngữ nghĩa củanó là được khám phá bởi sự phân tích cú pháp và ngữ nghĩa theo thứ tự định sẵn. Theo cáchđó, qui tắc xây dựng là tiềm ẩn trong ngôn ngữ và không

cần giải thích như một phần của ngôn ngữ đặc tả. Ngược lại, trong những ngôn ngữ lập trình trực quan chúng ta phân biệt 3 nguyên tắc để sắp xếp các biểu tượng: sự móc nối theo chiều ngang (biểu hiện bởi ký hiệu &), móc nối theo chiều đứng (biểu hiện bởi ký hiệu ^) và không gian bao phủ (biểu hiện bởi ký hiệu +).

Trong việc định dạng các ngôn ngữ lập trình trực quan, nó là lựa chọn để phân biệt các biểu tượng tiến trình từ các biểu tượng đối tượng. Sự tính toán các biểu thức trước, sau đó có thể phân nhỏ đến các biểu tượng đối tượng sơ cấp và các biểu tượng đối tượng phức hợp. Những biểu tượng đối tượng cơ sở cho biết các đối tượng sơ cấp trong ngôn ngữ, nhưng ngược lại cácđối tượng phức hợp cho biết các đối tượng được định dạng bởi một sự sắp xếp không gian củacác biểu tượng đối tượng sơ cấp.

Tóm lại, thuật ngữ biểu tượng sơ cấp (elementary icons) là để tham chiếu đến cả hai loại: biểu tượng tiến trình và biểu tượng đối tượng sơ cấp và biểu hiện bởi những biểu tượng của chúng đó là nguyên gốc trong ngôn ngữ. Từ một bức tranh (hoặc biểu tượng trong trường hợp này) là có hàng ngàn cách biểu đạt, chúng ta cố gắng minh hoạ tất cả những ý niệm trên đây trong hình vẽ sau:



Hình 1: Hệ thống biểu tượng của Heidelberg

Hình vẽ này giới thiệu một vài biểu tượng từ bộ biểu tượng Heidelberg. Những biểu tượngsơ cấp gồm (a) là một ký tự và (b) là một ký tự được chọn; các biểu tượng tiến trình: (c) là hoạt động chèn và (d) là hoạt động xoá; các biểu tượng phức hợp: (e) là một xâu (kết hợp cácký tự) và (f) là xâu được lựa chọn; (g) câu trực quan biểu thị sự thay thế của một xâu con trongmột xâu nào đó.

Một môn ngữ lập trình trực quan là được đặc tả bởi một bộ ba (ID,G0,B), ở đây ID là từ điển biểu tượng, G0 là một ngữ pháp và B là một cơ sở tri thức cho một lĩnh vực đặc biệt. Từ điển biểu tượng là một tập hợp các biểu tượng tổng quát và mỗi một phần tử là được trình bàybởi một cặp (Xm,Xi), với phần lô-gíc Xm (nghĩa) và phần vật lý Xi (hình ảnh). Ngữ pháp G0 chỉ định cách thức làm thế nào để những biểu tượng đối tượng phức

hợp có thể được xây dựngtừ các biểu tượng sơ cấp bằng việc sử dụng những toán tử sắp xếp không gian.

Cơ sở tri thức B chứa những thông tin của một lĩnh vực đặc biệt cần thiết cho việc xây dựng ngữ nghĩa của một câu trực quan cho trước. Nó chứa thông tin về các tên sự kiện, những quanhệ thuộc về khái niệm, tên của các đối tượng kết quả và những tham chiếu đến các đối tượng

b. Phân tích các ngôn ngữ lập trình trực quan

Danh sách không gian của các cách tiếp cận :

Picture –procesing grammars (ngữ pháp xử lý ảnh): khám phá ra cấu trúc của câu trực quan bởi trật tự của các điểm ảnh riêng lẻ đến các phần tử trực quan có thể nhận diện được (đường thẳng, hộp, cung tròn…).

Precedence grammars (ngữ pháp thứ tự): phân tích cú pháp của các câu trực quan được xây dựng từ các biểu tượng sơ cấp các toán tử biểu tượng.

Context-free and context-dependent grammars (ngữ pháp phụ thuộc ngữ cảnh và ngữ pháp phi ngữ cảnh): dùng để xác định tổ hợp của các câu trực quan trong việc sử dụng những hình thức quen thuộc và nhiều phương pháp tiểu chuẩn của sự phân tích.

Graph grammars (ngữ pháp đồ thị): đặc tả các ngôn ngữ trực quan

4. Những vấn đề của ngôn ngữ trực quan

a. Luồng điều khiển (Control Flow)

Ngôn ngữ trực quan gắn liền với hai khái niệm về luồng điều khiển trong các chương trình là: mệnh lệnh và khai báo:

-Với cách tiếp cận mệnh lệnh : một ngôn ngữ lập trình trực quan dựa trên một hoặc nhiều luồng điều khiển hoặc biểu đồ luồng dữ liệu mà ở đó nó cho biết làm thế nào xâu chuỗi các luồng điều khiển thông qua chương trình.

-Với cách tiếp cận khai báo : người ta chỉ lo lắng mỗi việc là những tính toán nào là sẽ được thực hiện và không cần biết làm thế nào những hoạt động hiện tại được tiến hành.

b. Sự trừu tượng hóa thủ tục (Procedural Abstraction)

Chúng ta phân biệt hai mức độ của việc trừu tượng hoá thủ tục. Những ngôn ngữ lập trình trực quan bậc cao, nghĩa là không thể viết và bảo quản toàn bộ chương trình

trong một ngôn ngữ và chắc chắn có một số lớp chứa những mô-đun không trực quan mà chúng được kết hợp lại bằng cách sử dụng một ngôn ngữ trực quan.

Cách tiếp cận này để lập trình trực quan là có thể được tìm thấy trong rất nhiều các hệ thống phục vụ cho một lĩnh vực đặc biệt, ví dụ như những công cụ bảo quản phần mềm và môi trường trực quan khoa học. Ở phía ngược lại là những ngôn ngữ trực quan mức độ thấp, chúng không cho phép người lập trình móc nối những đơn vị lô-gíc đến các mô-đun thủ tục.

c. Sự trừu tượng hóa dữ liệu (Data Abstraction)

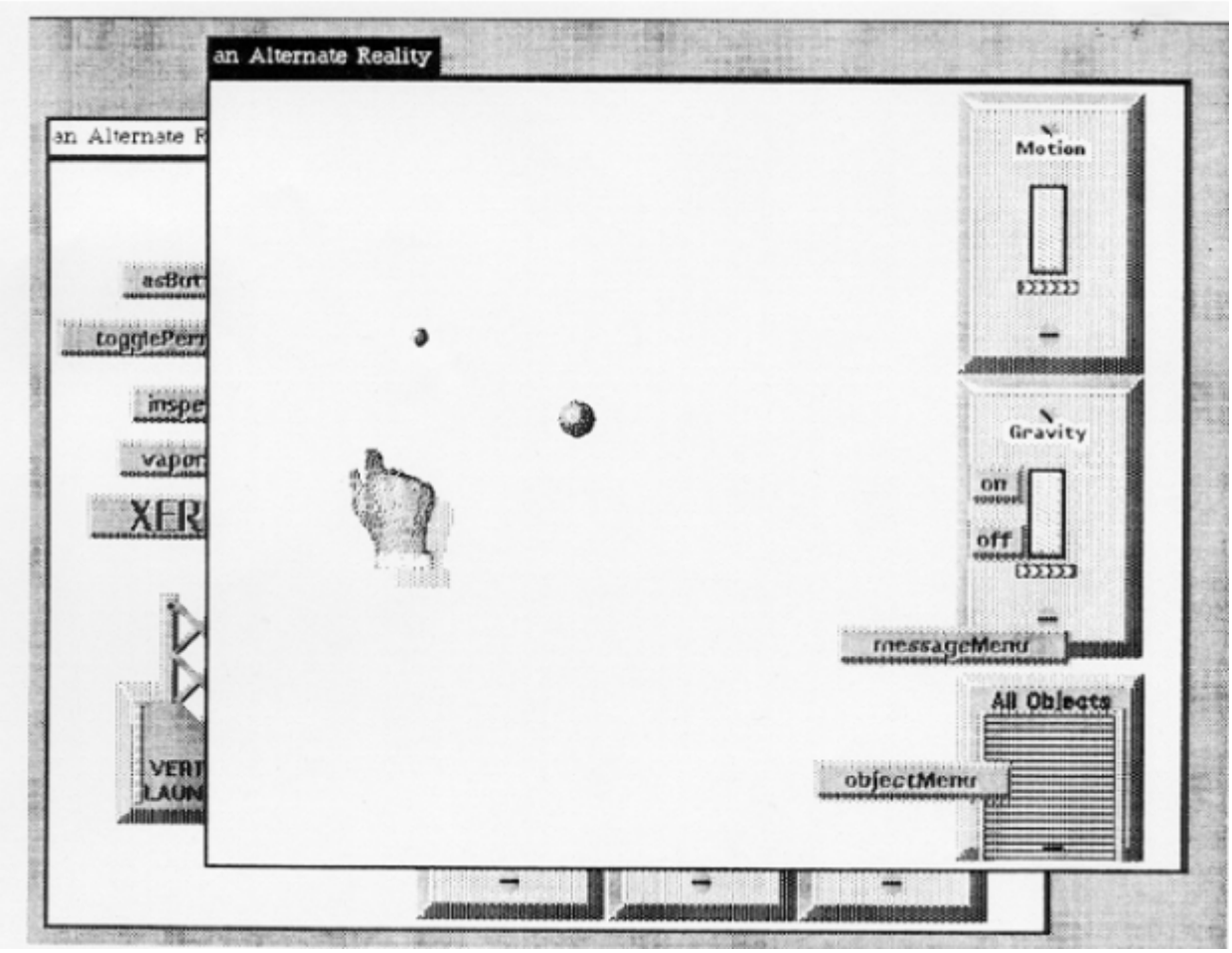
Những phương tiện cho phép trừu tượng hoá dữ liệu là chỉ có thể tìm thấy trong các ngôn ngữ lập trình trực quan đa năng. Khái niệm trừu tượng hoá dữ liệu trong lập trình trực quan là rất giống với khái niệm trừu tượng hoá dữ liệu trong những ngôn ngữ lập trình thông thường với chỉ những yêu cầu trừu tượng các kiểu dữ liệu đã được định nghĩa trực quan (trái với textually), có một sự trình bày trực quan (iconic) và được cung cấp để dành cho cách đối xử tương tác.

5. Các ngôn ngữ lập trình trực quan

a. ARK

Hơn 10 năm kể từ ngày bắt đầu, ngôn ngữ Alternate Reality Kit (ARK) được thiết kế bởi R. Smith tại Xerox PARC, đã ra đời một ngôn ngữ lập trình thuộc họ LẬP TRÌNH TRỰC QUAN để phục vụ cho một lĩnh vực đặc biệt. ARK được phát triển trên Smalltalk-80,cung cấp cho người dùng một môi trường động trong không gian 2 chiều dành để khởi tạo sự mô phỏng tương tác. Hệ thống được dự định để phục vụ cho những người lập trình nghiệp dư tạo ra các mô phỏng và để một nhóm khán giả rộng lớn tương tác với những mô phỏng này.

Những người sử dụng có thể hiệu chỉnh bất kỳ đối tượng nào trong việc sử dụng môi trường, các hộp thông điệp và các nút, nhìn thấy kết quả của những thay đổi của chúng trong thời gian thực. Việc mô phỏng thực hiện trong một “alternate reality” chứa trong một cửa sổ và tất cả được bao quanh một thế giới “siêu thực”. Người lập trình có thể di chuyển bàn tay giữa nguyên bản luân phiên và những đối tượng tác động ra ngoài sự mô phỏng và đến sự siêu thực bất kỳ lúc nào.



Hình 2: *Giao diện mô phỏng của hệ thống ARK*

Người lập trình có thể không chỉ khởi tạo sự mô phỏng bằng việc liên kết nhiều đối tượng và nhiều tương tác nhưng họ cũng có thể phát triển những bộ tương tác mới. Việc lập trình một mô phỏng, ví dụ như mô phỏng hoạt động của các hành tinh trong hình sau: Trong hình trên có phát sinh các đối tượng vật lý, giống như quả bóng .Bằng việc nhấp chuột lên trên nút ***objectMenu*** , người lập trình có thể chọn thuyết minh cho bất kỳ đối tượng nào.Để sắp xếp các thông điệp giữa các đối tượng người lập trình sử dụng nút ***messageMenu*** để các bộ tương tác trả lời bằng cách định vị các nút trên bộ tương tác và nhấn nó với bàn tay.

b. Prograph

Ngôn ngữ prograph có thể coi là ngôn ngữ thành công nhất trong số các ngôn ngữ trực quan đa năng (general-purpose visual languages). Progragh được nghiên cứu từ năm

1982 tại trường Đại học kỹ thuật Nova Scotia.Có nhiều phiên bản đã được đưa vào sử dụng , trong số đó được dùng phổ biến nhất là Progragh/CPX .

Prograph là ngôn ngữ hướng đối tượng trực quan. Nó móc nối những khái niệm về các lớp, các đối tượng với một cơ chế đặc tả dòng dữ liệu trực quan rất mạnh. Prograph còn là một ngôn ngữ mệnh lệnh, nó cung cấp những điều khiển rõ ràng trên trật tự các tính toán

Prograph cho phép người lập trình làm việc trên cả hai mức độ thấp và cao, cho phép họ thiết kế và bảo trì nhiều hoặc ít phần mềm phức tạp. Những sự tính toán nguyên thuỷ như các phép toán số học, các lời gọi các method... là đã được móc nối đến các phần thân của phương pháp biểu mẫu bởi các biểu đồ dòng dữ liệu. Các phương thức (methods) là được tổ chức thành các lớp. Ngoài ra, Prograph còn cung cấp cho người lập trình cơ chế gọi những đối tượng mà nó có thể là được lưu trữ trong một cơ sở dữ liệu Prograph giữa các lời đề nghị khác nhau của chương trình.

c. Form/3

Forms/3 là một ngôn ngữ lập trình khác thuộc dạng ngôn ngữ lập trực quan hướng đối tượng đa năng. Đặc điểm nổi bật của nó là trừu tượng hoá dữ liệu. Tuy nhiên, không giống như Prograph, nó không mang tính kế thừa và sự phân tích các thông điệp là có được hỗ trợ.

Forms/3 phỏng theo cách tổ chức bảng tính với các ô và công thức để trình bày dữ liệu và thứ tự tính toán. Một đặc trưng của Forms/3 là các ô ở đây đã được tổ chức thành một nhóm gọi được là form, một cơ chế trừu tượng hoá dữ liệu cơ bản. Một form có thể trình bày một ảnh cho trước (còn gọi là một biểu tượng) và nó có thể được minh hoạ cho một đối tượng. Theo nghĩa này, một form tương ứng đến một đối tượng nguyên mẫu trong các ngôn ngữ hướng đối tượng nguyên mẫu cơ bản (prototype-based object oriented languages).

Trong Forms/3, dữ liệu (values) và sự tính toán (formulas) là một cặp thống nhất. Mỗi một đối tượng được đặt trong một ô và đã được định nghĩa với khai báo để sử dụng một công thức. Các đối tượng có thể chỉ được khởi tạo theo các công thức và mỗi một công thức sản sinh một đối tượng giống như một kết quả của việc đánh giá nó. Những công thức cung cấp một sự dễ dàng để yêu cầu các kết quả từ các đối tượng khác và khởi tạo những đối tượng mới: điều này không cần sự phân tích các thông điệp.

Lời kết

Bên cạnh sự đa dạng của ngôn ngữ lập trình, nhiều mô thức lập trình được ra đời đáp ứng các nhu cầu về mô hình lập trình tổng quát cho các phương thức lập trình riêng biệt. Mỗi mô thức lập trình lại có những đặc trưng riêng về cách sử dụng cũng như tính trừu tượng hóa đối với các quá trình xác định.

Về mô thức lập trình trực quan, những ngôn ngữ trong không gian ba chiều không chỉ nhắm đến các vấn đề thể hiện một khối lượng lớn thông tin trên một màn hình nhỏ mà còn mô tả sự phối hợp giữa các ngôn ngữ lập trình, đồ hoạ máy tính và tương tác người máy mà ở đó nó đã được đạt được những thành tựu đáng khích lệ với một sự khởi đầu tốt đẹp.

Báo cáo này là sự tổng hợp, lược dịch và chỉnh sửa lại các bài viết đã có trên internet, sách in và một số tài liệu khác. Sai sót về mặt nội dung cũng như khiếm khuyết trong cách trình bày chắc chắn không thể tránh khỏi, em rất mong nhận được sự thông cảm và giúp đỡ của Thầy để báo cáo được hoàn thiện hơn!

Tài liệu tham khảo

1. Slide bài giảng Kĩ thuật lập trình của thầy Vũ Đức Vượng, Viện Công nghệ thông tin – Truyền thông, Đại học Bách Khoa Hà Nội
2. https://vi.scribd.com
3. https://www.wikipedia.org
4. http://people.cs.clemson.edu