

HỆ THỐNG ĐIỀU KHIỂN PHÂN TÁN

Lecture Notes

(Chưa cập nhật từ 8/2003)

TS. Hoàng Minh Sơn
BỘ MÔN ĐIỀU KHIỂN TỰ ĐỘNG, KHOA ĐIỆN
ĐẠI HỌC BÁCH KHOA HÀ NỘI

MỤC LỤC

1	NHẬP MÔN	5
1.1	Phạm vi đề cập	5
1.2	Nội dung chương trình	5
1.3	Yêu cầu kiến thức cơ sở	5
1.4	Tổng quan các giải pháp điều khiển	6
1.4.1	Đặc trưng các lĩnh vực ứng dụng điều khiển	6
1.4.2	Các hệ thống điều khiển công nghiệp	6
2	CẤU TRÚC CÁC HỆ THỐNG ĐIỀU KHIỂN VÀ GIÁM SÁT	8
2.1	Cấu trúc và các thành phần cơ bản	8
2.2	Mô hình phân cấp	9
2.2.1	Cấp chấp hành	10
2.2.2	Cấp điều khiển	10
2.2.3	Cấp điều khiển giám sát	10
2.3	Cấu trúc điều khiển	11
2.3.1	Điều khiển tập trung	11
2.3.2	Điều khiển tập trung với vào/ra phân tán	12
2.3.3	Điều khiển phân tán	12
2.3.4	Điều khiển phân tán với vào/ra phân tán	13
3	CÁC THÀNH PHẦN CỦA MỘT HỆ ĐIỀU KHIỂN PHÂN TÁN	15
3.1	Cấu hình cơ bản	15
3.1.1	Trạm điều khiển cục bộ	15
3.1.2	Bus trường và các trạm vào/ra từ xa	17
3.1.3	Trạm vận hành	18
3.1.4	Trạm kỹ thuật và các công cụ phát triển	19
3.1.5	Bus hệ thống	20
3.2	Phân loại các hệ DCS	21
3.2.1	Các hệ DCS truyền thống	21
3.2.2	Các hệ DCS trên nền PLC	22
3.2.3	Các hệ DCS trên nền PC	25
3.3	Các vấn đề kỹ thuật	26
4	XỬ LÝ THỜI GIAN THỰC VÀ XỬ LÝ PHÂN TÁN	27
4.1	Một số khái niệm cơ bản	27
4.1.1	Hệ thống thời gian thực	27
4.1.2	Xử lý thời gian thực	27
4.1.3	Hệ điều hành thời gian thực	28
4.1.4	Xử lý phân tán	29
4.2	Các kiến trúc xử lý phân tán	30
4.3	Cơ chế giao tiếp	31
4.4	Đồng bộ hóa trong xử lý phân tán	32

	2
4.4.1 Đồng bộ hóa các tín hiệu vào/ra	32
4.4.2 Đồng bộ hóa thời gian	32
5 CÔNG NGHỆ ĐỐI TƯỢNG TRONG ĐIỀU KHIỂN PHÂN TÁN	33
5.1 Lập trình hướng đối tượng	33
5.2 Phân tích và thiết kế hướng đối tượng	33
5.2.1 Ngôn ngữ mô hình hóa thống nhất UML	34
5.2.2 Mẫu thiết kế	35
5.2.3 Phần mềm khung	35
5.3 Phần mềm thành phần	36
5.4 Đối tượng phân tán	37
6 KIẾN TRÚC ĐỐI TƯỢNG PHÂN TÁN	38
6.1 Yêu cầu chung	38
6.2 Các mẫu thiết kế	38
6.3 Giới thiệu chuẩn CORBA	39
6.4 Giới thiệu chuẩn COM/DCOM	40
6.4.1 Giao diện	41
6.4.2 Đối tượng COM	41
6.4.3 Giao tiếp giữa client và object	44
6.4.4 Ngôn ngữ mô tả giao diện	46
6.4.5 Mô hình đối tượng thành phần phân tán DCOM	46
7 CÁC MÔ HÌNH ỨNG DỤNG ĐIỀU KHIỂN PHÂN TÁN	48
7.1 IEC-61131	48
7.1.1 Mô hình phần mềm	48
7.1.2 Mô hình giao tiếp	49
7.2 IEC-61499	51
7.2.1 Mô hình hệ thống	51
7.2.2 Mô hình thiết bị	52
7.2.3 Mô hình tài nguyên	52
7.2.4 Mô hình ứng dụng	53
7.2.5 Mô hình khối chức năng	54
7.2.6 Mô hình phân tán	56
7.2.7 Mô hình quản lý	56
7.2.8 Mô hình trạng thái hoạt động	56
8 MỘT SỐ CHUẨN GIAO TIẾP CÔNG NGHIỆP	58
8.1 MMS	58
8.2 IEC-61131-5	60
8.2.1 Mô hình giao tiếp mạng	60
8.2.2 Dịch vụ giao tiếp	61
8.2.3 Các khối chức năng giao tiếp	62
8.3 OPC	63
8.3.1 Tổng quan về kiến trúc OPC	63

	3
8.3.2 OPC Custom Interfaces	65
8.3.3 OPC Automation Interface	66
8.4 Ngôn ngữ đánh dấu khả mở XML	67
8.4.1 Giới thiệu chung	67
8.4.2 Ứng dụng XML trong phần mềm khung iPC	68
9 MÔ TẢ HỆ THỐNG ĐIỀU KHIỂN PHÂN TÁN	70
9.1 Các phương pháp mô tả đồ họa	70
9.2 Lưu đồ P&ID	71
9.2.1 Chuẩn ISA S5.1	71
9.2.2 Chuẩn ISA S5.3	75
9.3 Mô hình hóa hướng đối tượng	77
10 LẬP TRÌNH ĐIỀU KHIỂN PHÂN TÁN	78
10.1 Lập trình theo chuẩn IEC 61131-3	78
10.1.1 Kiểu dữ liệu	79
10.1.2 Tổ chức chương trình	81
10.1.3 Ngôn ngữ FBD	83
10.1.4 Ngôn ngữ ST	84
10.1.5 Ngôn ngữ SFC	85
10.2 Lập trình với ngôn ngữ bậc cao	85
11 CHỨC NĂNG ĐIỀU KHIỂN GIÁM SÁT	87
11.1 Giới thiệu chung về các hệ điều khiển giám sát	87
11.1.1 Các thành phần chức năng cơ bản	88
11.1.2 Công cụ phần mềm SCADA/HMI	89
11.2 Xây dựng cấu trúc hệ thống	91
11.3 Thiết kế giao diện người-máy	92
11.3.1 Yêu cầu chung	92
11.3.2 Các phương pháp giao tiếp người-máy	92
11.3.3 Thiết kế cấu trúc màn hình	92
11.3.4 Các nguyên tắc thiết kế	93
12 TÍNH SẴN SÀNG VÀ ĐỘ TIN CẬY CỦA CÁC HỆ ĐKPT	94
12.1 Đặt vấn đề	94
12.2 Cơ chế dự phòng	94
12.3 Cơ chế an toàn	95
12.4 Cơ chế khởi động lại sau sự cố	95
12.5 Bảo mật	95
12.6 Bảo trì	95
13 ĐÁNH GIÁ VÀ LỰA CHỌN GIẢI PHÁP ĐIỀU KHIỂN PHÂN TÁN	97
13.1 Đánh giá và lựa chọn các sản phẩm DCS tích hợp trọn vẹn	97
13.1.1 Phạm vi chức năng	97
13.1.2 Cấu trúc hệ thống và các thiết bị thành phần	97
13.1.3 Tính năng mở	97

	4
13.1.4 Phát triển hệ thống	97
13.1.5 Độ tin cậy và tính sẵn sàng	98
13.1.6 Giá thành, chi phí	98
13.2 So sánh giải pháp DCS tích hợp trọn vẹn với các giải pháp khác	98
14 GIỚI THIỆU MỘT SỐ HỆ ĐIỀU KHIỂN PHÂN TÁN TIÊU BIỂU	100
14.1 PCS7 của Siemens	100
14.2 PlantScape của Honeywell	100
14.3 DeltaV của Fisher Rosermount	100
14.4 Centum CS1000/CS3000 của Yokogawa	100
14.5 AdvantOCS của ABB	100
15 MỘT SỐ HƯỚNG NGHIÊN CỨU VÀ ỨNG DỤNG	101
15.1 Trí tuệ nhân tạo phân tán	101
15.2 Điều khiển và giám sát các hệ thống giao thông	102
15.2.1 Đặt vấn đề	102
15.2.2 Mô hình hệ thống điều khiển đèn tín hiệu giao thông bằng công nghệ Agent	102
15.3 Điều khiển và giám sát các hệ thống sản xuất và cung cấp điện	104
TÀI LIỆU THAM KHẢO	105

1 NHẬP MÔN

1.1 Phạm vi đề cập

Phạm vi đề cập của môn *Hệ thống điều khiển phân tán* là các hệ thống tự động hoá hiện đại có cấu trúc phân tán trong công nghiệp cũng như trong nhiều lĩnh vực khác. Môn học được xây dựng trên cơ sở ứng dụng các tiến bộ mới nhất của kỹ thuật điều khiển, kỹ thuật truyền thông công nghiệp, công nghệ phần mềm vào trong các hệ thống điều khiển và giám sát.

Mục đích của môn học cho sinh viên làm quen với cấu trúc và các thiết bị phần cứng cũng như các thành phần phần mềm của các hệ thống điều khiển và giám sát hiện đại, nắm được các nguyên tắc và phương pháp cơ bản cho hướng giải quyết những bài toán thường được đặt ra trong thực tế như thiết kế cấu trúc hệ thống, tích hợp hệ thống, đưa vào vận hành và chẩn đoán hệ thống. Bên cạnh đó, môn học đưa ra các hướng nghiên cứu lý thuyết và ứng dụng mới, tạo cơ sở cho các sinh viên muốn tiếp tục học và nghiên cứu ở các bậc sau đại học.

1.2 Nội dung chương trình

Nội dung bài giảng bao gồm các chủ đề chính sau:

- Cấu trúc các hệ thống điều khiển và giám sát: Mô hình phân cấp, các thành phần chức năng cơ bản, mô tả hệ thống
- Cơ sở tin học: Xử lý phân tán, công nghệ hướng đối tượng, phần mềm thành phần
- Các hệ thống điều khiển phân tán truyền thống (DCS): Cấu trúc hệ thống, các thành phần hệ thống, phương pháp phát triển hệ thống, giới thiệu một số hệ DCS tiêu biểu.
- Các hệ thống điều khiển phân tán trên nền PLC (PLC-based DCS)
- Các hệ thống điều khiển phân tán trên nền PC (PC-based DCS)
- Hệ thống điều khiển giám sát và thu thập dữ liệu (SCADA): Cấu trúc hệ thống, công cụ phần mềm, thiết kế giao diện người-máy
- Các chuẩn giao tiếp công nghiệp: MMS, OPC, XML
- Các hướng nghiên cứu và ứng dụng.

1.3 Yêu cầu kiến thức cơ sở

Phần lớn nội dung các bài giảng mang tính chất tổng hợp, liên môn, giành cho sinh viên năm cuối. Bên cạnh các môn cơ sở chuyên ngành, yêu cầu học viên phải nắm vững kiến thức cơ bản trong các môn học sau:

- Điều khiển số
- Mạng truyền thông công nghiệp
- Kỹ thuật lập trình C++ (hướng đối tượng)

1.4 Tổng quan các giải pháp điều khiển

1.4.1 Đặc trưng các lĩnh vực ứng dụng điều khiển

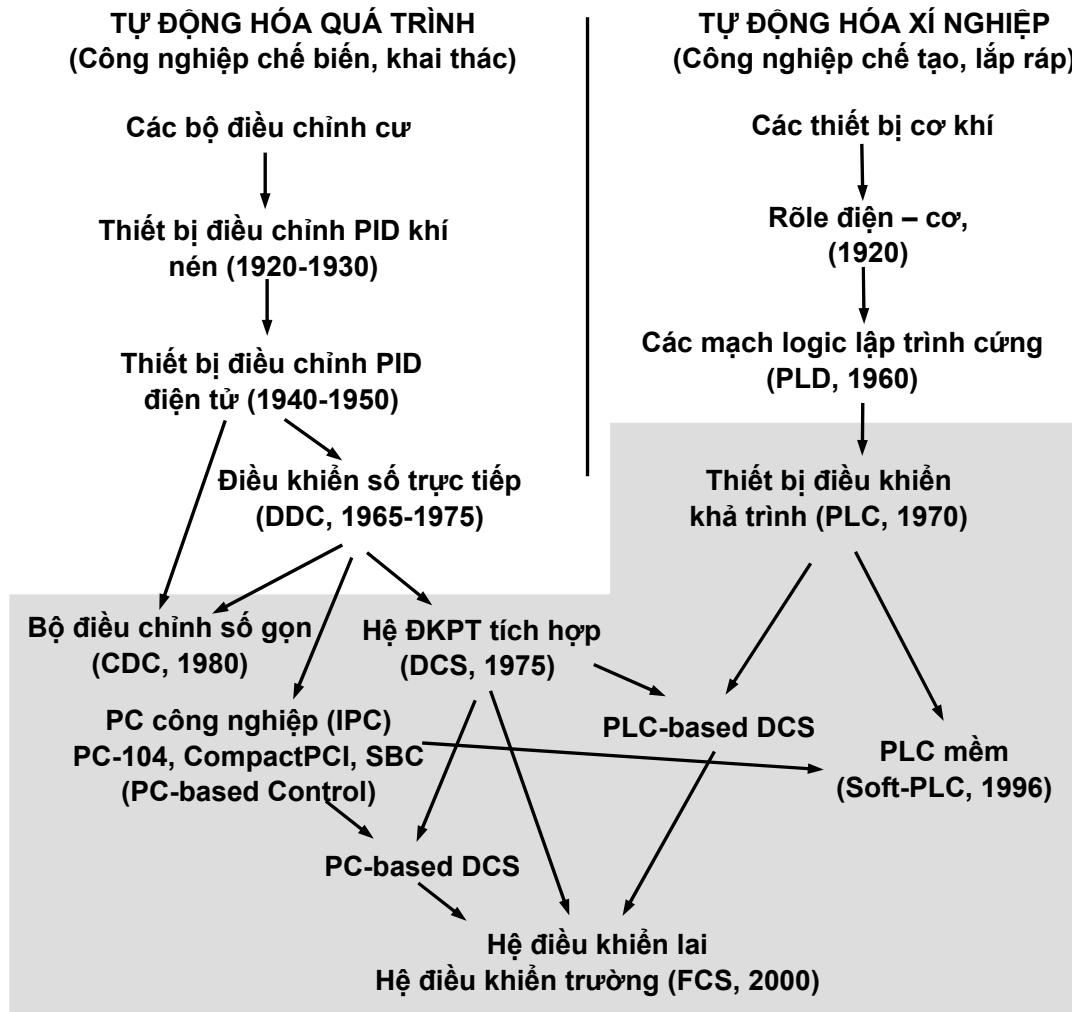
Khi xây dựng một giải pháp điều khiển, ta phải quan tâm tới qui mô và đặc thù của lĩnh vực ứng dụng. Một vài lĩnh vực ứng dụng tiêu biểu và các giải pháp điều khiển đặc thù tương ứng được tóm tắt dưới đây.

- Điều khiển các thiết bị và máy móc đơn lẻ (công nghiệp và gia dụng): Các máy móc, thiết bị được sản xuất hàng loạt, vì vậy yêu cầu đầu tư cho giải pháp điều khiển phải thật tiết kiệm (chương trình nhỏ, tốn ít bộ nhớ). Các bài toán điều khiển có thể rất khác nhau, từ điều khiển logic tới điều khiển phản hồi, điều khiển chuyển động, điều khiển mờ,... Các giải pháp điều khiển tiêu biểu là điều khiển nhúng (μP , μC), CNC, PLC,...
- Tự động hóa công nghiệp, được chia ra hai lĩnh vực:
- Công nghiệp chế biến, khai thác: Các bài toán điều khiển tiêu biểu là điều khiển quá trình (*process control*), điều khiển trình tự (*sequence control*), bên cạnh điều khiển logic. Các thiết bị được dùng phổ biến là PLC, DCS, (I)PC, Compact Digital Controllers.
- Công nghiệp chế tạo, lắp ráp: Các bài toán điều khiển tiêu biểu là điều khiển logic, điều khiển chuyển động, điều khiển sự kiện rời rạc. Các thiết bị được dùng chủ yếu là PLC, CNC, PC. Nay các hệ DCS cũng tìm được một số ứng dụng trong lĩnh vực này.
- Điều khiển các hệ thống giao thông, vận tải: Đặc thù là các bài toán điều khiển logic, điều khiển sự kiện rời rạc. Các thiết bị được dùng là PLC, DCS, PC, μP , μC ,...
- Điều khiển các hệ thống phân phối năng lượng (dầu khí, gas, điện): Kết hợp giữa các bài toán điều khiển quá trình với điều khiển sự kiện rời rạc, điều khiển logic, sử dụng PLC, DCS, IPC,...
- Tự động hóa tòa nhà: Role, PLC, μp , μC ,...
- Điều khiển và giám sát các hệ thống quốc phòng: IPC, μP , μC , DSP và các thiết bị đặc chủng khác.
- Điều khiển và giám sát các hệ thống thủy lợi, môi trường: PLC, IPC, ...
- ...

1.4.2 Các hệ thống điều khiển công nghiệp

Chương trình học đặt trọng tâm vào các giải pháp điều khiển công nghiệp, chia làm hai lĩnh vực ứng dụng cơ bản:

- Công nghiệp chế biến, khai thác (Process Industry): Dầu khí, hóa dầu, hóa mỹ phẩm, dược phẩm, xi măng, giấy, ...
- Công nghiệp chế tạo, lắp ráp (Manufacturing Industry): Công nghiệp ô tô, máy công cụ, công nghiệp điện tử, vi điện tử, thiết bị dân dụng,...

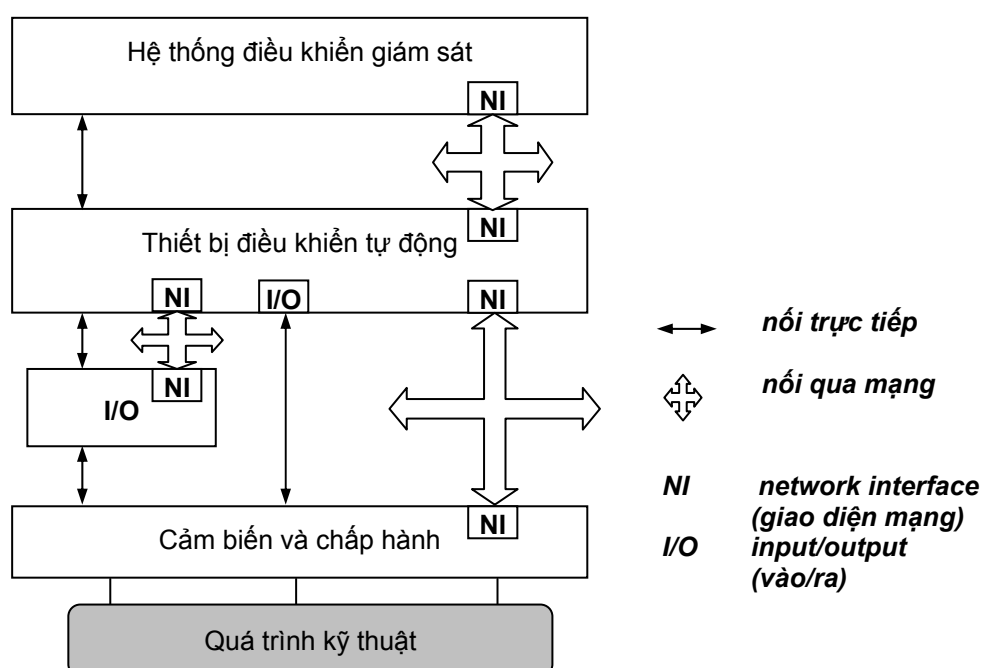


Hình 1-1: Lịch sử phát triển các giải pháp điều khiển

2 CẤU TRÚC CÁC HỆ THỐNG ĐIỀU KHIỂN VÀ GIÁM SÁT

2.1 Cấu trúc và các thành phần cơ bản

Các thành phần cơ bản của một hệ thống điều khiển và giám sát quá trình được minh họa trên Hình 2-1. Các cảm biến và cơ cấu chấp hành đóng vai trò là giao diện giữa các thiết bị điều khiển với quá trình kỹ thuật. Trong khi đó, hệ thống điều khiển giám sát đóng vai trò giao diện giữa người vận hành và máy. Các thiết bị có thể được ghép nối trực tiếp điểm-điểm, hoặc thông qua mạng truyền thông.



Hình 2-1: Các thành phần cơ bản của một hệ thống điều khiển và giám sát

Tùy theo loại cảm biến, tín hiệu của chúng đưa ra có thể là tín hiệu nhị phân, tín hiệu số hay tín hiệu tương tự theo các chuẩn điện học thông dụng khác nhau (1..10V, 0..5V, 4..20mA, 0..20mA, v.v...). Trước khi có thể xử lý trong máy tính số, các tín hiệu đo cần được chuyển đổi, thích ứng với chuẩn giao diện vào/ra của máy tính. Bên cạnh đó, ta cũng cần các biện pháp cách ly điện học để tránh sự ảnh hưởng xấu lẫn nhau giữa các thiết bị. Đó chính là các chức năng của các module vào/ra (I/O).

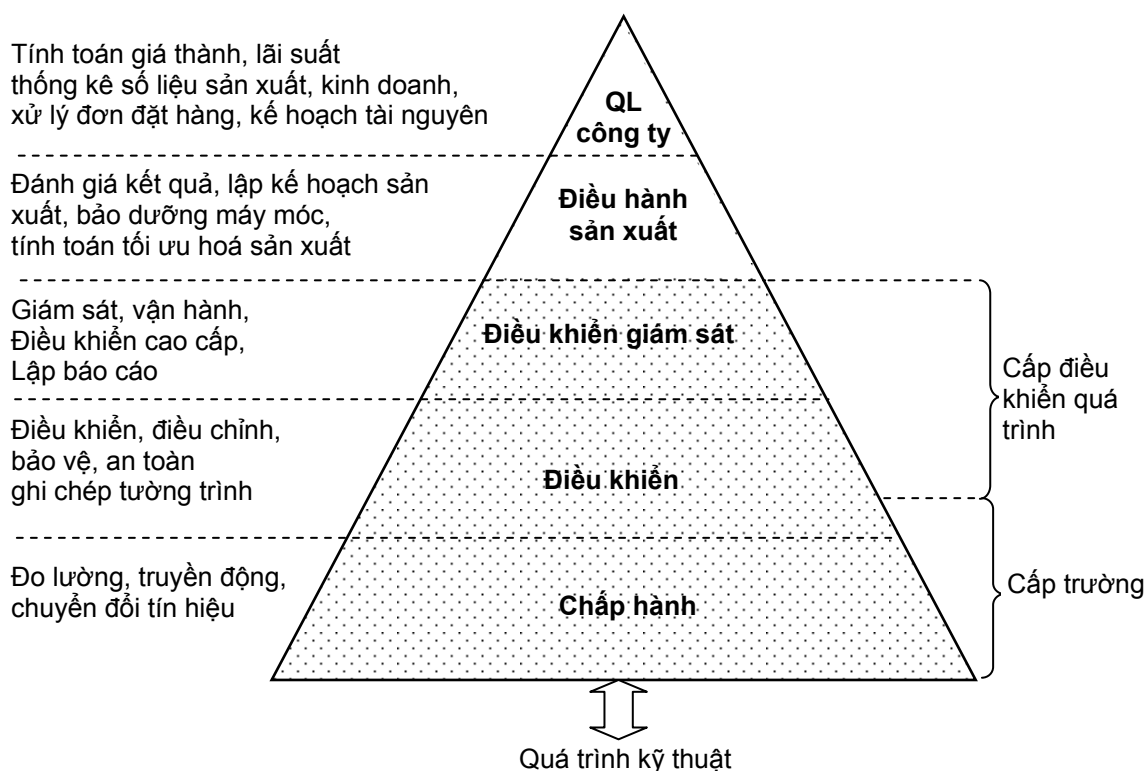
Tóm lại, một hệ thống điều khiển và giám sát bao gồm các thành phần chức năng chính sau đây:

- Giao diện quá trình: Các cảm biến và cơ cấu chấp hành, ghép nối vào/ra, chuyển đổi tín hiệu.

- Thiết bị điều khiển tự động: Các thiết bị điều khiển như các bộ điều khiển chuyên dụng, bộ điều khiển khả trình PLC (*programmable logic controller*), thiết bị điều chỉnh số đơn lẻ (*compact digital controller*) và máy tính cá nhân cùng với các phần mềm điều khiển tương ứng.
- Hệ thống điều khiển giám sát: Các thiết bị và phần mềm giao diện người máy, các trạm kỹ thuật, các trạm vận hành, giám sát và điều khiển cao cấp.
- Hệ thống truyền thông: Ghép nối điểm-điểm, bus cảm biến/chấp hành, bus trường, bus hệ thống.
- Hệ thống bảo vệ, cơ chế thực hiện chức năng an toàn.

2.2 Mô hình phân cấp

Càng ở những cấp dưới thì các chức năng càng mang tính chất cơ bản hơn và đòi hỏi yêu cầu cao hơn về độ nhanh nhạy, thời gian phản ứng. Một chức năng ở cấp trên được thực hiện dựa trên các chức năng cấp dưới, tuy không đòi hỏi thời gian phản ứng nhanh như ở cấp dưới, nhưng ngược lại lượng thông tin cần trao đổi và xử lý lại lớn hơn nhiều. Thông thường, người ta chỉ coi ba cấp dưới thuộc phạm vi của một hệ thống điều khiển và giám sát. Tuy nhiên, biểu thị hai cấp trên cùng (quản lý công ty và điều hành sản xuất) trên giúp ta hiểu thêm một mô hình lý tưởng cho cấu trúc chức năng tổng thể cho các công ty sản xuất công nghiệp.



Hình 2-2: Mô hình phân cấp chức năng của một hệ thống điều khiển và giám sát

2.2.1 Cấp chấp hành

Các chức năng chính của *cấp chấp hành* là đo lường, truyền động và chuyển đổi tín hiệu trong trường hợp cần thiết. Thực tế, đa số các thiết bị cảm biến (*sensor*) hay cơ cấu chấp hành (*actuator*) cũng có phần điều khiển riêng cho việc thực hiện đo lường/truyền động được chính xác và nhanh nhạy. Các thiết bị thông minh¹ cũng có thể đảm nhận việc xử lý thô thông tin, trước khi đưa lên cấp điều khiển.

2.2.2 Cấp điều khiển

Nhiệm vụ chính của *cấp điều khiển* là nhận thông tin từ các cảm biến, xử lý các thông tin đó theo một thuật toán nhất định và truyền đạt lại kết quả xuống các cơ cấu chấp hành. Khi còn điều khiển thủ công, nhiệm vụ đó được người đứng máy trực tiếp đảm nhiệm qua việc theo dõi các công cụ đo lường, sử dụng kiến thức và kinh nghiệm để thực hiện những thao tác cần thiết như ấn nút đóng/mở van, điều chỉnh cần gạt, núm xoay v.v... Trong một hệ thống điều khiển tự động hiện đại, việc thực hiện thủ công những nhiệm vụ đó được thay thế bằng máy tính.

2.2.3 Cấp điều khiển giám sát

Cấp điều khiển giám sát có chức năng giám sát và vận hành một quá trình kỹ thuật. Khi đa số các chức năng như đo lường, điều khiển, điều chỉnh, bảo toàn hệ thống được các cấp cơ sở thực hiện, thì nhiệm vụ của cấp điều khiển giám sát là hỗ trợ người sử dụng trong việc cài đặt ứng dụng, thao tác, theo dõi, giám sát vận hành và xử lý những tình huống bất thường. Ngoài ra, trong một số trường hợp, cấp này còn thực hiện các bài toán điều khiển cao cấp như điều khiển phối hợp, điều khiển trình tự và điều khiển theo công thức (ví dụ trong chế biến dược phẩm, hoá chất). Khác với các cấp dưới, việc thực hiện các chức năng ở cấp điều khiển giám sát thường không đòi hỏi phương tiện, thiết bị phần cứng đặc biệt ngoài các máy tính thông thường (máy tính cá nhân, máy trạm, máy chủ, terminal,...).

Như ta sẽ thấy, phân cấp chức năng như trên sẽ tiện lợi cho việc thiết kế hệ thống và lựa chọn thiết bị. Trong thực tế ứng dụng, sự phân cấp chức năng có thể khác một chút so với trình bày ở đây, tùy thuộc vào mức độ tự động hoá và cấu trúc hệ thống cụ thể. Trong những trường hợp ứng dụng đơn giản như điều khiển trang thiết bị dân dụng (máy giặt, máy lạnh, điều hòa độ ẩm,...), sự phân chia nhiều cấp có thể hoàn toàn không cần thiết. Ngược lại, trong tự động hóa một nhà máy lớn hiện đại như điện nguyên tử, sản xuất xi măng, lọc dầu, ta có thể chia nhỏ hơn nữa các cấp chức năng để tiện theo dõi.

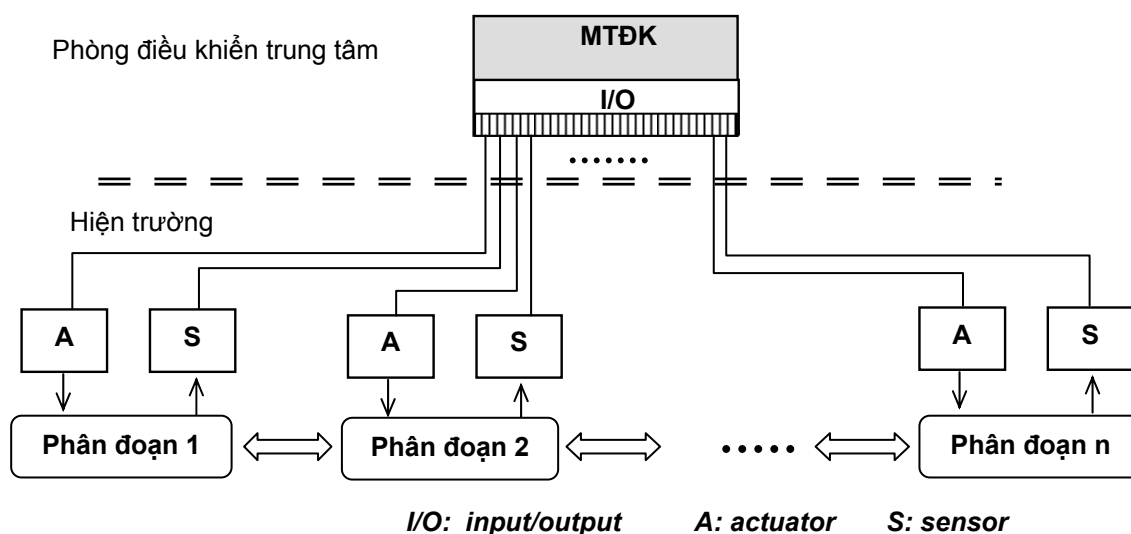
¹ Một thiết bị được gọi là thông minh, khi nó có khả năng xử lý thông tin. Thực tế, mỗi thiết bị thông minh phải có ít nhất một bộ vi xử lý riêng.

2.3 Cấu trúc điều khiển

Biến thể của cấu trúc cơ bản trên Hình 2-1 tìm thấy trong các giải pháp thực tế khác nhau ở sự phân bố chức năng điều khiển cũng như ở sự phân bố vị trí các máy tính quá trình và phụ kiện được lựa chọn. Căn cứ vào đó, ta có thể phân biệt giữa cấu trúc điều khiển tập trung và cấu trúc điều khiển phân tán, cấu trúc vào/ra tập trung và cấu trúc vào/ra phân tán.

2.3.1 Điều khiển tập trung

Cấu trúc tiêu biểu của một hệ điều khiển tập trung (*centralized control system*) được minh họa trên Hình 2-3. Một máy tính duy nhất được dùng để điều khiển toàn bộ quá trình kỹ thuật. Máy tính điều khiển ở đây (MTĐK) có thể là các bộ điều khiển số trực tiếp (DDC), máy tính lớn, máy tính cá nhân hoặc các thiết bị điều khiển khả trình. Trong điều khiển công nghiệp, máy tính điều khiển tập trung thông thường được đặt tại phòng điều khiển trung tâm, cách xa hiện trường. Các thiết bị cảm biến và cơ cấu chấp hành được nối trực tiếp, điểm-điểm với máy tính điều khiển trung tâm qua các cổng vào/ra của nó. Cách bố trí vào/ra tại máy tính điều khiển như vậy cũng được gọi là *vào/ra tập trung (central I/O)*.



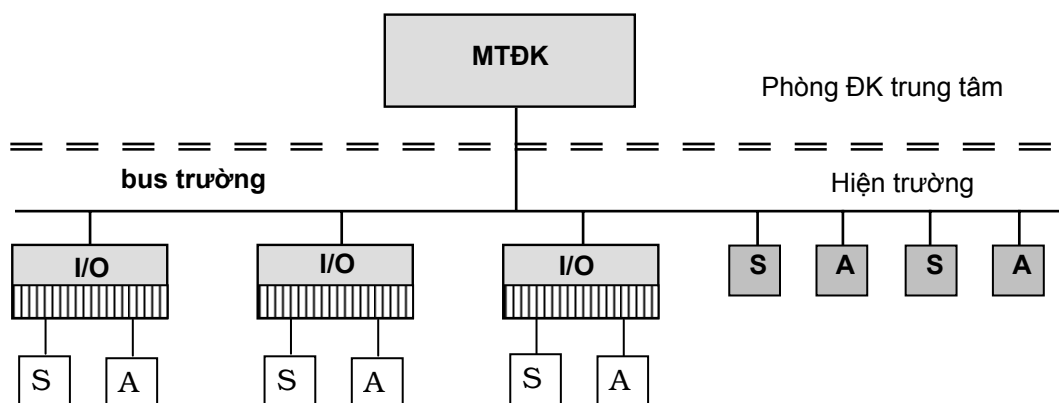
Hình 2-3: Cấu trúc điều khiển tập trung với vào/ra tập trung

Đây là cấu trúc điều khiển tiêu biểu trong những năm 1965-1975. Ngày nay, cấu trúc tập trung trên đây thường thích hợp cho các ứng dụng tự động hóa qui mô vừa và nhỏ, điều khiển các loại máy móc và thiết bị bởi sự đơn giản, dễ thực hiện và giá thành một lần cho máy tính điều khiển. Điểm đáng chú ý ở đây là sự tập trung toàn bộ “trí tuệ”, tức chức năng xử lý thông tin trong một thiết bị điều khiển duy nhất. Tuy nhiên, cấu trúc này bộc lộ những hạn chế sau:

- Công việc nối dây phức tạp, giá thành cao
- Việc mở rộng hệ thống gặp khó khăn
- Độ tin cậy kém.

2.3.2 Điều khiển tập trung với vào/ra phân tán

Cấu trúc vào/ra tập trung với cách ghép nối điểm-điểm thể hiện một nhược điểm cơ bản là số lượng lớn các cáp nối, dẫn đến giá thành cao cho dây dẫn và công thiết kế, lắp đặt. Một hạn chế khác nữa là phương pháp truyền dẫn tín hiệu thông thường giữa các thiết bị trường và thiết bị điều khiển dễ chịu ảnh hưởng của nhiễu, gây ra sai số lớn. Vấn đề này được khắc phục bằng phương pháp dùng bus trường như đã nêu trong phần trước. Hình 2-4 minh họa một cấu hình mạng đơn giản. Ở đây các module vào/ra được đẩy xuống cấp trường gần kề với các cảm biến và cơ cấu chấp hành, vì vậy được gọi là các vào/ra phân tán (*Distributed I/O*) hoặc vào/ra từ xa (*Remote I/O*). Một cách ghép nối khác là sử dụng các cảm biến và cơ cấu chấp hành thông minh (màu xám trên hình vẽ), có khả năng nối mạng trực tiếp không cần thông qua các module vào/ra. Bên cạnh khả năng xử lý giao thức truyền thông, các thiết bị này còn đảm nhiệm một số chức năng xử lý tại chỗ như lọc nhiễu, chỉnh định thang đo, tự đặt chế độ, điểm làm việc, chẩn đoán trạng thái, v.v... Trong nhiều trường hợp, các thiết bị có thể đảm nhiệm cả nhiệm vụ điều khiển đơn giản.



Hình 2-4: Cấu trúc điều khiển tập trung với vào/ra phân tán

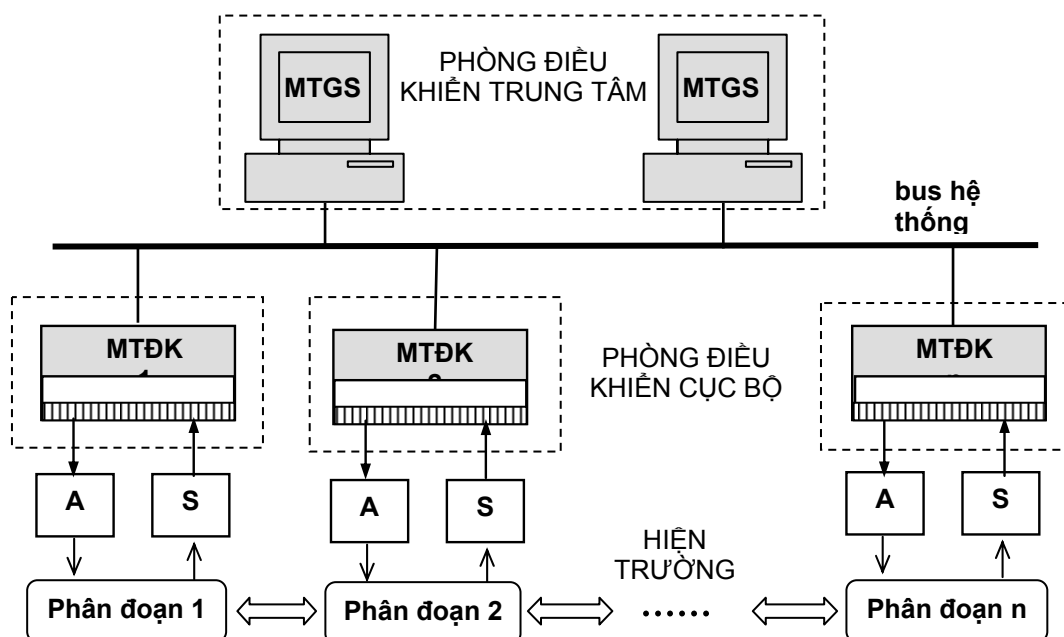
Sử dụng bus trường và cấu trúc vào/ra phân tán mang lại các ưu điểm sau:

- Tiết kiệm dây dẫn và công đi dây, nối dây
- Giảm kích thước hộp điều khiển
- Tăng độ linh hoạt hệ thống nhờ sử dụng các thiết bị có giao diện chuẩn và khả năng ghép nối đơn giản
- Thiết kế và bảo trì dễ dàng nhờ cấu trúc đơn giản
- Khả năng chẩn đoán tốt hơn (các thiết bị hỏng được phát hiện dễ dàng)
- Tăng độ tin cậy của toàn hệ thống.

2.3.3 Điều khiển phân tán

Trong đa số các ứng dụng có qui mô vừa và lớn, phân tán là tính chất cố hữu của hệ thống. Một dây chuyền sản xuất thường được phân chia thành nhiều phân đoạn, có thể được phân bố tại nhiều vị trí cách xa nhau. Để khắc

phục sự phụ thuộc vào một máy tính trung tâm trong cấu trúc tập trung và tăng tính linh hoạt của hệ thống, ta có thể điều khiển mỗi phân đoạn bằng một hoặc một số máy tính cục bộ, như Hình 2-5 minh họa.



Hình 2-5: Cấu trúc điều khiển phân tán với vào/ra tập trung

Các máy tính điều khiển cục bộ thường được đặt rải rác tại các phòng điều khiển/phòng điện của từng phân đoạn, phân xưởng, ở vị trí không xa với quá trình kỹ thuật. Các phân đoạn có liên hệ tương tác với nhau, vì vậy để điều khiển quá trình tổng hợp cần có sự điều khiển phối hợp giữa các máy tính điều khiển. Trong phần lớn các trường hợp, các máy tính điều khiển được nối mạng với nhau và với một hoặc nhiều máy tính giám sát (MTGS) trung tâm qua *bus hệ thống*. Giải pháp này dẫn đến các hệ thống có cấu trúc điều khiển phân tán, hay được gọi là các *hệ điều khiển phân tán* (HĐKPT).

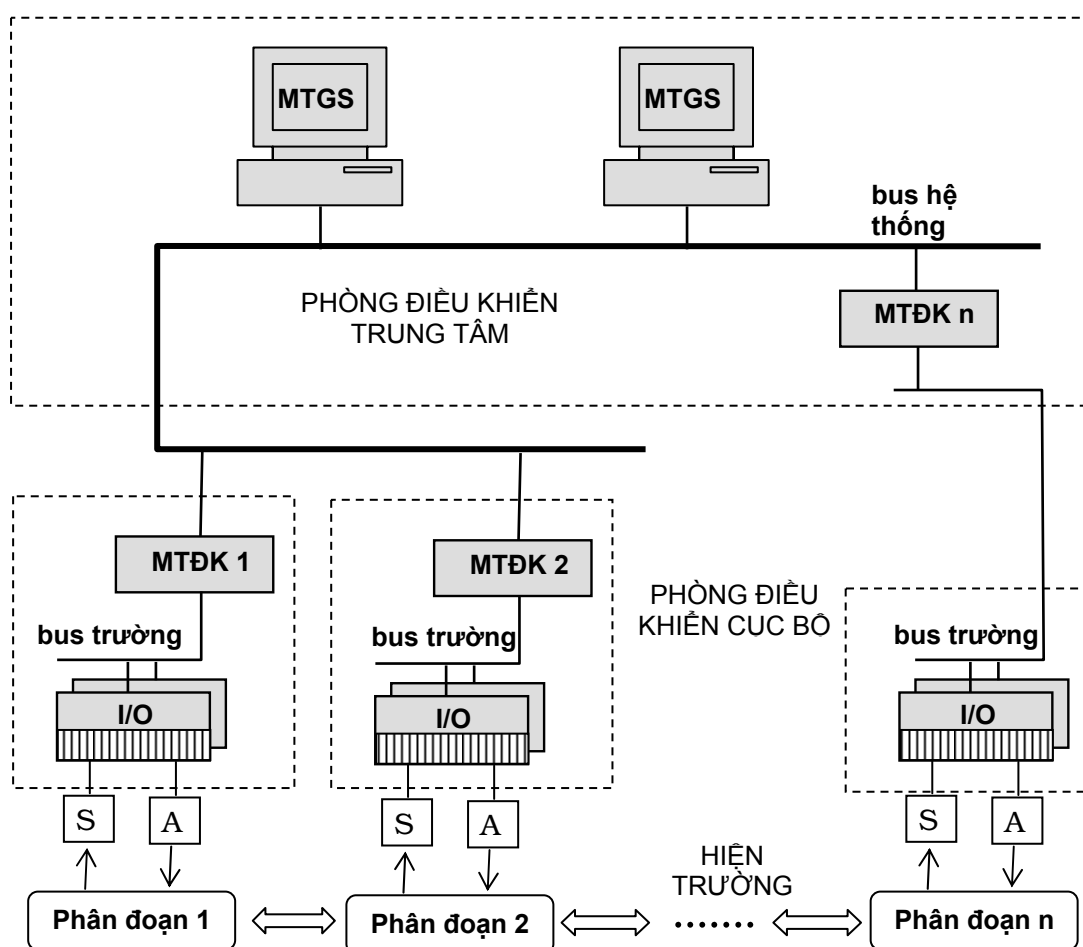
Ưu thế của cấu trúc điều khiển phân tán không chỉ dừng lại ở độ linh hoạt cao hơn so với cấu trúc tập trung. Hiệu năng cũng như độ tin cậy tổng thể của hệ thống được nâng cao nhờ sự phân tán chức năng xuống các cấp dưới. Việc phân tán chức năng xử lý thông tin và phối hợp điều khiển có sự giám sát từ các trạm vận hành trung tâm mở ra các khả năng ứng dụng mới, tích hợp trọn vẹn trong hệ thống như lập trình cao cấp, điều khiển trình tự, điều khiển theo công thức và ghép nối với cấp điều hành sản xuất.

2.3.4 Điều khiển phân tán với vào/ra phân tán

Lưu ý rằng Hình 2-5 chỉ minh họa cách ghép nối điểm-điểm giữa một máy tính điều khiển với các cảm biến và cơ cấu chấp hành, sử dụng vào/ra tập trung. Tuy nhiên, ta cũng có thể sử dụng bus trường để thực hiện cấu trúc vào/ra phân tán như trên Hình 2-6. Khi đó, máy tính điều khiển có thể đặt tại

phòng điều khiển trung tâm hoặc tại các phòng điều khiển cục bộ, tùy theo qui mô của hệ thống và khả năng kéo dài của bus trường.

Giải pháp sử dụng các hệ điều khiển phân tán với cấu trúc vào/ra phân tán và các thiết bị trường thông minh chính là xu hướng trong xây dựng các hệ thống điều khiển và giám sát hiện đại. Bên cạnh độ tin cậy cao, tính năng mở và độ linh hoạt cao thì yếu tố kinh tế cũng đóng vai trò quan trọng. Việc phân tán chức năng xử lý thông tin, chức năng điều khiển theo bề rộng cũng như theo chiều sâu là tiền đề cho kiến trúc “trí tuệ phân tán” (*distributed intelligence*) trong tương lai.



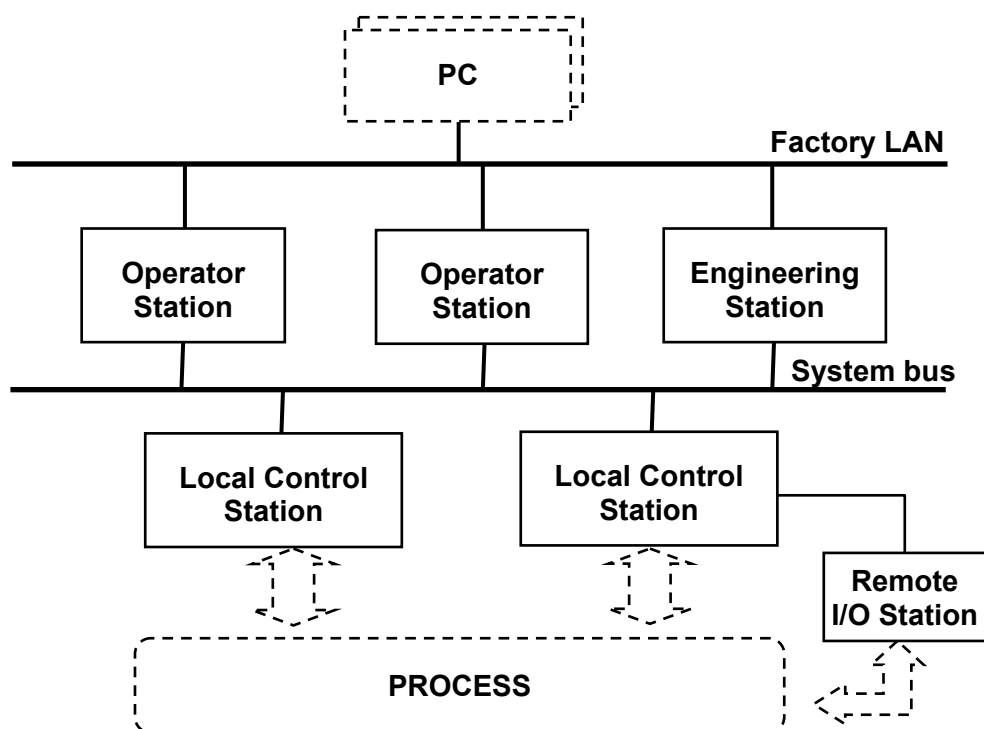
Hình 2-6: Cấu trúc điều khiển phân tán với vào/ra phân tán

3 CÁC THÀNH PHẦN CỦA MỘT HỆ ĐIỀU KHIỂN PHÂN TÁN

3.1 Cấu hình cơ bản

Cấu hình cơ bản một hệ điều khiển phân tán được minh họa trên Hình 3-1, bao gồm các thành phần sau:

- Các trạm điều khiển cục bộ (local control station, LCS), đôi khi còn được gọi là các khối điều khiển cục bộ (local control unit, LCU) hoặc các trạm quá trình (process station, PS).
- Các trạm vận hành (operator station, OS)
- Trạm kỹ thuật (engineering station, ES) và các công cụ phát triển
- Hệ thống truyền thông (field bus, system bus).



Hình 3-1: Cấu hình cơ bản một hệ điều khiển phân tán

Đây là cấu hình tối thiểu, các cấu hình cụ thể có thể chứa các thành phần khác như trạm vào/ra từ xa (remote I/O station), các bộ điều khiển chuyên dụng,...

3.1.1 Trạm điều khiển cục bộ

Thông thường, các trạm điều khiển cục bộ được xây dựng theo cấu trúc module. Các thành phần chính bao gồm:

- Bộ cung cấp nguồn, thông thường có dự phòng
- Khối xử lý trung tâm (CPU), có thể lựa chọn loại có dự phòng

- Giao diện với bus hệ thống, thông thường cũng có dự phòng
- Giao diện với bus trường nếu sử dụng cấu trúc vào/ra phân tán
- Các module vào/ra số cũng như tương tự, đặc biệt là các module vào/ra an toàn cháy nổ

Trong cấu trúc vào/ra tập trung, các module vào/ra được nối với CPU thông qua bus nội bộ đằng sau giá đỡ (*backplane-bus*). Chính vì vậy, các module này cũng phải do nhà sản xuất cung cấp kèm theo CPU.

Trong các hệ thống điều khiển quá trình, một trạm điều khiển cục bộ cũng thường được cài đặt giao diện HART và các module ghép nối phụ kiện khác. Các thiết bị này được lắp đặt trong tủ điều khiển cùng với các linh kiện hỗ trợ khác như hàng kẹp đấu dây, các bộ chuyển đổi tín hiệu (*transducers*), các khối đầu cuối (*terminal block*),... Các tủ điều khiển thường được đặt trong phòng điều khiển/phòng điện ở bên cạnh phòng điều khiển trung tâm hoặc rải rác gần khu vực hiện trường.

Các chức năng do trạm điều khiển cục bộ đảm nhiệm bao gồm:

- Điều khiển quá trình (*process control*): Điều khiển các mạch vòng kín (nhiệt độ, áp suất, lưu lượng, độ pH, độ đậm đặc,...). Hầu hết các mạch vòng đơn được điều khiển trên cơ sở luật PID, giải quyết bài toán điều khiển điều chỉnh, điều khiển tỉ lệ, điều khiển tăng. Các hệ thống hiện đại cho phép điều khiển mờ, điều khiển dựa mô hình (*model-based control*), điều khiển thích nghi, ...
- Điều khiển trình tự (*sequential control*, *sequence control*)
- Điều khiển logic
- Thực hiện các công thức (*recipe control*).
- Đặt các tín hiệu đầu ra về trạng thái an toàn trong trường hợp có sự cố hệ thống
- Lưu trữ tạm thời các tín hiệu quá trình trong trường hợp mất liên lạc với trạm vận hành
- Nhận biết các trường hợp vượt ngưỡng giá trị và tạo các thông báo báo động.

Chính vì đây là thành phần quan trọng nhất trong hệ thống, đại đa số các trạm điều khiển cục bộ có tính năng kiểm tra và sửa lỗi (*error checking and correcting*, ECC), cũng như cho phép lựa chọn cấu hình dự phòng. Một điều quan trọng là một trạm điều khiển cục bộ phải có khả năng đảm bảo tiếp tục thực hiện các chức năng nói trên trong trường hợp trạm vận hành hoặc đường truyền bus hệ thống có sự cố.

Các máy tính điều khiển có thể là máy tính đặc chủng của nhà cung cấp (*vendor-specific controller*), PLC hoặc máy tính cá nhân công nghiệp. Dựa trên cơ sở này có thể phân loại các hệ thống điều khiển phân tán có mặt hiện nay trên thị trường thành các hệ các hệ truyền thống (sau đây gọi là *DCS truyền thống*), các hệ trên nền PLC (*PLC-based DCS*) và các hệ trên nền PC (*PC-based DCS*).

Bất kể chủng loại thiết bị nào được sử dụng, các yêu cầu quan trọng nhất về mặt kỹ thuật được đặt ra cho một trạm điều khiển cục bộ là:

- Tính năng thời thực
- Độ tin cậy và tính sẵn sàng
- Lập trình thuận tiện, cho phép sử dụng/cài đặt các thuật toán cao cấp
- Khả năng điều khiển lai (liên tục, trình tự và logic).

3.1.2 Bus trường và các trạm vào/ra từ xa

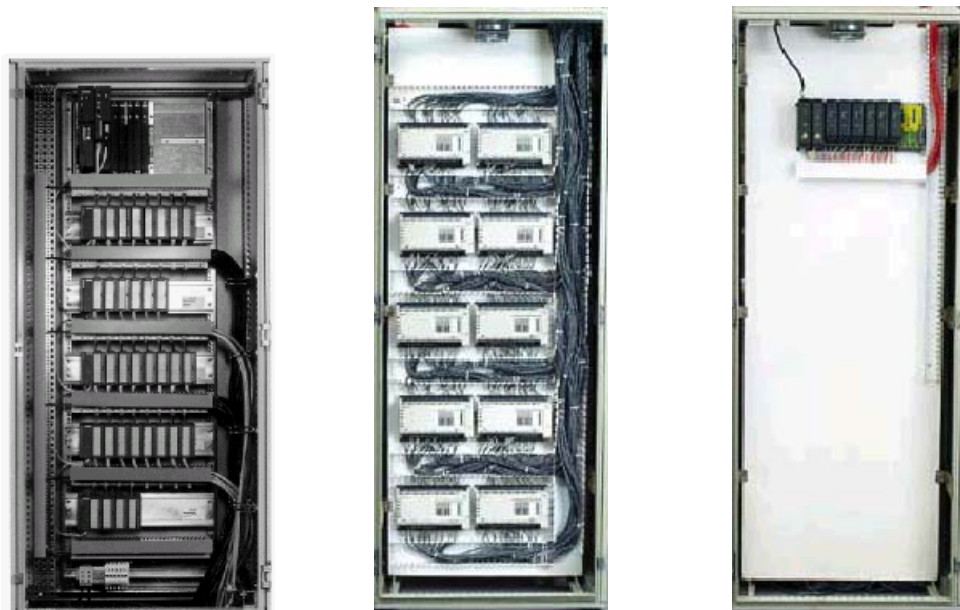
Khi sử dụng cấu trúc vào/ra phân tán, các trạm điều khiển cục bộ sẽ được bổ sung các module giao diện bus để nối với các trạm vào/ra từ xa (*remote I/O station*) và một số thiết bị trường thông minh. Các yêu cầu chung đặt ra với bus trường là tính năng thời gian thực, mức độ đơn giản và giá thành thấp. Bên cạnh đó, đối với môi trường dễ cháy nổ còn các yêu cầu kỹ thuật đặc biệt khác về chuẩn truyền dẫn, tính năng điện học của các linh kiện mạng, cáp truyền,... Các loại bus trường được hỗ trợ mạnh nhất là Profibus-DP, Foundation Fieldbus, DeviceNet và AS-I. Trong môi trường đòi hỏi an toàn cháy nổ thì Profibus-PA và Foundation Fieldbus H1 là hai hệ được sử dụng phổ biến nhất.

Một trạm vào/ra từ xa thực chất có cấu trúc không khác lắm so với một trạm điều khiển cục bộ, duy chỉ thiếu khối xử lý trung tâm cho chức năng điều khiển. Thông thường, các trạm vào/ra từ xa được đặt rất gần với quá trình kỹ thuật, vì thế tiết kiệm nhiều cáp truyền và đơn giản hóa cấu trúc hệ thống. Trạm vào/ra từ xa cũng có thể đặt cùng vị trí với trạm điều khiển cục bộ, tuy nhiên như vậy không lợi dụng được ưu điểm của cấu trúc này.

Khác với cấu trúc vào/ra tập trung, cấu trúc vào/ra phân tán cho phép sử dụng các trạm vào/ra từ xa của các nhà cung cấp khác với điều kiện có hỗ trợ loại bus trường qui định. Tuy nhiên, để có thể khai thác tối đa khả năng các công cụ phần mềm tích hợp và đảm bảo tương thích hoàn toàn giữa các thành phần trong một hệ DCS, việc dùng trọn sản phẩm của một hãng vẫn là giải pháp an toàn nhất.

Bên cạnh phương pháp ghép nối thiết bị điều khiển với quá trình kỹ thuật thông qua các module vào/ra, ta có thể sử dụng các cảm biến hoặc cơ cấu chấp hành có giao diện bus trường. Qua đó có thể đơn giản hóa cấu trúc hệ thống hơn nữa, tiết kiệm tiết kiệm chỗ trong tủ điều khiển và nâng cao tính năng thời gian thực của hệ thống do tận dụng được khả năng xử lý thông tin của các thiết bị trường.

Trên Hình 3-4 là hình ảnh một số tủ điều khiển DCS. Hình bên trái là một trạm PCS7 (Siemens) với bộ điều khiển lắp đặt cùng các module vào/ra phân tán. Hình giữa minh họa một trạm vào/ra từ xa lắp độc lập. Tủ điều khiển bên phải minh họa trạm điều khiển cục bộ DeltaV (Fisher-Rosermount) sử dụng giải pháp Foundation Fieldbus (không cần các module các vào/ra).



Hình 3-2: Một số hình ảnh tủ điều khiển DCS

3.1.3 Trạm vận hành

Trạm vận hành và trạm kỹ thuật được đặt tại phòng điều khiển trung tâm. Các trạm vận hành có thể hoạt động song song, độc lập với nhau. Để tiện cho việc vận hành hệ thống, người ta thường sắp xếp mỗi trạm vận hành tương ứng với một phân đoạn hoặc một phân xưởng. Tuy nhiên, các phần mềm chạy trên tất cả các trạm hoàn toàn giống nhau, vì thế trong trường hợp cần thiết mỗi trạm đều có thể thay thế chức năng của các trạm khác.

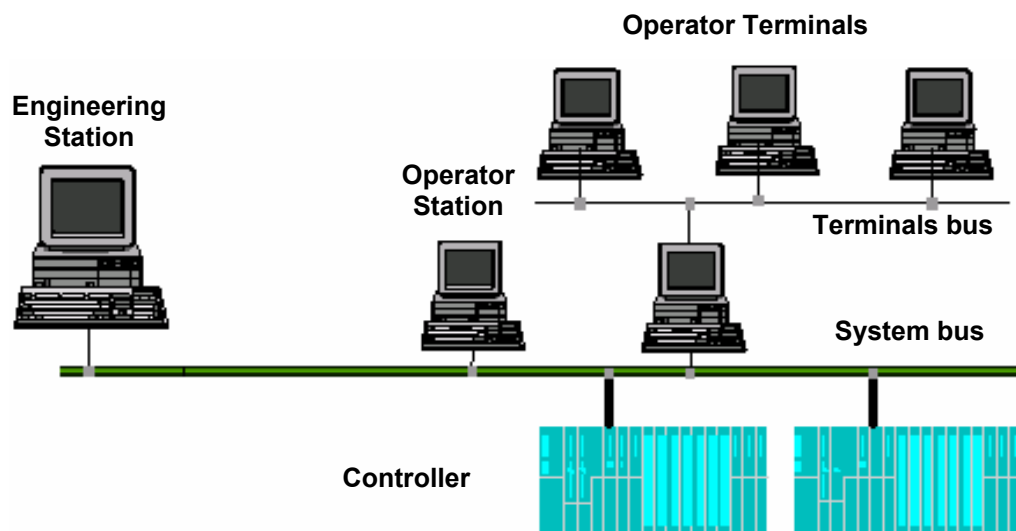
Các chức năng tiêu biểu của một trạm vận hành gồm có:

- Hiển thị các hình ảnh chuẩn (hình ảnh tổng quan, hình ảnh nhóm, hình ảnh từng mạch vòng, hình ảnh điều khiển trình tự, các đồ thị thời gian thực và đồ thị quá khứ)
- Hiển thị các hình ảnh đồ họa tự do (lưu đồ công nghệ, các phím điều khiển)
- Hỗ trợ vận hành hệ thống qua các công cụ thao tác tiêu biểu, các hệ thống hướng dẫn chỉ đạo và hướng dẫn trợ giúp
- Tạo và quản lý các công thức điều khiển (cho điều khiển mẻ)
- Xử lý các sự kiện, sự cố
- Xử lý, lưu trữ và quản lý dữ liệu
- Chẩn đoán hệ thống, hỗ trợ người vận hành và bảo trì hệ thống
- Hỗ trợ lập báo cáo tự động

Khác với các trạm điều khiển, hầu hết các hệ DCS hiện đại đều sử dụng các sản phẩm thương mại thông dụng như máy tính cá nhân (công nghiệp) chạy trên nền WindowsNT/2000, hoặc các máy tính trạm chạy trên nền UNIX. Cùng với các màn hình màu lớn (thường là 19inch) với độ phân giải cao để theo dõi quá trình sản xuất, một trạm vận hành hiện đại bao giờ cũng có các

thiết bị thao tác chuẩn như bàn phím và chuột. Một trạm vận hành có thể bố trí theo kiểu một người sử dụng (một hoặc nhiều màn hình), hoặc nhiều người sử dụng với nhiều Terminals (Hình 3-3).

Các phần mềm trên trạm vận hành bao giờ cũng đi kèm đồng bộ với hệ thống, song thường hỗ trợ các chuẩn phần mềm và chuẩn giao tiếp công nghiệp như TCP/IP, DDE (*Dynamic Data Exchange*), OLE (*Object Linking and Embedding*), ODBC (*Open Data Base Connection*), OPC (*OLE for Process Control*).



Hình 3-3: Các phương pháp bố trí trạm vận hành

Đặc điểm tiêu biểu của các trạm vận hành hiện đại là sử dụng kỹ thuật giao diện người-máy kiểu đa cửa sổ với các phần tử giao diện chuẩn. Tuy nhiên, việc thiết kế các màn hình giao diện công nghiệp khác với các giao diện ứng dụng văn phòng, đòi hỏi kiến thức tổng hợp về quá trình công nghệ, mỹ thuật công nghiệp, tâm lý học công nghiệp và công nghệ phần mềm. Vấn đề này sẽ được đề cập chi tiết sau.

3.1.4 Trạm kỹ thuật và các công cụ phát triển

Trạm kỹ thuật là nơi cài đặt các công cụ phát triển, cho phép đặt cấu hình cho hệ thống, tạo và theo dõi các chương trình ứng dụng điều khiển và giao diện người máy, đặt cấu hình và tham số hóa các thiết bị trường. Việc tạo ứng dụng điều khiển hầu hết được thực hiện theo phương pháp khai báo, đặt tham số và ghép nối các khối chức năng có sẵn trong thư viện. Cũng như các trạm vận hành, thiết bị sử dụng thông thường là các máy tính cá nhân (công nghiệp) chạy trên nền Windows95/98/NT/2000 hoặc UNIX.

Một số đặc tính tiêu biểu của các công cụ phát triển trên trạm kỹ thuật là:

- Các công cụ phát triển được tích hợp sẵn trong hệ thống
- Công việc phát triển (*Engineering*) không yêu cầu có phần cứng DCS tại chỗ

- Các ngôn ngữ lập trình thông dụng là sơ đồ khối hàm (FBD-*Function Block Diagram*, hoặc CFC-*Continuous Function Chart*) và biểu đồ tiến trình (SFC-*Sequential Function Chart*), tương tự IEC61131-3 FBD () và SFC
- Một dự án có thể do nhiều người cùng phối hợp phát triển song song
- Giao diện với các hệ thống cấp trên (CAD/CAM, MES, PPS, ERP,...)

Để việc phát triển hệ thống phần mềm được thuận lợi, các nhà sản xuất cung cấp các thư viện khối hàm chuyên dụng. Bên cạnh đó, nhiều nhà sản xuất cũng cung cấp phần mềm mô phỏng để người phát triển hệ thống có thể tạo các đầu vào/ra mô phỏng, giúp cho việc phát triển phần mềm được chắc chắn, an toàn hơn.

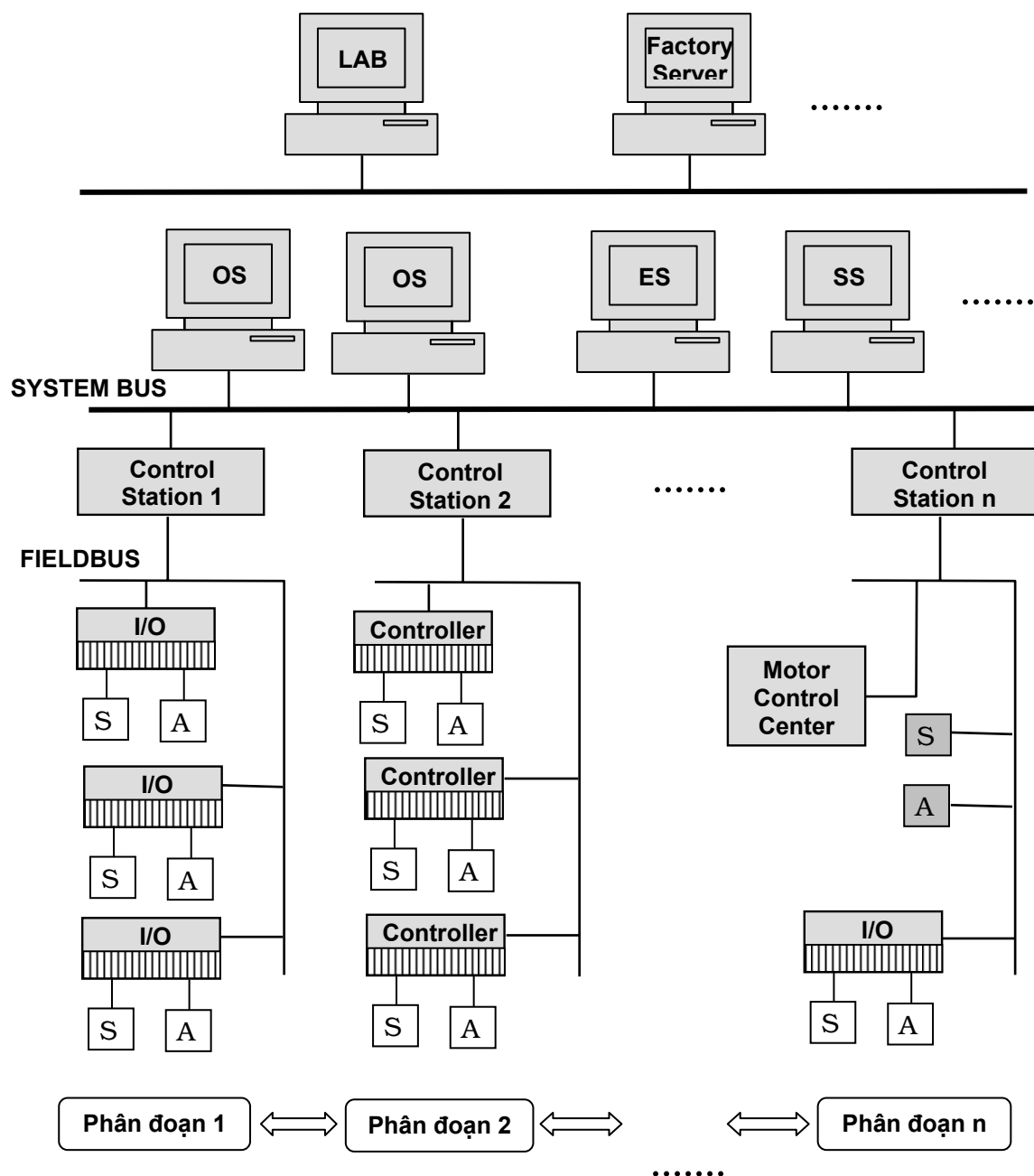
Trong một số hệ thống, người ta không phân biệt giữa trạm vận hành và trạm kỹ thuật, mà sử dụng một bàn phím có khóa chuyển qua lại giữa hai chế độ vận hành và phát triển.

3.1.5 Bus hệ thống

Bus hệ thống có chức năng nối mạng các trạm điều khiển cục bộ với nhau và với các trạm vận hành và trạm kỹ thuật. Trong đa số các hệ thống ứng dụng, người ta lựa chọn cấu hình có dự phòng cho bus hệ thống. Thêm nữa, để cải thiện tính năng thời gian thực, nhiều khi một mạng riêng biệt (có thể có cả dự phòng) được sử dụng để ghép nối các trạm điều khiển cục bộ (*bus điều khiển, control bus*). Giải pháp mạng có thể đặc chủng của riêng công ty, hoặc dựa trên một mạng chuẩn quốc tế. Các hệ thống mạng được sử dụng nhiều nhất là Ethernet, Profibus-FMS và ControlNet.

Đặc điểm của việc trao đổi thông tin qua bus hệ thống là lưu lượng thông tin lớn, vì vậy tốc độ đường truyền phải tương đối cao. Tính năng thời gian thực cũng là một yêu cầu được đặt ra (nhất là đối với bus điều khiển), tuy nhiên không nghiêm ngặt như với bus trường. Thời gian phản ứng thường chỉ yêu cầu nằm trong phạm vi 0,1s trở lên. Số lượng trạm tham gia thường không lớn và nhu cầu trao đổi dữ liệu không có đột biến lớn. Vì vậy đối với mạng Ethernet, tính bất định của phương pháp truy nhập bus CSMA/CD thường không phải là vấn đề gây lo ngại.

Hình 3-4 minh họa cấu hình tiêu biểu của một hệ điều khiển phân tán hiện đại. Bên cạnh các thành phần đã mô tả, một cấu hình tiêu biểu thường có thêm một số trạm server, máy tính phân tích, máy in, một số bộ điều khiển cục bộ chuyên dụng...



Hình 3-4: Cấu hình tiêu biểu một hệ điều khiển phân tán hiện đại

3.2 Phân loại các hệ DCS

3.2.1 Các hệ DCS truyền thống

Các hệ này sử dụng các bộ điều khiển quá trình đặc chủng theo kiến trúc riêng của nhà sản xuất. Các hệ cũ thường đóng kín, ít tuân theo các chuẩn giao tiếp công nghiệp, các bộ điều khiển được sử dụng cũng thường chỉ làm nhiệm vụ điều khiển quá trình, vì vậy phải sử dụng kết hợp PLC cho các bài toán điều khiển logic và điều khiển trình tự. Các hệ mới có tính năng mở tốt hơn, một số bộ điều khiển lại đảm nhiệm cả các chức năng điều khiển quá trình, điều khiển trình tự và điều khiển logic (*hybrid controller*).

Để hỗ trợ các bài toán điều khiển quá trình diễn ra đồng thời, khối xử lý trung tâm được cài đặt một hệ điều hành thời gian thực, đa nhiệm - hoặc của riêng nhà sản xuất phát triển hoặc một sản phẩm thông dụng như pSOS, TSOS, VRTX,... Chu kỳ thời gian nhỏ nhất thực hiện các mạch vòng điều khiển thường nằm trong khoảng 10-100ms, trong trường hợp đặc biệt (ví dụ cho nhà máy điện) có thể tới 1ms.

Một số sản phẩm tiêu biểu cùng với tên trạm điều khiển cục bộ được liệt kê dưới đây:

- AdvantOCS (ABB): Advant Controller, hệ điều hành riêng
- Freelance 2000 (ABB): D-PS học D-FC, hệ điều hành pSOS
- Symphonie (ABB): Melody, hệ điều hành pSOS
- DeltaV (Fisher-Rosermount): Visual Controller, hệ điều hành TSOS
- I/A Series (Foxboro): CP60, hệ điều hành VRTX
- PlantScape (Honeywell): PlantScape Controller, hệ điều hành riêng
- Centum CS1000/CS3000 (Yokogawa): PFCx-E, AFS10x/AFS20x, hệ điều hành ORKID

3.2.2 Các hệ DCS trên nền PLC

Thiết bị điều khiển khả trình (PLC, *programmable logic controller*) là một loại máy tính điều khiển chuyên dụng, do nhà phát minh người Mỹ Richard Morley lần đầu tiên đưa ra ý tưởng vào năm 1968. Dựa trên yêu cầu kỹ thuật của General Motors là xây dựng một thiết bị có khả năng lập trình mềm dẻo thay thế cho mạch điều khiển logic cứng, hai công ty độc lập là Allen Bradley và Bedford Associates (sau này là Modicon) đã đưa ra trình bày các sản phẩm đầu tiên. Các thiết bị này chỉ xử lý được một tập lệnh logic cơ bản, 128 điểm vào/ra (1 bit) và 1kByte bộ nhớ. Lúc đầu, cái tên *programmable controller*, viết tắt là PC, được sử dụng rộng rãi. Trong khi đó, *programmable logic controller* hay PLC là thương hiệu đăng ký của công ty Allen Bradley. Sau này, khi máy tính cá nhân trở nên phổ biến thì từ viết tắt PLC hay được dùng hơn để tránh nhầm lẫn. Vì vậy từ đây về sau ta sẽ dùng khái niệm *thiết bị điều khiển khả trình* nhưng với từ viết tắt là *PLC*.

Với cấu trúc ghép nối vào/ra linh hoạt, nguyên tắc làm việc đơn giản theo chu kỳ, khả năng lập trình và lưu trữ chương trình trong bộ nhớ không cần can thiệp trực tiếp tới phần cứng, PLC nhanh chóng thu hút sự chú ý trong giới chuyên ngành. Vào thời điểm các máy tính điều khiển chuyên dụng và không chuyên dụng đều có kích cỡ rất lớn và giá thành rất cao, thì việc sử dụng PLC là giải pháp lý tưởng để thay thế các mạch logic tổ hợp và tuần tự trong điều khiển các quá trình gián đoạn.

Cho đến nay, danh mục các chủng loại PLC có mặt trên thị trường thật phong phú đến mức khó có thể bao quát. Chúng không những khác nhau ở công suất làm việc của bộ xử lý trung tâm, ở dung lượng bộ nhớ và ở số lượng các cổng vào/ra, mà còn ở các đặc tính chức năng như cấu trúc linh hoạt,

phương pháp lập trình và khả năng nối mạng. Trừ một số loại nhỏ dùng trong các ứng dụng đơn giản, hầu hết các PLC hiện đại đều không dừng lại ở việc thực hiện các phép tính logic đơn giản, mà còn có khả năng làm việc với các tín hiệu tương tự và thực hiện các phép toán số học, thậm chí cả các thuật toán điều khiển phản hồi như điều khiển nhiều điểm, PID và điều khiển mờ. Các bộ đếm, bộ định thời và một số hàm toán học thông dụng thuộc phạm vi chức năng chuẩn của một PLC. Việc sử dụng PLC vì vậy không chỉ dừng lại ở các quá trình gián đoạn, mà nay đã rất phổ biến đối với điều khiển các quá trình liên tục như trong công nghiệp chế biến, khai thác, công nghệ môi trường v.v...

Một số hệ DCS trên nền PLC tiêu biểu là SattLine (ABB), Process Logix (Rockwell), Modicon TSX (Schneider Electric), PCS7 (Siemens),... Thực chất, ngày nay đa số các PLC vừa có thể sử dụng cho bài toán điều khiển logic và điều khiển quá trình. Tuy nhiên, các PLC được sử dụng trong các hệ điều khiển phân tán thường có cấu hình mạnh, hỗ trợ điều khiển trình tự cùng với các phương pháp lập trình hiện đại (ví dụ SFC).

Cấu trúc phân cứng

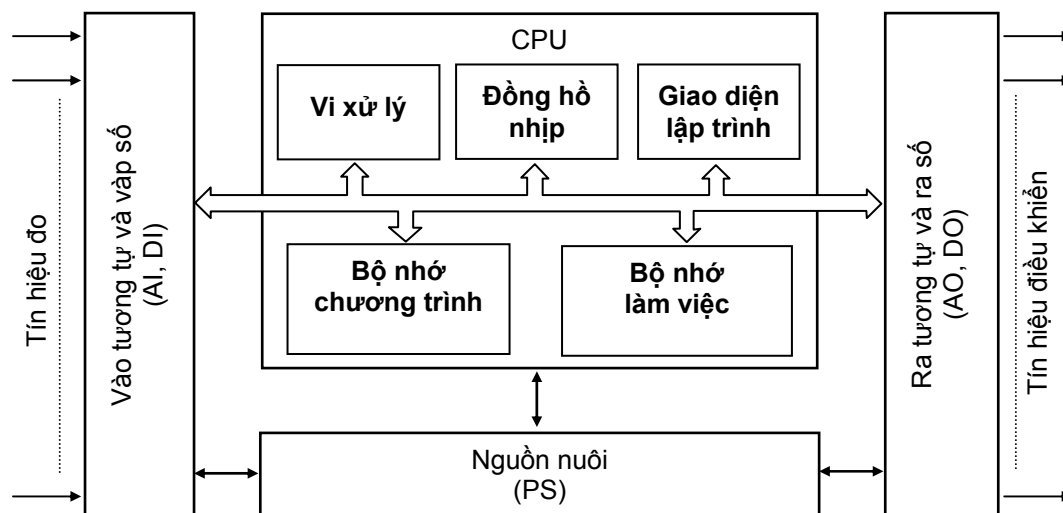
Hình 3-5 minh họa các thành phần chức năng chính của một hệ thống thiết bị điều khiển khả trình và quan hệ tương tác giữa chúng. Về cơ bản, một PLC cũng có các thành phần giống như một máy vi tính thông thường, đó là vi xử lý, các bộ nhớ làm việc và bộ nhớ chương trình, giao diện vào/ra và cung cấp nguồn. Tuy nhiên, một điểm khác cơ bản là các thành phần giao diện người-máy như màn hình, bàn phím và chuột không được trang bị ở đây. Việc lập trình vì vậy phải được thực hiện gián tiếp bằng một máy tính riêng biệt, ghép nối với CPU thông qua giao diện thiết bị lập trình (thường là một cổng nối tiếp theo chuẩn RS-232 hoặc RS-485).

Bộ xử lý trung tâm (*Central Processing Unit*, CPU) bao gồm một hoặc nhiều vi xử lý, bộ nhớ chương trình, bộ nhớ làm việc, đồng hồ nhịp và giao diện với thiết bị lập trình, được liên kết với nhau thông qua một hệ bus nội bộ. Nhiệm vụ chính của CPU là quản lý các cổng vào/ra, xử lý thông tin, thực hiện các thuật toán điều khiển. Bộ nhớ chương trình thường có dạng EPROM (*Erasable and Programmable Read Only Memory*) hoặc EEPROM (*Electrically Erasable and Programmable Read Only Memory*), chứa hệ điều hành và mã chương trình ứng dụng. Dữ liệu vào/ra cũng như các dữ liệu tính toán khác được lưu trong bộ nhớ làm việc RAM (*Random Access Memory*). Đồng hồ nhịp có vai trò tạo ngắt cứng để điều khiển chương trình theo chu kỳ, thông thường trong khoảng từ 0,01 giây tới 1000 phút.

Các thành phần vào/ra (*input/output*, I/O) đóng vai trò là giao diện giữa CPU và quá trình kỹ thuật. Nhiệm vụ của chúng là chuyển đổi, thích ứng tín hiệu và cách ly galvanic giữa các thiết bị ngoại vi (cảm biến, cơ cấu chấp hành) và CPU. Các thành phần vào/ra được liên kết với CPU thông qua một hệ bus nội bộ hoặc qua một hệ bus trường (xem chương 3).

Bộ cung cấp nguồn (*power supply*, PS) có vai trò biến đổi và ổn định nguồn nuôi (thông thường 5V) cho CPU và các thành phần chức năng khác từ một nguồn xoay chiều (110V, 220V,...) hoặc một chiều (12V, 24V,...).

Bên cạnh các thành phần chính nêu trên, một hệ thống PLC có thể có các thành phần chức năng khác như ghép nối mở rộng, điều khiển chuyên dụng và xử lý truyền thông.



Hình 3-5: Các thành phần chức năng chính của một PLC

Thiết kế module và thiết kế gọn

Tùy theo sự phân chia chức năng trên các thành phần thiết bị, ta có thể phân biệt giữa các PLC có thiết kế module và các PLC có thiết kế gọn. Trong một PLC có thiết kế gọn, tất cả các chức năng được tích hợp gọn trong một thiết bị. Thông thường, loại PLC này có sẵn một số cổng vào/ra cố định. Một số cũng được tích hợp giao diện truyền thông cho một loại bus trường. Tuy nhiên, một số ít loại có cấu trúc gọn vẫn cho phép tăng số lượng cổng vào/ra hoặc bổ sung giao diện mạng bằng các module mở rộng đặc biệt. PLC có cấu trúc gọn thích hợp với các bài toán đơn giản.

Đối với các ứng dụng có qui mô vừa và lớn, ta cần sử dụng các PLC có thiết kế module bởi độ linh hoạt cao. Ở đây, hầu hết mỗi thành phần chức năng được thực hiện bởi một module phần cứng riêng biệt, được lắp đặt trên một hoặc nhiều giá đỡ. Bên cạnh các thành phần cơ bản là CPU, nguồn và các module vào/ra, một PLC còn có thể chứa các module chức năng, các module ghép nối và module truyền thông. Hệ bus nội bộ được sử dụng để ghép nối các module mở rộng với CPU thường được gọi là bus mặt sau (*backplane bus*).

Các module chức năng (*function module*, FM) được sử dụng để thực hiện một số nhiệm vụ điều khiển riêng, ví dụ module điều khiển PID, module điều khiển động cơ bước, module cân,... Các module này hoạt động tương đối độc lập với CPU, tuy nhiên có thể trao đổi dữ liệu quá trình và dữ liệu tham số thông qua bus nội bộ và các hàm hoặc khối hàm giao tiếp hệ thống.

Các module ghép nối (*interface module*, IM) được sử dụng trong việc mở rộng hệ thống khi số lượng các module lớn, không đủ chỗ trên một giá đỡ. Thông thường, mỗi giá đỡ cần có một module nguồn riêng bên cạnh module ghép nối. Thông qua các module ghép nối, một CPU có thể quản lý tất cả các module trên các giá đỡ. Số lượng và chủng loại các module cho phép trên một giá đỡ cũng như số lượng tổng cộng phụ thuộc vào khả năng quản lý của loại CPU cụ thể.

Các module truyền thông (*communication module*, CM) có vai trò là giao diện mạng, được sử dụng để ghép nối nhiều PLC với nhau, với các thiết bị trường và với máy tính giám sát. Các module truyền thông đảm nhiệm xử lý giao thức một cách độc lập với CPU. Tuy nhiên trong một số trường hợp, bộ xử lý trung tâm cũng được tích hợp sẵn giao diện mạng cho một hệ bus trường thông dụng.

3.2.3 Các hệ DCS trên nền PC

Giải pháp sử dụng máy tính cá nhân (PC) trực tiếp làm thiết bị điều khiển không những được bàn tới rộng rãi, mà đã trở thành thực tế phổ biến trong những năm gần đây. Nếu so sánh với các bộ điều khiển khả trình (PLC) và các bộ điều khiển DCS đặc chủng thì thế mạnh của PC không những nằm ở tính năng mở, khả năng lập trình tự do, hiệu năng tính toán cao và đa chức năng, mà còn ở khía cạnh kinh tế. Các bước tiến lớn trong kỹ thuật máy tính, công nghiệp phần mềm và công nghệ bus trường chính là các yếu tố thúc đẩy khả năng cạnh tranh của PC trong điều khiển công nghiệp.

DCS trên nền PC là một hướng giải pháp tương đối mới, mới có một số sản phẩm trên thị trường như PCS7 (Siemens, giải pháp Slot-PLC), 4Control (Softing), Stardom (Yokogawa), Ovation (Westinghouse-Emerson Process Management)... Hướng giải pháp này thể hiện nhiều ưu điểm về mặt giá thành, hiệu năng tính toán và tính năng mở. Một trạm điều khiển cục bộ chính là một máy tính cá nhân công nghiệp được cài đặt một hệ điều hành thời gian thực và các card giao diện bus trường và card giao diện bus hệ thống.

Trong giải pháp điều khiển dùng máy tính cá nhân thì một vấn đề thường rất được quan tâm là độ tin cậy của máy tính. Một phần ta có thể yên tâm bởi với cấu trúc vào/ra phân tán, máy tính điều khiển được đặt trong phòng điều khiển trung tâm với điều kiện môi trường làm việc tốt. Mặt khác, trên thị trường cũng đã có rất nhiều loại máy tính cá nhân công nghiệp, đảm bảo độ tin cậy cao không kém một PLC. Một khi máy tính chỉ được cài đặt hệ điều hành và phần mềm điều khiển thì khả năng gây lỗi do phần mềm cũng sẽ được giảm thiểu.

Tuy nhiên, đối với các ứng dụng có yêu cầu cao về tính sẵn sàng, độ tin cậy của hệ thống, ta cần có một giải pháp dự phòng thích hợp. Giải pháp đơn giản và tiết kiệm nhất là “dự phòng lạnh”, có nghĩa là trong trường hợp có sự cố tại

máy tính điều khiển xảy ra ta chỉ cần thay thế một máy mới với cấu hình và các phần mềm đã được cài đặt giống hệt máy chính. Song giải pháp tốt hơn là sử dụng một cấu hình dự phòng nóng.

3.3 Các vấn đề kỹ thuật

Các vấn đề kỹ thuật dưới đây đóng vai trò đặc biệt quan trọng khi nghiên cứu về các hệ điều khiển phân tán, sẽ được đề cập chi tiết trong các phần sau.

- Kiến trúc xử lý phân tán (*distributed processing*): Cấu trúc phân tán về mặt vật lý (địa lý) dẫn đến phân tán về mặt xử lý thông tin. *Xử lý phân tán* là một khái niệm vay mượn từ lĩnh vực tin học. Xử lý phân tán khác với *xử lý cục bộ* và khác với *xử lý nối mạng* ở tính thống nhất, xuyên suốt trong việc xây dựng ứng dụng và trao đổi dữ liệu giữa các trạm.
- Tính năng thời gian thực (*real-time*): Tính năng của một hệ thống luôn sẵn sàng phản ứng với các sự kiện bên ngoài và đưa ra đáp ứng một cách đúng đắn và kịp thời. Với kiến trúc xử lý phân tán, việc đáp ứng tính năng thời gian thực được cải thiện bởi khả năng xử lý thông tin tại chỗ. Song cũng nhiều vấn đề được đặt ra trong việc giao tiếp giữa các thành phần (*real-time interprocess communication*), trong đó vấn đề giao thức mạng đóng một vai trò quan trọng.
- Tính sẵn sàng (*availability*) và độ tin cậy (*reliability*): Một đặc điểm nổi bật so với các hướng giải pháp khác là tính sẵn sàng cao thông qua khả năng dự phòng tích hợp, có thể lựa chọn dự phòng cho từng thành phần. Tính sẵn sàng, phương pháp giao tiếp số, kiến trúc xử lý phân tán, phần mềm đóng gói, phần cứng chuẩn hóa công nghiệp, độ tích hợp cao giữa các thành phần phần cứng và phần mềm là các yếu tố giúp cho các hệ thống điều khiển phân tán có độ tin cậy cao.
- Hỗ trợ chuẩn (*standard support*): Thực ra, đây không phải là đặc điểm tiêu biểu của các hệ DCS truyền thống. Nhưng đây là một yêu cầu không thể thiếu được trong các hệ DCS mới. Đặc biệt, sự tương thích với các chuẩn công nghiệp là tiền đề cho tính năng mở, cho khả năng tương tác với các thiết bị của các hãng thứ ba.
- Công cụ phần mềm (*software tools*): Việc xây dựng các ứng dụng điều khiển được hỗ trợ bởi các công cụ “lập trình” hoặc “cấu hình” rất mạnh và các thư viện phần mềm đóng gói chuẩn, dựa theo các chuẩn quốc tế. Các công cụ phần mềm điều khiển giám sát cũng được tích hợp và sử dụng chung một cơ sở dữ liệu trong hệ thống. Khác với các giải pháp điều khiển đơn lẻ như PLC hoặc PC, ta không phải sử dụng một công cụ riêng, xây dựng riêng giao diện người-máy (HMI) và các chức năng SCADA khác. Quá trình tạo giao diện người- máy, tạo hệ thống cảnh báo, tạo công thức điều khiển,... nằm trong việc phát triển ứng dụng, đi đôi với việc xây dựng chương trình điều khiển cấp thấp.

4 XỬ LÝ THỜI GIAN THỰC VÀ XỬ LÝ PHÂN TÁN

4.1 Một số khái niệm cơ bản

4.1.1 Hệ thống thời gian thực

Một hệ thống thời gian thực là một hệ thống mà sự hoạt động tin cậy của nó không chỉ phụ thuộc vào sự chính xác của kết quả, mà còn phụ thuộc vào thời điểm đưa ra kết quả để phản ứng với sự kiện bên ngoài. Hệ thống có lỗi khi thời gian yêu cầu không được thoả mãn.

Một hệ thống thời gian thực có các đặc điểm tiêu biểu sau:

- Tính bị động: Hệ thống phải phản ứng với các sự kiện xuất hiện vào các thời điểm không biết trước.
- Tính nhanh nhạy: Hệ thống phải xử lý thông tin một cách nhanh chóng để có thể đưa ra kết quả phản ứng một cách kịp thời.
- Tính tiên định: Dự đoán trước được thời gian phản ứng tiêu biểu, thời gian phản ứng chậm nhất cũng như trình tự đưa ra các phản ứng.

Tuy tính nhanh nhạy là một đặc điểm tiêu biểu, nhưng một hệ thống có tính năng thời gian thực không nhất thiết phải có đáp ứng thật nhanh mà quan trọng hơn là phải có phản ứng kịp thời đối với các yêu cầu, tác động bên ngoài. Có thể nói, tất cả các hệ thống điều khiển là các hệ thống thời gian thực. Ngược lại, một số lớn các hệ thống thời gian thực là các hệ thống điều khiển. Một bộ điều khiển phải đưa ra được tín hiệu điều khiển kịp thời sau một thời gian nhận được tín hiệu đo để đưa quá trình kỹ thuật về trạng thái mong muốn. Một hệ thống truyền thông có tính năng thời gian thực phải có khả năng truyền tin một cách tin cậy và kịp thời đối với các yêu cầu của các đối tác truyền thông. Tính năng thời gian thực của một hệ thống điều khiển phân tán không chỉ phụ thuộc vào tính năng thời gian thực của từng thành phần trong hệ thống, mà còn phụ thuộc vào sự phối hợp hoạt động giữa các thành phần đó.

4.1.2 Xử lý thời gian thực

Xử lý thời gian thực là hình thức xử lý thông tin trong một hệ thống để đảm bảo tính năng thời gian thực của nó. Như vậy, xử lý thời gian thực cũng có các đặc điểm tiêu biểu nêu trên như tính bị động, tính nhanh nhạy và tính tiên định. Để có thể phản ứng với nhiều sự kiện diễn ra cùng một lúc, một hệ thống xử lý thời gian thực sử dụng các *quá trình tính toán* đồng thời.

Quá trình tính toán là một tiến trình thực hiện một hoặc một phần chương trình tuân tự do hệ điều hành quản lý trên một máy tính, có thể tồn tại đồng thời với các quá trình khác kể cả trong thời gian thực hiện lệnh và thời gian xếp hàng chờ đợi thực hiện.

Quá trình tính toán được chia thành hai loại:

- *Quá trình nặng cân (process)*: là quá trình tính toán có không gian địa chỉ riêng.
- *Quá trình nhẹ cân (thread)*: là quá trình không có không gian địa chỉ riêng.

Các hình thức tổ chức các quá trình tính toán đồng thời:

- *Xử lý cạnh tranh*: Nhiều quá trình tính toán chia sẻ thời gian xử lý thông tin của một bộ xử lý.
- *Xử lý song song*: Các quá trình tính toán được phân chia thực hiện song song trên nhiều bộ xử lý của một máy tính.
- *Xử lý phân tán*: Mỗi quá trình tính toán được thực hiện riêng trên một máy tính.

Trong các hệ thống điều khiển, khái niệm *task* cũng hay được sử dụng bên cạnh quá trình tính toán. Có thể nói, task là một nhiệm vụ xử lý thông tin trong hệ thống, có thể thực hiện theo cơ chế tuần hoàn (periodic task) hoặc theo sự kiện (event task). Ví dụ, một task thực hiện nhiệm vụ điều khiển cho một hoặc nhiều mạch vòng kín có chu kỳ trích mẫu giống nhau. Hoặc, một task có thể thực hiện nhiệm vụ điều khiển logic, điều khiển trình tự theo các sự kiện xảy ra. Task có thể thực hiện dưới dạng một quá trình tính toán duy nhất, hoặc một dãy các quá trình tính toán khác nhau.

4.1.3 Hệ điều hành thời gian thực

Các trạm điều khiển trong một hệ điều khiển phân tán bao giờ cũng hoạt động dựa trên nền một hệ điều hành thời gian thực. Hệ điều hành thời gian thực là một hệ điều hành hỗ trợ các chương trình ứng dụng xử lý thời gian thực. Bản thân hệ điều hành thời gian thực cũng là một hệ thời gian thực theo đúng nghĩa của nó, vì vậy cũng có các đặc điểm tiêu biểu đã đề cập. Một hệ điều hành thời gian thực bao giờ cũng là một hệ đa nhiệm (multi-tasking), hỗ trợ xử lý cạnh tranh hoặc/và xử lý song song. Lập lịch, đồng bộ hóa quá trình và giao tiếp liên quá trình là các khái niệm quan trọng trong một hệ điều hành thời gian thực.

Phương pháp lập lịch (Scheduling)

Việc lập lịch thực hiện cho các task có thể được thực hiện theo hai cách:

- *Lập lệnh tĩnh*: thứ tự thực hiện các quá trình tính toán không thay đổi mà được xác định trước.
- *Lập lệnh động*: hệ điều hành xác định lệnh trước hoặc sau khi quá trình tính toán đã bắt đầu.

Tuy nhiên, ta cần có một *sách lược lập lệnh* (strategy) để áp dụng đối với từng tình huống cụ thể. Có thể chọn một trong những cách sau:

- *FIFO (First In First Out)*: một tiến trình đến trước sẽ được thực hiện trước.
- *Mức ưu tiên cố định/động*: tại cùng một thời điểm, các tiến trình được đặt các mức ưu tiên cố định hoặc có thể thay đổi nếu cần.

- *Preemptive*: còn gọi là chen hàng, tức là chọn một tiến trình để thực hiện trước các tiến trình khác.
- *Non - Preemptive*: không chen hàng. Các tiến trình được thực hiện bình thường dựa trên mức ưu tiên của chúng.

Việc tính mức ưu tiên của mỗi tiến trình được thực hiện theo một trong số các *thuật toán lập lịch* sau:

- *Rate monotonic*: càng thường xuyên càng được ưu tiên.
- *Deadline monotonic*: càng gấp càng được ưu tiên.
- *Least laxity*: tỷ lệ thời gian tính toán/thời hạn cuối cùng(deadline) càng lớn càng được ưu tiên.

Đồng bộ hoá quá trình

Khi các quá trình tính toán cùng sử dụng một tài nguyên loại trừ lẫn nhau như một vùng nhớ, cổng vào/ra, hoặc chúng phụ thuộc lẫn vào nhau ví dụ quá trình 1 chờ kết quả của quá trình 2... sẽ rất dễ dẫn đến tình trạng *tắc nghẽn* (Deadlock), hay tạo ra một *tình huống chạy đua* (Race Condition). Do vậy việc đồng bộ hoá các quá trình là điều cần thiết.

Có thể thực hiện việc này theo các phương pháp sau:

- Mutex (Mutual exclusion)
- Critical Section
- Semaphore
- Monitor

Giao tiếp liên quá trình

Giao tiếp liên quá trình là giao tiếp giữa các quá trình tính toán thuộc cùng một hệ điều hành trên một máy. Có hai loại:

- Giữa các Thread thuộc cùng một Process: sử dụng các biến toàn cục.
- Giữa các Process khác nhau hoặc giữa các Thread thuộc các Process khác nhau: sử dụng các phương pháp như shared memory, slot, pipes, mailbox, files...

4.1.4 Xử lý phân tán

Xử lý phân tán giúp nâng cao năng lực xử lý thông tin của một hệ thống, góp phần cải thiện tính năng thời gian thực, nâng cao độ tin cậy và tính linh hoạt của hệ thống.

Phân biệt các khái niệm:

- Xử lý cục bộ - ứng dụng đơn độc
- Xử lý cạnh tranh - ứng dụng đa nhiệm
- Xử lý tập trung - ứng dụng tập trung
- Xử lý nối mạng - ứng dụng mạng
- Xử lý phân tán - ứng dụng phân tán

Cần phân biệt rõ giữa ứng dụng mạng và ứng dụng phân tán. Trong một ứng dụng mạng, các chương trình trên mỗi trạm tồn tại hoàn toàn độc lập với nhau và việc giao tiếp giữa chúng được thực hiện qua cơ chế “hiện” (*explicit communication*). Web là một ứng dụng mạng tiêu biểu. Trong một ứng dụng phân tán, các chương trình trên các trạm hợp tác chặt chẽ với nhau thông qua cơ chế giao tiếp ngầm (*implicit communication*) để cùng thực hiện một nhiệm vụ tổng thể của hệ thống. Chức năng điều khiển trong một hệ điều khiển phân tán được thực hiện dưới dạng một ứng dụng phân tán.

Các vấn đề của xử lý phân tán

- Phân chia và phối hợp nhiệm vụ
- Giao tiếp giữa các trạm
- Đồng bộ hóa các quá trình xử lý phân tán
- Dự phòng, khắc phục lỗi

4.2 Các kiến trúc xử lý phân tán

Kiến trúc Master/Slave

- Các chức năng xử lý thông tin được phân chia trên nhiều trạm tớ
- Một trạm chủ phối hợp hoạt động của nhiều trạm tớ
- Các trạm tớ có vai trò, nhiệm vụ tương tự như nhau (tuy với các đối tượng khác nhau)
- Các trạm tớ có thể giao tiếp trực tiếp, hoặc không
- Ví dụ tiêu biểu: Ứng dụng điều khiển sử dụng bus trường, trạm điều khiển là trạm chủ, các vào/ra từ xa hoặc thiết bị trường là các trạm tớ.

Kiến trúc Client/Server

- Chức năng xử lý thông tin được phân chia thành hai phần khác nhau, phần sử dụng chung cho nhiều bài toán được thực hiện trên các server, phần riêng thực hiện trên từng client.
- Giữa các client không cần thiết có giao tiếp trực tiếp
- Vai trò chủ động trong giao tiếp thuộc về client
- Ví dụ tiêu biểu: Trong cấp điều khiển giám sát, có thể sử dụng một trạm chủ cho việc thu thập và quản lý, lưu trữ dữ liệu và cảnh giới báo động, các trạm vận hành là thực hiện giao diện người-máy với vai trò là client.

Kiến trúc bình đẳng

- Các trạm có vai trò bình đẳng, phối hợp hoạt động trực tiếp với nhau không qua trung gian
- Ví dụ tiêu biểu: Trong cấp điều khiển, các trạm điều khiển cục bộ phân chia thực hiện chức năng điều khiển cho cả dây chuyền sản xuất.

Kiến trúc tự trị

- Các trạm có vai trò bình đẳng, có thể hoạt động tương đối độc lập nhưng sự phối hợp hoạt động tạo hiệu quả cao nhất.
- Ví dụ tiêu biểu: Kiến trúc điều khiển thông minh các hệ thống đèn tín hiệu giao thông.

4.3 Cơ chế giao tiếp

Dữ liệu toàn cục (Global Data)

- Giống như một vùng nhớ chung
- Mỗi trạm đều chứa một ảnh của bảng dữ liệu toàn cục, trong đó có toàn bộ dữ liệu cần trao đổi của tất cả các trạm khác
- Mỗi trạm gửi phần dữ liệu của nó tới tất cả các trạm, mỗi trạm tự cập nhật ảnh của bảng dữ liệu toàn cục
- Đơn giản, tiên định nhưng kém hiệu quả
- Áp dụng cho lượng dữ liệu nhỏ, tuần hoàn, thích hợp trong kiến trúc bình đẳng (ví dụ giữa các trạm điều khiển).

Hỏi tuần tự (Polling, Scanning)

- Một trạm đóng vai trò Master
- Cơ chế hỏi/đáp tuần tự
- Đơn giản, tiên định, hiệu quả cao
- Áp dụng cho trao đổi dữ liệu tuần hoàn, thích hợp trong kiến trúc Master/Slave

Tay đôi (Peer-To-Peer)

- Hình thức có liên kết hoặc không liên kết, cấu hình trước hoặc không cấu hình trước, có xác nhận hoặc không xác nhận, có yêu cầu hoặc không có yêu cầu
- Linh hoạt nhưng thủ tục có thể phức tạp
- Áp dụng cho trao đổi dữ liệu tuần hoàn hoặc không tuần hoàn, thích hợp cho tất cả các kiến trúc khác nhau.

Chào/đặt hàng (Subscriber/Publisher)

- Nội dung thông báo được một trạm chủ chào và các trạm client đặt theo cơ chế tuần hoàn hoặc theo sự kiện
- Thông báo chỉ được gửi tới các trạm đặt (có thể gửi riêng hoặc gửi đồng loạt)
- Linh hoạt, tiên định, hiệu suất cao
- Áp dụng cho trao đổi dữ liệu tuần hoàn hoặc không tuần hoàn, thích hợp cho kiến trúc Client/Server hoặc kiến trúc bình đẳng.

Hộp thư (Mailbox)

- Các trạm sử dụng một môi trường trung gian như files, một cơ sở dữ liệu hoặc một chương trình server khác để ghi và đọc dữ liệu
- Mỗi bức thư mang dữ liệu và mã căn cước (nội dung thư hoặc/và người nhận)
- Gửi và nhận thư có thể diễn ra tại bất cứ thời điểm nào
- Linh hoạt nhưng kém hiệu quả, khó đảm bảo tính năng thời gian thực
- Áp dụng cho trao đổi dữ liệu có tính chất ít quan trọng, thích hợp cho kiến trúc Client/Server hoặc kiến trúc tự trị.

4.4 Đồng bộ hóa trong xử lý phân tán

4.4.1 Đồng bộ hóa các tín hiệu vào/ra

Cấu trúc vào/ra phân tán sử dụng bus trường làm nảy sinh một vấn đề chưa được xét tới trong lý thuyết điều khiển số. Đó là sự không đồng bộ của các tín hiệu vào/ra do thời gian trễ từng kênh khác nhau, khó xác định.

Có hai cách giải quyết sau:

- Đặt cấu hình bus và chọn chu kỳ điều khiển sao cho chu kỳ bus nhỏ hơn nhiều so với chu kỳ điều khiển để có thể bỏ qua thời gian trễ từng kênh khác nhau.
- Sử dụng loại bus trường có hỗ trợ đồng bộ hóa các đầu vào/ra, ví dụ Profibus-DP.

Ví dụ, các lệnh dưới đây được sử dụng trong Profibus-DP để đồng bộ hóa các đầu vào/ra:

- FREEZE: hi nhận được lệnh này, các DP-Slave sẽ nhận dữ liệu đầu ra gửi từ DP-Master và sau đó sẽ không nhận dữ liệu đầu ra gửi từ DP-Master nữa cho đến khi kết thúc lệnh FREEZE.
- UNFREEZE: Lệnh này làm kết thúc lệnh FREEZE. Các DP-Slave sẽ tiếp tục nhận dữ liệu đầu ra gửi từ DP-Master.
- SYNC: Khi nhận được lệnh này, tất cả hoặc một vài DP-Slave sẽ gửi dữ liệu tới DP-Master và sau đó DP-Master sẽ không nhận dữ liệu đầu vào gửi từ DP-Slave nữa cho đến khi kết thúc lệnh SYNC.
- UNSYNC: Lệnh này làm ngừng lệnh SYNC. DP-Master sẽ tiếp tục nhận dữ liệu đầu vào gửi từ các DP-Slave.

4.4.2 Đồng bộ hóa thời gian

Giữa các trạm điều khiển cục bộ và các trạm vận hành cần có một sự đồng bộ hóa thời gian một cách chặt chẽ, vì đây là vấn đề liên quan hệ trọng tới tính chính xác và độ tin cậy của các thông tin điều khiển, vận hành, thông báo báo động.

Để đồng bộ hoá thời gian trong một hệ điều khiển phân tán, một trạm vận hành có thể được chọn làm qui chiếu, tất cả các trạm khác nối với bus hệ thống được đồng bộ hoá thời gian theo trạm này thông qua các thông báo gửi đồng loạt. Trong một số hệ thống mạng có hỗ trợ trực tiếp việc đồng bộ hóa thời gian, người ta có thể chọn một thiết bị đặc chủng (*time master*) phục vụ mục đích này. Ta có thể định nghĩa 2 trạm vận hành làm qui chiếu nhưng tại một thời điểm chỉ có một trạm mang tín hiệu đồng bộ hoá, nếu trạm đó bị lỗi thì trạm còn lại tự động hoạt động.

Trong công nghiệp chế biến, khoảng thời gian chênh lệch cho phép giữa các trạm thường ở trong phạm vi $\pm 5\text{ms}$. Các thông báo thời gian cần gửi đồng loạt theo chu kỳ tối đa 1 phút.

5 CÔNG NGHỆ ĐỐI TƯỢNG TRONG ĐIỀU KHIỂN PHÂN TÁN

5.1 Lập trình hướng đối tượng

Lập trình hướng đối tượng được coi là phương pháp lập trình chuẩn hiện nay bởi nó có nhiều ưu điểm lớn so với các phương pháp cổ điển. Mục tiêu mà lập trình hướng đối tượng đặt ra là:

- Đơn giản hoá việc xây dựng và sử dụng các thư viện
- Cho phép dùng lại mã. Nếu hàm thư viện không phù hợp với yêu cầu của người lập trình thì người lập trình có khả năng sửa đổi dễ dàng mà không cần tìm hiểu ngọn nguồn, không cần phải có mã nguồn của hàm đó trong tay. Mã sinh ra từ thực nghiệm dễ dàng được dùng lại trong mã chính thức. Nói khác đi, người lập trình có điều kiện để thoải mái sáng tạo.
- Cải thiện khả năng bảo trì của mã, mã phải dễ hiểu, dễ sửa đổi. Trên thực tế, việc biên soạn tài liệu bao giờ cũng đi sau khá xa so với mã được viết ra.
- Cho phép tạo ra chương trình dễ mở rộng. Có thể thêm chức năng cho chương trình mà không ảnh hưởng dây chuyền đến mã đã viết. Mã đang có là mô hôi, là tiền bạc, không thể trả giá đắt cho mỗi chức năng thêm vào.

Lập trình hướng đối tượng phải được thực hiện thông qua một ngôn ngữ lập trình hướng đối tượng. Để đạt được các mục tiêu trên, mọi ngôn ngữ lập trình hướng đối tượng đều thể hiện ba khái niệm: đóng gói (*encapsulation*, *packaging*), đa hình (*polymorphism*) và thừa kế (*inheritance*). Các ngôn ngữ lập trình hướng đối tượng thông dụng là C++, Java, Ada...

5.2 Phân tích và thiết kế hướng đối tượng

Theo dòng phát triển của công nghệ công tin, phương pháp lập trình đã tiến hoá từ lập trình không có cấu trúc lên lập trình có cấu trúc và tới nay là lập trình hướng đối tượng. Phương pháp phân tích, thiết kế phần mềm cũng đi theo các bước tiến hoá này. Trước đây, người ta phân tích, thiết kế phần mềm theo kiểu hướng thủ tục (*procedure-oriented*) hoặc hướng dữ liệu (*data-oriented*). Theo phương pháp này, phần mềm cần xây dựng được chia thành giải thuật và cấu trúc dữ liệu. Trong quá trình phân tích, giải thuật được phân chia thành các giải thuật con đơn giản hơn, cấu trúc dữ liệu lớn được chia thành những cấu trúc nhỏ hơn. Quá trình tương tự cũng được tiến hành trong quá trình thiết kế.

Phương pháp phân tích, thiết kế hướng thủ tục hoặc hướng dữ liệu có ưu điểm đơn giản, nhanh chóng tạo ra kết quả (do tiến hành theo kiểu từ trên xuống) nhưng kết quả tạo ra không phản ánh bản chất của thể giới thực dẫn

đến cùng nhắc, khó thay đổi khi yêu cầu đặt ra thay đổi, khó mở rộng khi hệ thống phát triển.

Phương pháp phân tích, thiết kế phần mềm tiên tiến hiện nay là hướng đối tượng (*object-oriented*), trong đó khối cơ bản để xây dựng nên phần mềm là *đối tượng* hay *lớp*. Nói một cách đơn giản, đối tượng là sự phản ánh thế giới tự nhiên xung quanh. Ví dụ nếu trong hệ thống điều khiển có các thiết bị vào/ra số/tương tự như AI, AO, DI, DO thì trong phần mềm cũng có các lớp AI, AO, DI, DO ; trong hệ thống điều khiển có khâu điều khiển PID thì trong phần mềm cũng có lớp PID,...

Trong các hệ thống điều khiển, các đối tượng có thể đại diện các thành phần hệ thống như:

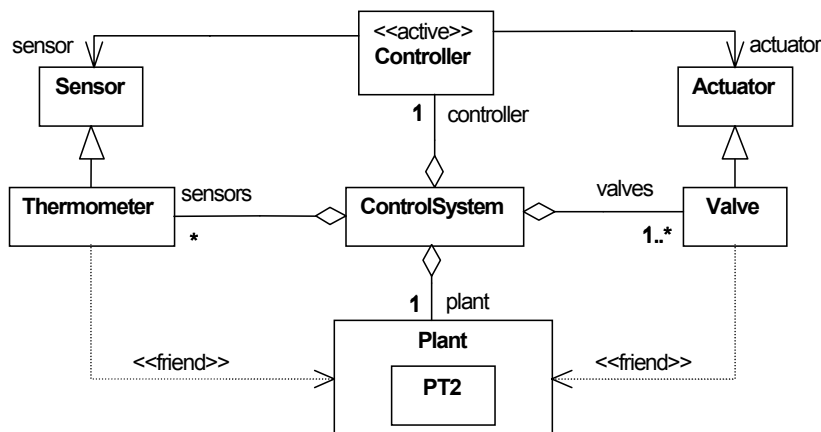
- Các thuật toán điều khiển
- Xử lý sự kiện và báo động
- Xử lý mệnh lệnh
- Quan sát và chẩn đoán
- Cấu hình vào/ra
- Mô phỏng
- Thông tin thiết kế

Việc trừu tượng hoá thế giới tự nhiên thành các lớp đối tượng như vậy được gọi là mô hình hoá hướng đối tượng. Dựa trên mô hình đối tượng thu được, phương pháp phân tích, thiết kế phần mềm hướng đối tượng sẽ bổ sung thêm các liên kết và lớp đối tượng mới, tinh chỉnh lại,... để tạo ra một mô hình đối tượng chi tiết của phần mềm. Cuối cùng, người lập trình sử dụng một ngôn ngữ lập trình nào đó (không nhất thiết phải là ngôn ngữ hướng đối tượng) thể hiện mô hình đối tượng chi tiết thành mã nguồn. Ưu điểm lớn nhất của phân tích, thiết kế phần mềm hướng đối tượng không phải nằm ở chỗ tạo ra chương trình nhanh tốn ít công sức, mà nằm ở chỗ nó gần với thực tế và do đó thúc đẩy việc tái sử dụng lại những thành quả đã xây dựng được như mã lệnh hay bản thiết kế.

5.2.1 Ngôn ngữ mô hình hóa thống nhất UML

Để phục vụ cho công việc mô hình hoá vốn là cốt lõi của phân tích, thiết kế phần mềm hướng đối tượng, *ngôn ngữ UML (unified modeling language)* được sử dụng rộng rãi. UML - một chuẩn quốc tế được quản lý bởi tổ chức OMG (*object management group*) - là một ngôn ngữ đồ họa dùng để trực quan hóa, đặc tả, xây dựng và tư liệu hóa hệ thống thiên về phần mềm. UML đem lại cho người sử dụng phương pháp chuẩn để viết bản thiết kế hệ thống bao trùm từ những thứ cụ thể như các lớp viết bằng một ngôn ngữ lập trình nào đó, các thành phần phần mềm có thể tái sử dụng,... cho đến những yếu tố trừu tượng như chức năng của toàn bộ hệ thống. Vì lý do này, ngôn ngữ UML không chỉ được sử dụng để mô tả, xây dựng kiến trúc và thiết kế của các hệ thống phần mềm, mà còn là một công cụ mô hình hóa thích hợp cho các hệ thống kỹ

thuật nói chung và các hệ thống điều khiển nói riêng. Trên Hình 5-1 là một biểu đồ lớp UML, minh họa đơn giản các lớp đối tượng và quan hệ của chúng trong một hệ thống điều khiển.



Hình 5-1: Mô hình hóa một hệ thống điều khiển sử dụng UML

Có thể nói, UML là một ngôn ngữ mô hình hóa rất mạnh, đa năng. Tài liệu về UML có rất nhiều, ví dụ [1][2].

5.2.2 Mẫu thiết kế

Mẫu thiết kế là sự hình thức hoá của cách tiếp cận tới một vấn đề thường gặp trong ngữ cảnh cụ thể. Mỗi mẫu thiết kế mô tả một giải pháp cho một vấn đề thiết kế cụ thể trong một ngữ cảnh xác định. Giải pháp này đã được chứng minh là hợp lý, sử dụng nhiều lần đem lại kết quả tốt và do đó được trừu tượng hoá thành một mẫu. Nói một cách ngắn gọn, mẫu thiết kế là kinh nghiệm thiết kế đúc kết lại. Bằng cách dùng các mẫu thiết kế, người thiết kế không phải thiết kế hệ thống của mình từ đầu mà sử dụng lại kinh nghiệm đã có từ trước, dẫn đến chất lượng thiết kế tốt hơn, tăng tính tái sử dụng của bản thiết kế. Một số mẫu thiết kế tiêu biểu là *Abstract factory*, *Iterator*, *Prototype*, *Singleton* và *Template method*. Để xây dựng các hệ thống phân tán hiện đại, người ta sử dụng các mẫu thiết kế như *Proxy*, *Broker*, *Marshaling/Unmarshaling*, *Adapter* và *Interface Mapping*. Một tác phẩm được coi là kinh điển viết về đề tài này là [3].

5.2.3 Phần mềm khung

Phần mềm khung là một dạng phần mềm bao gồm thư viện và các mẫu thiết kế giúp người sử dụng dễ dàng tạo các chương trình ứng dụng bằng cách bổ sung các phần mã ứng dụng cụ thể vào các khung có sẵn. Điểm khác nhau giữa một phần mềm khung với một thư viện lớp hay một thư viện hàm đơn thuần là:

- Một thư viện chỉ là một tập hợp của các lớp hay hàm hoàn chỉnh phục vụ một mục đích ứng dụng nào đó. Mã của một thư viện lớp hay hàm được chương trình ứng dụng chủ động gọi.
- Một phần mềm khung chứa một số lớp chưa hoàn chỉnh, tức chưa sử dụng tạo thể nghiệm được ngay (lớp trừu tượng), mà bắt buộc phải dẫn xuất và bổ sung mã ứng dụng cụ thể. Việc xây dựng một chương trình ứng dụng phải tuân theo các mẫu thiết kế có sẵn. Không những chương trình ứng dụng gọi mã trong phần mềm khung, mà mã trong phần mềm khung cũng chủ động gọi mã ứng dụng.

Có thể so sánh một phần mềm khung như một khung nhà bê tông đã được đúc sẵn theo một thiết kế, người thi công cần bổ sung các phần tường bao, tường ngăn, cửa sổ... theo thiết kế đó, sử dụng các thư viện là các viên gạch, cánh cửa, tấm vách ngăn làm sẵn.

Một số ví dụ phần mềm khung tiêu biểu là MFC (*Microsoft Foundation Class*), Microsoft's COM (*Component Object Model*), Borland's VCL (*Virtual Component Library*).

5.3 Phần mềm thành phần

Phần mềm thành phần (*component software*) là một hướng đi mới, phát triển trên cơ sở phương pháp lập trình hướng đối tượng. Lập trình hướng đối tượng cho phép sử dụng lại phần mềm (dưới dạng các *class*) vào giai đoạn biên dịch (*compile-time*), trong khi phần mềm thành phần cho phép sử dụng lại phần mềm (dưới dạng các *component*) vào cả giai đoạn biên dịch và giai đoạn chạy (*run-time*). Do vậy, theo tư tưởng phần mềm thành phần, ngôn ngữ lập trình cũng như "lớp" là thứ yếu, giao diện mới là quan trọng. Nói như vậy tức là một thành phần phần mềm (*component*) là các phần mềm có thể được viết ở các ngôn ngữ khác nhau, đã được hoàn chỉnh, biên dịch và đóng gói, có các giao diện chuẩn để có thể sử dụng thuận tiện, linh hoạt trong nhiều ứng dụng khác nhau mà không cần biên dịch lại. Thậm chí trong một số trường hợp, việc sử dụng các thành phần phần mềm có sẵn không đòi hỏi lập trình. Ví dụ người soạn thảo một văn bản có thể sử dụng kết hợp các thành phần phần mềm có sẵn như trình soạn thảo công thức, vẽ đồ thị, ... mà cảm tưởng như tất cả đều nằm trong chương trình soạn thảo văn bản.

Một số ví dụ mô hình phần mềm thành phần tiêu biểu là:

- Delphi VC
- JavaBeans
- Visual Basic VBX
- ActiveX-Controls

Có thể nói, hầu hết các hệ thống phát triển ứng dụng trong các hệ điều khiển phân tán hiện nay thực hiện triệt để tư tưởng hướng đối tượng và phần

mềm thành phần. Tư tưởng sử dụng khối hàm, các khối đồ họa, các khối chương trình trong nhiều hệ thống là những ví dụ tiêu biểu.

5.4 Đối tượng phân tán

Đối tượng phân tán cũng là một hướng phát triển tự nhiên từ phương pháp luận hướng đối tượng, bên cạnh phần mềm thành phần. Trong khi phần mềm thành phần quan tâm tới việc đóng gói các đối tượng để có thể sử dụng lại một cách thuận tiện, thì đối tượng phân tán tập trung vào vấn đề kiến trúc các đối tượng có khả năng giao tiếp một cách trong suốt trên các nền và hệ thống mạng khác nhau (*giao tiếp ngầm*). Cũng giống như phần mềm thành phần, một đối tượng phân tán có thể thực hiện ở một ngôn ngữ bất kỳ, nhưng nó phải có các giao diện theo một chuẩn nào đó để có thể hợp tác với nhau liên quá trình và xuyên mạng một cách đơn giản như hai đối tượng trong một chương trình. Nói như vậy, một đối tượng phân tán cũng được sử dụng khi đã biên dịch, đóng gói hoàn chỉnh dưới dạng một *server*. Tuy nhiên, việc sử dụng chúng có thể vẫn đòi hỏi phải lập trình phía *client* (một đối tượng phân tán hoặc một chương trình ứng dụng thông thường). Ngày nay, đối tượng phân tán và phần mềm thành phần đã gặp nhau ở nhiều điểm, ví dụ trong công nghệ COM/DCOM/ActiveX.

Nói tóm lại, một đối tượng phân tán là một đối tượng phần mềm trong một hệ thống phân tán, có thể được sử dụng bởi các chương trình ứng dụng hoặc các đối tượng khác thuộc cùng một quá trình tính toán, thuộc một quá trình tính toán khác hoặc thuộc một trạm khác trong mạng theo một phương thức thống nhất thông qua giao tiếp ngầm (không để ý tới giao thức truyền thông cụ thể, trong suốt với hệ điều hành, kiến trúc phần cứng và hệ thống mạng). Một đối tượng phân tán có các thuộc tính có thể truy cập được từ xa, có các phép toán có thể gọi được từ xa.

Mỗi đối tượng phân tán (*distributed object*) - bất kể dạng thực hiện, nền triển khai và vị trí cài đặt - đều có căn cước phân biệt và có thể được sử dụng như các đối tượng nội trình (*in-process object*). Lợi thế quyết định ở đây là việc tạo dựng một ứng dụng phân tán được thực hiện ở mức trừu tượng cao hơn so với kiểu lập trình mạng cổ điển, nhờ vậy trên nguyên tắc không khác biệt so với tạo dựng một ứng dụng đơn độc (*stand-alone application*).

Để đạt được điều đó, ta cần sự hỗ trợ hữu hiệu của một phần mềm khung (*framework*). Hiện nay có hai mô hình chuẩn cho những công trình khung đó là DCOM và CORBA. CORBA cho phép sử dụng một cách rộng rãi và linh hoạt hơn, trong khi DCOM hiện nay hầu như chỉ sử dụng được trên các hệ Microsoft Windows (95, 98, NT, 2000).

6 KIẾN TRÚC ĐỐI TƯỢNG PHÂN TÁN

Một kiến trúc đối tượng phân tán định nghĩa mô hình các đối tượng phân tán, mô hình giao tiếp và chuẩn giao tiếp giữa các đối tượng phân tán.

6.1 Yêu cầu chung

Một kiến trúc đối tượng phân tán tạo điều kiện cho việc lập trình ở một mức trừu tượng cao hơn so với phương pháp hướng đối tượng cổ điển. Cụ thể, điểm khác biệt so với lập trình hướng đối tượng cổ điển nằm ở “*tính trong suốt phân tán*” (*distribution transparency*), thể hiện qua các đặc tính sau:

- Trong suốt vị trí: Một client không cần biết rằng đối tượng server nằm trong cùng một quá trình tính toán, thuộc một quá trình tính toán khác trên cùng một trạm hoặc trên một trạm khác, cách sử dụng đối tượng server là thống nhất. Ngược lại cũng vậy.
- Trong suốt thể hiện: Client không cần quan tâm tới việc đối tượng server được thể hiện bằng ngôn ngữ lập trình nào và bằng phương pháp nào, mà chỉ cần quan tâm tới giao diện để có thể sử dụng.
- Trong suốt nền: Một client không cần biết rằng đối tượng server nằm trên hệ điều hành nào, trên nền máy tính kiến trúc ra sao.
- Trong suốt truyền thông: Mã thực hiện client và các đối tượng server không liên quan tới mạng truyền thông và giao thức truyền thông cụ thể.

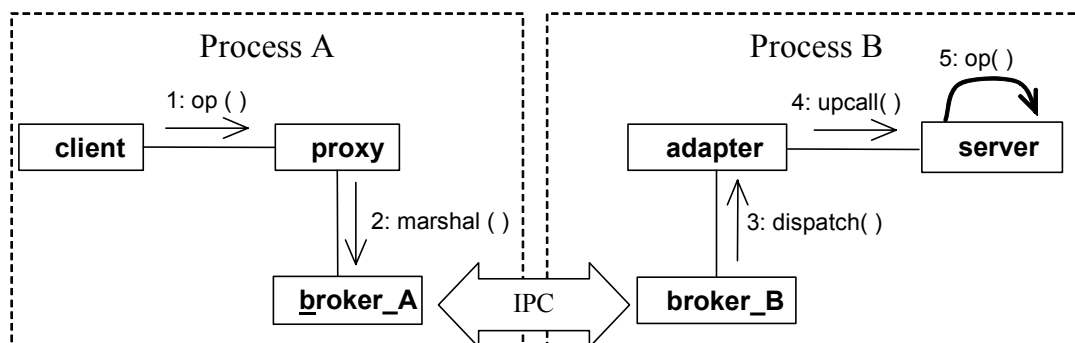
6.2 Các mẫu thiết kế

Để đáp ứng các yêu cầu trên, người ta thường áp dụng các mẫu thiết kế sau đây:

- *Proxy*: Một đối tượng đại diện cho server bên phía client, để client có thể sử dụng đối tượng server đơn giản như với một đối tượng nội trình (ví dụ thông qua con trỏ).
- *Broker*: Bộ phận che dấu chi tiết về cơ chế truyền thông cụ thể, tạo điều kiện cho client/proxy và server giao tiếp với nhau mà không phụ thuộc vào nền và hệ thống truyền thông bên dưới. Broker có mặt cả bên client và server.
- *Adapter*: Đối tượng trung gian, có vai trò thích ứng giao diện giữa broker và server, tạo điều kiện cho việc phát triển server một cách độc lập, cũng như sử dụng các server có sẵn (chưa tuân theo kiến trúc đối tượng phân tán).
- *Marshaling/Unmarshaling*: Cơ chế thực hiện mã hóa và đóng gói các lời gọi hàm bên client thành các bức điện tương ứng với cơ chế truyền thông cấp thấp, cũng mở gói và giải mã các bức điện thành lời gọi hàm chi tiết bên đối tượng server. Các phương thức tương tự cũng được thực hiện để

chuyển kết quả từ server trở lại client. Các nhiệm vụ này do proxy, server hoặc/và adapter đảm nhiệm.

- *Interface Mapping*: Giải quyết vấn đề trong suốt thể hiện bằng cách mô tả các giao diện bằng một ngôn ngữ độc lập IDL (*Interface Description Language*) và cho phép ánh xạ sang thực hiện bằng một ngôn ngữ cụ thể.

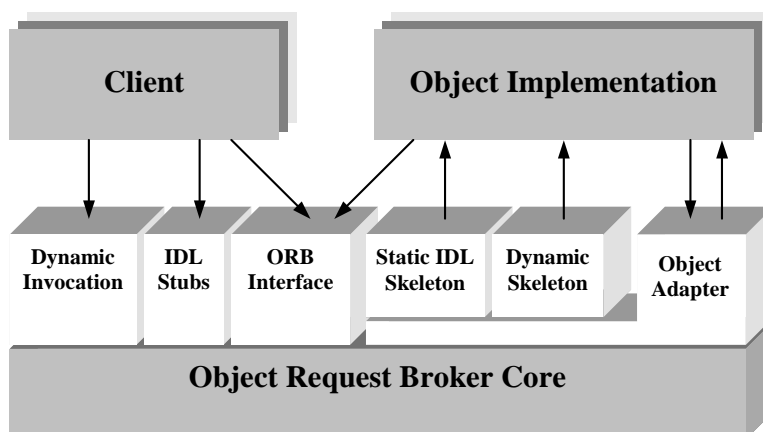


Hình 6-1: Các mẫu thiết kế giao tiếp giữa client và đối tượng server

6.3 Giới thiệu chuẩn CORBA

Chuẩn CORBA [4] do tổ chức OMG (*Object Management Group*) quản lý và phát triển. Đây là hiệp hội lớn nhất của các nhà phát triển, sản xuất và ứng dụng phần mềm trên thế giới, hiện nay có gần 1.000 thành viên. Chuẩn CORBA đưa ra một kiến trúc đối tượng phân tán cùng với các đặc tả ứng dụng cho nhiều lĩnh vực khác nhau, nhiều nền khác nhau và nhiều ngôn ngữ lập trình khác nhau. Vì tính trung lập của nó, CORBA được hỗ trợ rất rộng rãi, đặc biệt trong các hệ thống thông tin thương mại, phần mềm giao dịch kinh doanh và dịch vụ viễn thông. Tuy nhiên, cũng do tính độc lập của nó dẫn đến nhiều lý do mà CORBA không thực sự mạnh ở các hệ thống ứng qui mô vừa và nhỏ.

Hình 6-2 minh họa cấu trúc mô hình CORBA, trong đó bộ phận trừu tượng trung gian mang tên *Object Request Broker* (ORB) giữ vai trò quan trọng nhất (*broker*). ORB cho phép khách hàng (*client*) sử dụng dịch vụ của đối tượng phục vụ (*server*) mà không cần biết cụ thể dạng thực hiện, nền triển khai và vị trí cài đặt của đối tượng phục vụ. Kiến trúc ở đây được thực hiện theo các mẫu thiết kế đã trình bày trong phần trước.



Hình 6-2: Cấu trúc mô hình CORBA

6.4 Giới thiệu chuẩn COM/DCOM

COM (*Component Object Model*) là một mô hình đối tượng thành phần, một mô hình cơ sở cho nhiều công nghệ phần mềm quan trọng của hãng Microsoft. COM định nghĩa chuẩn nhị phân và đặc tả kết nối cho việc tương tác giữa các thành phần của một phần mềm với một thành phần khác trên cùng một quá trình tính toán, trên nhiều quá trình khác nhau hay trên các máy tính riêng biệt. Hãng Microsoft cũng hy vọng một ngày không xa COM cũng được sử dụng phổ biến trên các nền phần cứng và hệ điều hành khác nhau.

COM là một *mô hình lập trình cơ sở đối tượng* thiết kế để nâng cao sự tương tác giữa các thành phần phần mềm, nghĩa là, cho phép hai hoặc nhiều ứng dụng hay các thành phần dễ dàng giao tiếp với nhau cho dù chúng được viết bởi nhiều người khác nhau trong những khoảng thời gian khác nhau, bằng nhiều ngôn ngữ lập trình khác nhau thậm chí chạy trên các máy tính khác nhau, không hay cài đặt cùng một hệ điều hành. Để thực hiện điều này, COM định nghĩa và thực thi các kỹ thuật cho phép các ứng dụng kết nối với nhau như các *đối tượng phần mềm*.

Nói cách khác, COM đưa ra một mô hình tương tác mà qua đó một khách hàng (*client*) có thể kết nối với các nhà cung cấp dịch vụ (*object*) đó một cách thuận tiện. Với COM, các ứng dụng kết nối với nhau và với hệ thống qua các tập hợp của các lời gọi hàm - xem như là các phương thức hay những hàm thành viên, còn gọi là *giao diện*.

Theo cách tư duy COM, một giao diện là một “quy ước” kiểu mạnh giữa các thành phần phần mềm nhằm cung cấp những liên quan dù nhỏ nhưng hữu dụng tập các thao tác liên quan danh nghĩa. Một đối tượng được định nghĩa phù hợp với COM là một sự thể hiện đặc biệt của đối tượng. Một đối tượng COM giống như một đối tượng C++ nhưng khác ở chỗ một client không truy nhập trực tiếp vào đối tượng COM mà sẽ qua các giao diện mà đối tượng cung cấp.

6.4.1 Giao diện

Cách duy nhất để truy cập dữ liệu hoặc tác động lên một đối tượng COM là thông qua giao diện của nó. Một giao diện thực chất là một nhóm các hàm có sẵn liên quan với nhau. Có thể so sánh một giao diện với một lớp cơ sở trừu tượng chỉ gồm các hàm thuần ảo trong ngôn ngữ C++. Giao diện định nghĩa cú pháp các hàm thành viên, gọi là các phương thức (*methods*), kiểu trả về, số lượng và các kiểu tham số. Một giao diện không qui định cụ thể các phương thức đó được thực hiện như thế nào. Thực chất việc thể hiện giao diện là sử dụng con trỏ truy nhập vào một mảng các con trỏ khác và các con trỏ này trỏ tới các hàm của giao diện.

Thông thường, tên của giao diện được bắt đầu bằng chữ cái I, ví dụ như IUnknown, IData... Định danh thật của giao diện thể hiện ở chỉ danh GUID của nó, còn tên chỉ để thuận tiện cho việc lập trình và hệ thống COM sẽ sử dụng các chỉ danh này khi thao tác trên giao diện.

Thêm vào đó, khi giao diện có tên hoặc kiểu cụ thể và tên của các hàm thành viên, nó chỉ định nghĩa làm thế nào một client có thể sử dụng giao diện đó và những đáp ứng mong đợi từ đối tượng qua giao diện đó. Ví dụ, giao diện IStack với hai hàm thành viên PUSH và POP chỉ định nghĩa những thông số và kiểu trả về của hai hàm này và những gì chúng được mong đợi thực hiện từ client.

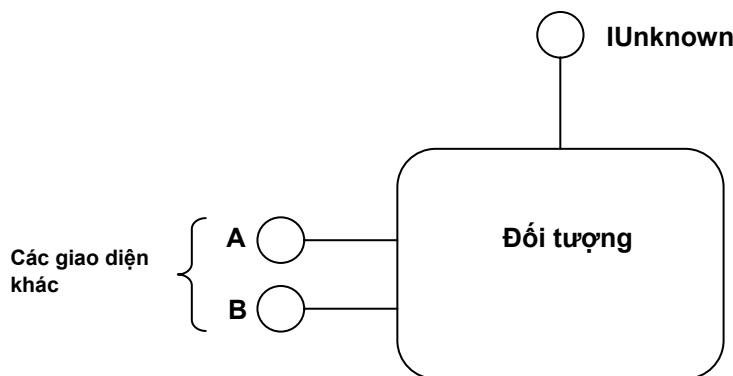
Có thể nói, giao diện là phương tiện để đối tượng đưa ra những dịch vụ của nó. Có bốn điểm quan trọng về giao diện mà ta cần biết:

- *Một giao diện không phải là một lớp* theo định nghĩa lớp thông thường bởi một lớp có thể được thể hiện qua một đối tượng còn một giao diện thì không bởi nó không kèm theo sự thực thi.
- *Một giao diện không phải là một đối tượng* bởi một giao diện chỉ đơn thuần là một nhóm các hàm liên quan và là chuẩn nhị phân mà qua đó client và object có thể giao tiếp với nhau. Còn đối tượng thì có thể thực thi trên nhiều ngôn ngữ với nhiều thể hiện của trạng thái bên trong, và do đó nó có thể cung cấp con trỏ đến các hàm thành viên của giao diện.
- *Giao diện có kiểu mạnh*: Mỗi một giao diện đều có một định danh riêng nên ngăn chặn được khả năng xung đột có thể xảy ra đối với các tên ta đặt cho giao diện. Điều này tăng thêm tính bền vững cho chương trình.
- *Các giao diện được phân biệt rõ ràng*: Mọi sự thay đổi như thêm hoặc xóa hàm thành viên, thay đổi ngữ nghĩa đều dẫn tới hình thành một giao diện mới và được gán một định danh mới. Do đó giao diện mới và cũ không thể xung đột với nhau cho dù mọi sự thay đổi chỉ đơn thuần là về ngữ nghĩa.

6.4.2 Đối tượng COM

Một đối tượng COM có thể được lập trình bằng một ngôn ngữ thông dụng như C, C++ hoặc VB. Một đối tượng có thể cung cấp nhiều giao diện. Tất cả

các đối tượng COM đều có một giao diện cơ bản là IUnknown. Đây là giao diện cơ sở cho tất cả các giao diện khác trong COM mà mọi đối tượng phải hỗ trợ. Bên cạnh đó, đối tượng cũng có khả năng thực thi nhiều giao diện khác. Các đối tượng với nhiều giao diện có thể cung cấp các con trỏ truy nhập vào nhiều bảng chứa các hàm. Các con trỏ này có thể gọi được con trỏ giao diện. Trong COM, giao diện là một bảng các con trỏ (giống như vtable trong C++) vào các hàm được thực hiện bởi đối tượng.



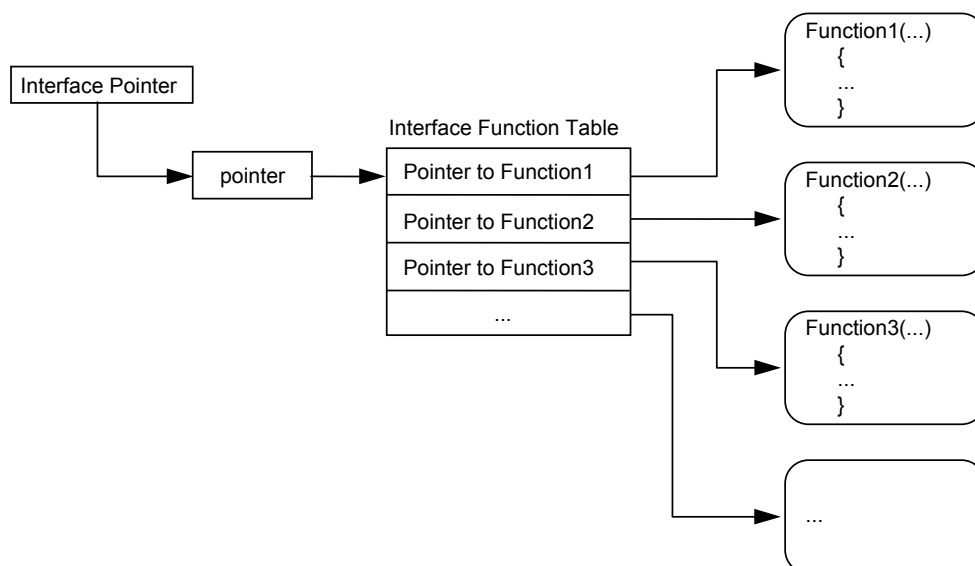
Hình 6-3: Mô hình một đối tượng COM

Trong thực tế, một con trỏ trỏ đến một giao diện là một con trỏ tới một con trỏ trỏ tới bảng các con trỏ vào các hàm thành viên. Tuy nhiên, để tránh cách diễn đạt quanh co này khi nói về giao diện người ta thường sử dụng thuật ngữ con trỏ giao diện để thay thế. Khi đó sự thực thi giao diện đơn giản là dùng con trỏ trỏ tới mảng các con trỏ tới các hàm. Hình 6-4 sau minh họa cho điều này.

Giao diện IUnknown

Như đã trình bày ở trên, mọi đối tượng COM, không có sự loại trừ, đều hỗ trợ giao diện IUnknown. Giao diện này có ba phương thức *AddRef()*, *Release()* và *QueryInterface()*. Tất cả các giao diện khác đều dẫn xuất từ giao diện IUnknown và đều có các con trỏ đến các phương thức này.

Hai phương thức đầu tính toán số đếm tham chiếu để điều khiển thời gian sống của đối tượng. Khi đối tượng được tạo lần đầu, ta cần gọi phương thức *AddRef()* của đối tượng để tăng số đếm. Khi không còn cần tới đối tượng, ta gọi phương thức *Release()* để giảm số đếm tham chiếu. Khi người dùng cuối cùng gọi phương thức *Release()*, số đếm giảm về 0 thì đối tượng sẽ tự hủy.



Hình 6-4: Sự thực thi con trỏ giao diện

Ta có thể thấy rõ vấn đề hơn qua sự thực thi đơn giản hai phương thức `IUnknown::AddRef()` và `IUnknown::Release()` dưới đây:

```

//tăng biến thành viên chứa số đếm tham chiếu
ULONG IUnknown::AddRef() {
    return ++m_RefCount;
}
//giảm biến chứa số đếm tham chiếu, nếu bằng 0 thì huỷ đối tượng
ULONG IUnknown::Release() {
    --m_RefCount;
    if (m_RefCount == 0) {
        delete this;
        return 0;
    }
    return m_RefCount;
}

```

Khi ta có một con trỏ đến đối tượng thì thực chất, những gì ta có chỉ là một con trỏ đến một trong số các giao diện của nó, còn đó là giao diện nào thì lại phụ thuộc vào cách mà ta có con trỏ đó. Từ con trỏ vào một giao diện, ta có thể truy cập được con trỏ vào các giao diện khác mà đối tượng hỗ trợ. Đối tượng có thể hoặc không hỗ trợ giao diện mà ta quan tâm, nhưng mọi đối tượng đều được đảm bảo hỗ trợ giao diện `IUnknown` nên ta có thể yêu cầu các giao diện khác qua phương thức `IUnknown::QueryInterface()`.

Các giao diện được định danh bởi các *IIDs* (*Interface IDs*) ví dụ như `IID_IUnknown` của giao diện `IUnknown`. Khi ta gọi phương thức `QueryInterface()`, ta gửi IID của giao diện mà ta cần cho nó và một con trỏ đến tham số đầu ra. Nếu đối tượng hỗ trợ giao diện yêu cầu, nó sẽ trả lại đoạn mã báo thành công `S_OK` (định nghĩa là 0) và đặt vào tham số đầu ra mà ta cung cấp con trỏ đến giao diện yêu cầu. Nếu đối tượng không hỗ trợ giao diện này nó báo lỗi và đặt tham số đầu ra là `NULL`. Ta xét một sự thực thi đơn giản sau:

```

HRESULT IUnknown::QueryInterface (REFIID riid , LPVOID *ppv) {
    //kiểm tra IID xem đối tượng có hỗ trợ giao diện yêu cầu không,

```

```

//nếu hỗ trợ ta tăng số đếm tham chiếu, đưa vào biến đầu ra cung cấp
// một con trỏ đến giao diện, và báo rằng đã thành công
    if (riid == IID_IUnknown)    {
        AddRef();
        *ppv = (LPVOID) this;
        return S_OK;
    }
//nếu không hỗ trợ giao diện ta đặt biến đầu ra cung cấp là NULL và
// trả về một mã thông báo lỗi
    else    {
        *ppv = NULL;
        return E_NOINTERFACE;
    }
}

```

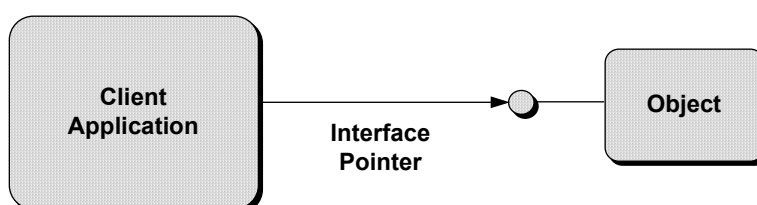
Quan hệ giữa đối tượng và các giao diện

- *Các client chỉ kết nối qua con trỏ tới các giao diện.* Khi một client truy nhập vào một đối tượng, client chỉ đơn thuần thông qua con trỏ giao diện. Con trỏ giấu đi nội dung của thao tác bên trong, ta không thể thấy chi tiết về đối tượng mà chỉ có thể thấy thông tin về trạng thái của chúng.
- *Đối tượng có thể thực thi nhiều giao diện.* Một lớp thực thi đối tượng có thể thực thi nhiều giao diện, ví dụ qua phương pháp đa thừa kế.

6.4.3 Giao tiếp giữa client và object

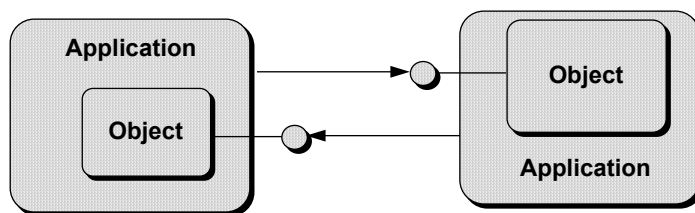
Cách thức của sự giao tiếp

Trước khi sử dụng một đối tượng COM trong một ứng dụng, ta cần khởi tạo cơ chế COM trong ứng dụng bằng lời gọi CoInitialize(...) và sau đó tạo đối tượng COM mong muốn. Client kết nối với object thông qua con trỏ giao diện và không bao giờ truy nhập trực tiếp vào object. Khi cần sử dụng dịch vụ nào đó của đối tượng, client hiểu rằng nó cần có con trỏ đến một hay nhiều giao diện của đối tượng. Để tạo một đối tượng COM và nhận một con trỏ vào giao diện, ta có thể gọi một trong hai hàm CoCreateInstance() hoặc CoCreateInstanceEx() với các tham số xác định đối tượng.



Hình 6-5: *Giao tiếp giữa đối tượng và khách hàng*

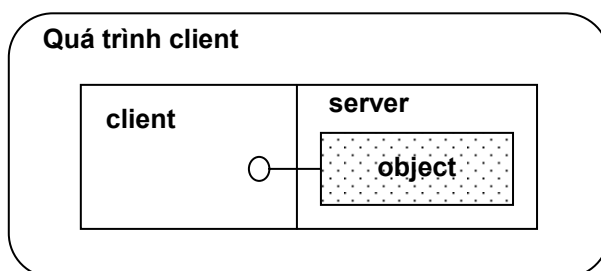
Trong một số trường hợp, bản thân client sẽ đóng vai trò một object và cung cấp cho các đối tượng khác những chức năng gọi các sự kiện hoặc đưa ra các dịch vụ của nó. Lúc này client là một đối tượng thực thi còn object là một khách hàng.



Hình 6-6: Giao tiếp giữa hai đối tượng

Giao tiếp trên cùng một quá trình tính toán

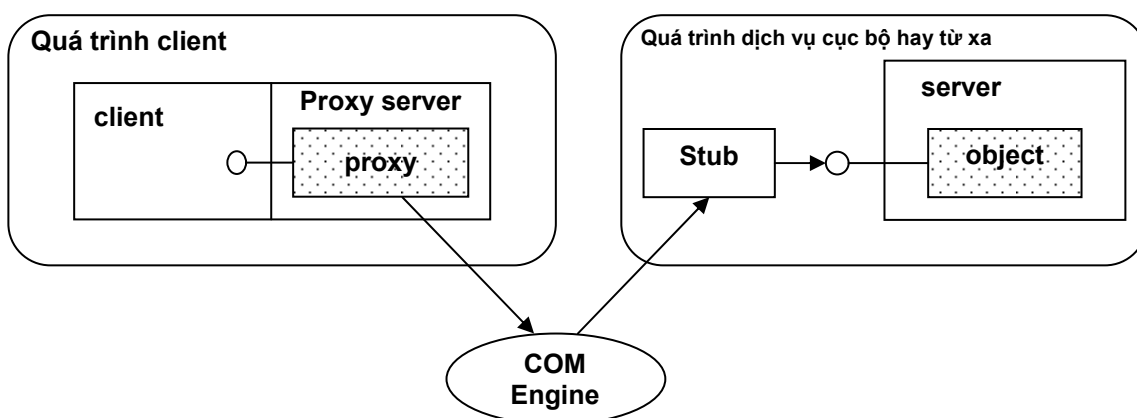
Khi client và đối tượng COM cùng nằm trên một quá trình tính toán thì client sẽ kết nối trực tiếp với object qua con trỏ giao diện.



Hình 6-7: Giao tiếp giữa đối tượng và khách hàng trên cùng quá trình

Giao tiếp liên quá trình

Nếu client và object không cùng nằm trên một không gian địa chỉ hay nằm trên các máy tính khác nhau thì COM sẽ thiết lập một đối tượng đại diện (proxy) bên phía client và một đối tượng gốc (stub) bên phía object. Proxy và stub sẽ kết nối với nhau qua kênh giao tiếp (channel). Khi đó, client sẽ thực hiện lời gọi dịch vụ trong không gian địa chỉ của nó tức là giao tiếp trực tiếp với proxy. Proxy sẽ thu thập (marshal) các thông số, gửi chúng đến stub qua kênh giao tiếp. Stub thực hiện lời gọi đến đối tượng dịch vụ, đóng gói kết quả và đưa về cho proxy.



Hình 6-8: Giao tiếp giữa đối tượng và khách hàng trên hai quá trình khác nhau

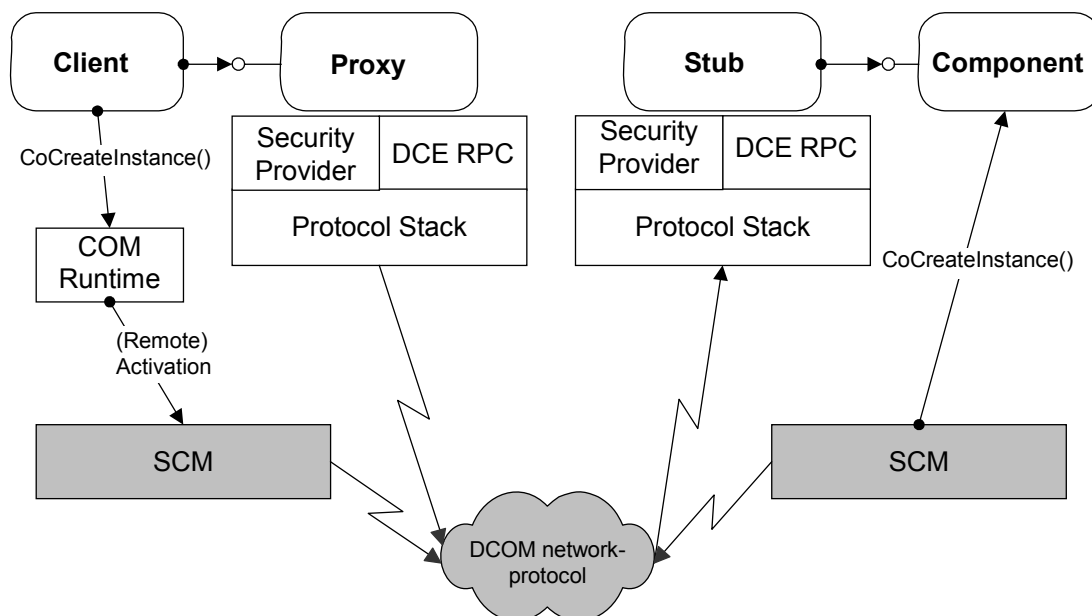
6.4.4 Ngôn ngữ mô tả giao diện

IDL (*Interface Description Language*) là một ngôn ngữ kiểu mạnh dùng để mô tả giao diện của đối tượng COM, độc lập với ngôn ngữ lập trình. Cú pháp của ngôn ngữ này không phức tạp so với một ngôn ngữ lập trình. Khi xây dựng một đối tượng COM, ta cần mô tả các phương thức của giao diện bằng cách sử dụng ngôn ngữ này. Sau khi tập xong file mô tả giao diện này, ta cần lưu nó ở dạng *.idl để chương trình dịch có thể hiểu được. Chương trình dịch (IDL-Compiler) sẽ dịch sang một ngôn ngữ lập trình, ví dụ C++. Khi đó một giao diện sẽ được chuyển sang thực hiện bằng một cấu trúc thích hợp trong ngôn ngữ lập trình, ví dụ một lớp thuần ảo trong C++.

6.4.5 Mô hình đối tượng thành phần phân tán DCOM

DCOM (Distributed COM) mở rộng COM cho việc giao tiếp giữa các đối tượng phân tán, thuộc các chương trình chạy trên nhiều máy tính khác nhau trên mạng LAN, WAN hay Internet. Với DCOM, các ứng dụng có thể phân tán trên nhiều vị trí đem lại sự thuận lợi cho chính ứng dụng. Ngày nay khi người ta nói tới COM là cũng thường bao hàm DCOM trong đó.

DCOM là một công nghệ lý tưởng cho những ứng dụng nhiều tầng lớp bởi vì nó cho phép những thành phần ActiveX làm việc ngang qua mạng. Nhiều người có thể phát triển thêm cùng một thành phần mà không cần phải lo lắng về lập trình mạng, tính tương thích hệ thống hoặc sự hợp nhất của những thành phần xây dựng từ những ngôn ngữ khác nhau. Nó dẫn tới hạ thấp giá thành và làm giảm sự phức tạp của việc phân tán các ứng dụng thành phần.



Hình 6-9: Giao tiếp giữa đối tượng và khách hàng trên hai máy khác nhau với DCOM

Khi các đối tượng ở trên các máy tính khác nhau, DCOM đơn giản thực hiện sự thay thế truyền thông liên quá trình cục bộ bởi giao thức mạng. Hình dưới đây minh họa rõ nét cách thức giao tiếp giữa các đối tượng nằm trên hai máy tính khác nhau.

Thư viện COM Run-Time cung cấp những dịch vụ hướng đối tượng tới khách hàng và thành phần muốn giao tiếp với nhau đồng thời sử dụng RPC và nhà cung cấp an toàn để tạo chuẩn nối mạng đóng gói tuân theo giao thức truyền thông cho DCOM.

Một ứng dụng client có thể tạo một đối tượng trên một máy tính khác qua hàm API `CoCreateInstance()`. Ta xét một ví dụ đơn giản sau:

```
HRESULT hr = CoCreateInstance(  
    CLSID_CData,          // định danh lớp của đối tượng yêu cầu  
    NULL,                 // tham số khởi tạo  
    CLSCTX_REMOTE_SERVER, // dịch vụ từ xa được yêu cầu  
    &si);                  // tham số đầu ra để chứa con trỏ giao diện
```


7 CÁC MÔ HÌNH ỨNG DỤNG ĐIỀU KHIỂN PHÂN TÁN

7.1 IEC-61131

IEC (*International Electrotechnical Commission*) là một tổ chức toàn cầu bao gồm các hội đồng ở các quốc gia. Mục tiêu của tổ chức này là thúc đẩy công việc chuẩn hoá trong lĩnh vực điện và điện tử.

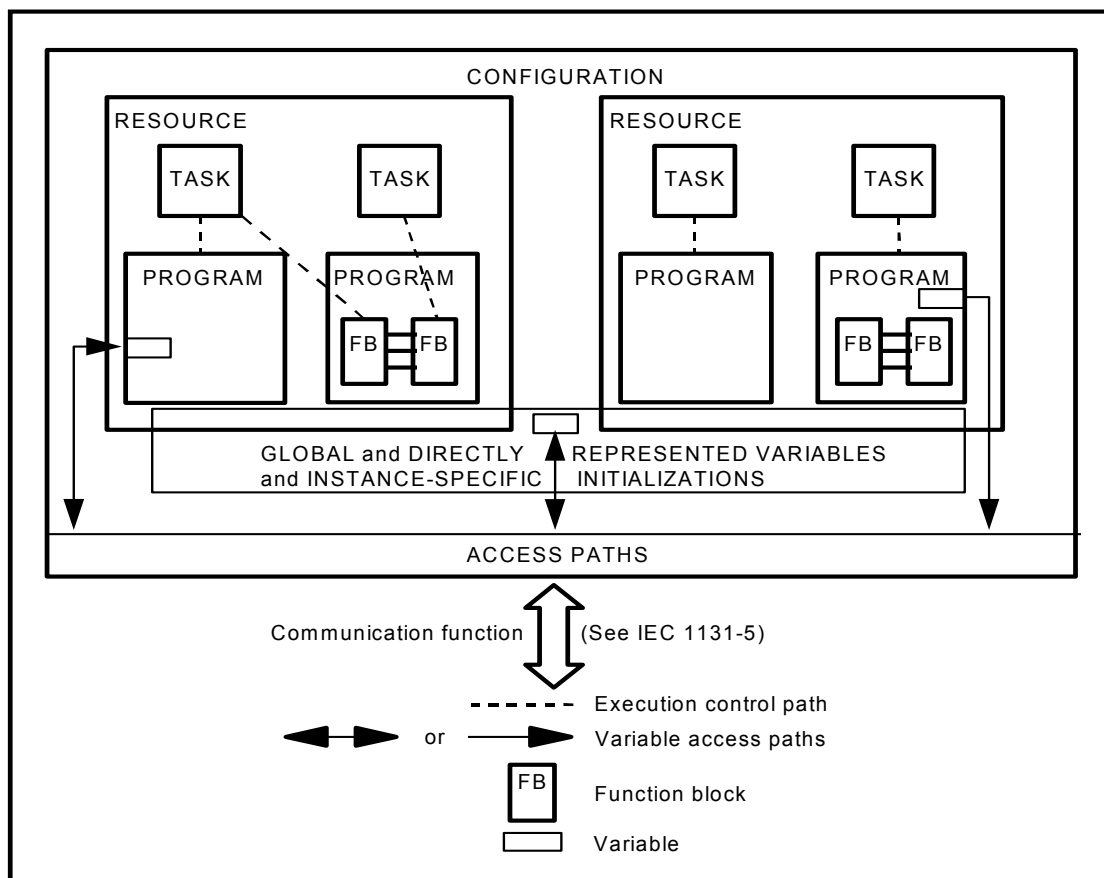
IEC 61131 là tiêu chuẩn về bộ điều khiển khả trình PLC và các thiết bị ngoại vi đi kèm với nó. Chuẩn IEC 61131 bao gồm 9 phần, trong đó các phần 1 đến 5 là quan trọng nhất:

- Phần 1 (*General Information*): Đưa ra các định nghĩa chung và các đặc tính chức năng tiêu biểu cho mỗi hệ thống điều khiển sử dụng PLC, ví dụ cơ chế thực hiện tuần hoàn, ảnh quá trình, thiết bị lập trình và giao diện người-máy.
- Phần 2 (*Equipment requirements*): Đặt ra các yêu cầu điện học, cơ học và chức năng cho các thiết bị; định nghĩa phương pháp kiểm tra và thử nghiệm các kiểu thiết bị tương ứng. Các yêu cầu được định nghĩa là nhiệt độ, độ ẩm, cung cấp nguồn, độ kháng nhiễu, phạm vi tín hiệu logic và sức bền cơ học của các thiết bị.
- Phần 3 (*Programming languages*): Định nghĩa các ngôn ngữ lập trình cho các thiết bị điều khiển khả trình. Ngoài ba ngôn ngữ kinh điển là biểu đồ hình thang (*Ladder Diagram*, LD), biểu đồ khối chức năng (*Function Block Diagram*, FBD) và liệt kê lệnh (*Instruction List*, IL), và một ngôn ngữ bậc cao kiểu văn bản có cấu trúc (*Structured Text*, ST) thì một phương pháp lập trình đồ họa phục vụ biểu diễn các thuật toán điều khiển trình tự là SFC (*Sequential Function Chart*) cũng đã được chuẩn hóa.
- Phần 4 (*Guidelines for users*): Đưa ra các nguyên tắc chỉ đạo cho người sử dụng trong các quá trình của một dự án, từ phân tích hệ thống cho tới lựa chọn thiết bị, vận hành và bảo trì hệ thống.
- Phần 5 (*Communication*): Đề cập tới phương pháp truyền thông giữa các PLC cũng như giữa PLC và một thiết bị khác trên cơ sở các khối hàm chuẩn. Các dịch vụ truyền thông này mở rộng chuẩn ISO/IEC 9506-1/2, thực chất là một tập con trong các dịch vụ được qui định trong MMS.

7.1.1 Mô hình phần mềm

Mỗi PLC tại một thời điểm bất kỳ chỉ có một *cấu hình* (configuration) nào đó. Một cấu hình bao gồm một hay nhiều *tài nguyên* (resource) bên trong đặc trưng cho khả năng xử lý tín hiệu của PLC. Mỗi tài nguyên bao gồm ít nhất một *chương trình* (program) hoạt động dưới sự điều khiển của *tác vụ* (task). Chương trình được xây dựng nên từ các *khối chức năng* (function block) hoặc các yếu tố ngôn ngữ khác (có cả thấy 5 ngôn ngữ lập trình được định nghĩa

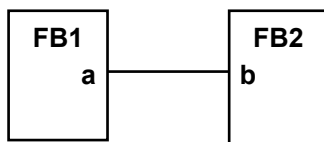
trong phần này). Các *biến toàn cục* (global variable) và *lối truy cập* (access path) là những cơ chế giao tiếp trong chương trình và giữa các tài nguyên với nhau.



Hình 7-1: Mô hình phần mềm theo IEC 61131-3

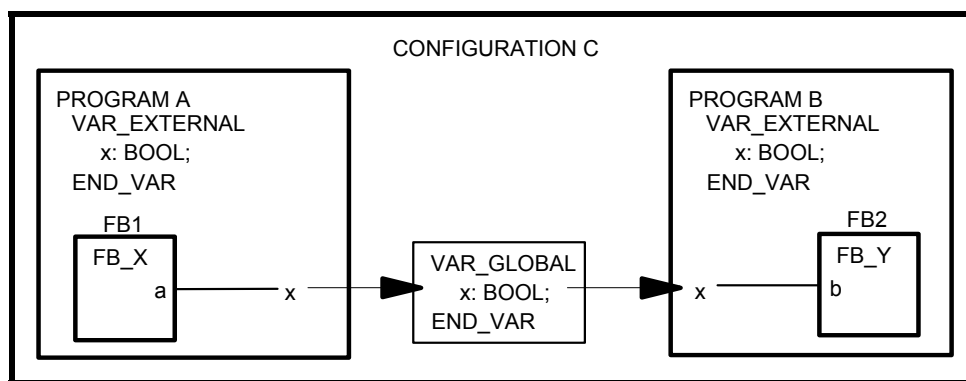
7.1.2 Mô hình giao tiếp

Như biểu diễn trong Hình 7-2, giá trị của biến có thể được truyền trực tiếp trong nội bộ chương trình bằng cách kết nối đầu ra của đơn vị ngôn ngữ này tới đầu vào của đơn vị ngôn ngữ khác. Mối liên kết này được biểu diễn một cách trực quan trong các ngôn ngữ đồ họa hoặc ẩn trong các ngôn ngữ văn bản.



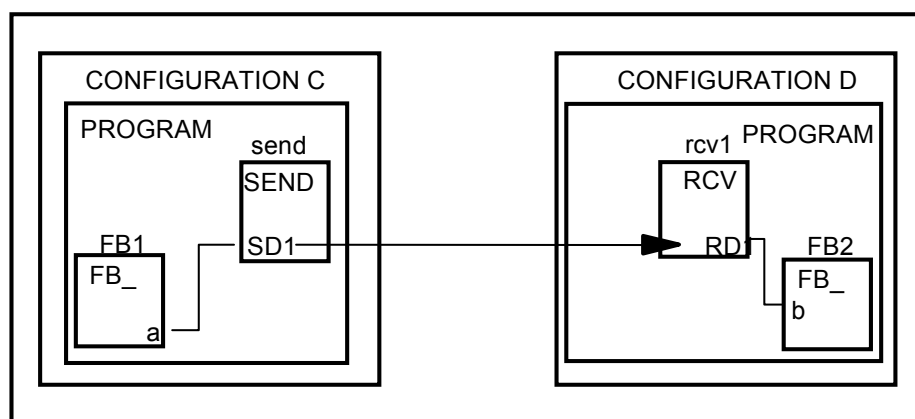
Hình 7-2: Giao tiếp bên trong chương trình

Giá trị của biến cũng có thể được truyền giữa các chương trình trong cùng một cấu hình thông qua biến toàn cục kiểu như biến *x* biểu diễn trong Hình 7-3. Biến này được khai báo là EXTERNAL đối với tất cả các chương trình sử dụng nó.



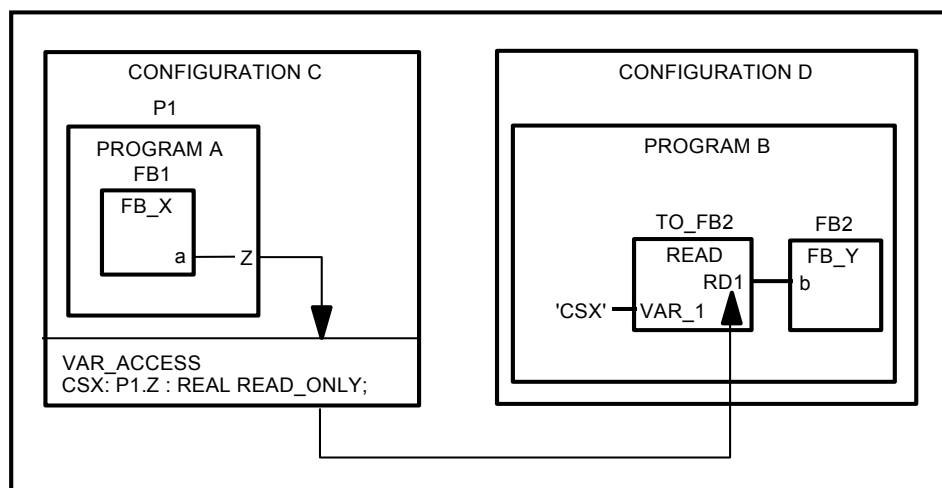
Hình 7-3: Giao tiếp giữa các chương trình trong cùng một cấu hình bằng biến toàn cục

Hình 7-4 biểu diễn cách giao tiếp đa năng nhất thông qua khối chức năng giao tiếp. Chi tiết về loại khối chức năng này được mô tả trong IEC 61131-5, ở đây chỉ lưu ý rằng, sử dụng khối chức năng truyền thông, giá trị của biến có thể được truyền giữa các bộ phận của chương trình, giữa các chương trình trong cùng một cấu hình hay trong các cấu hình khác nhau, thậm chí giữa chương trình chạy trong PLC với các hệ thống khác không phải PLC.



Hình 7-4: Giao tiếp qua khối chức năng

Sau cùng, bộ điều khiển khả trình và hệ thống không phải PLC có thể truyền dữ liệu qua lại thông qua lối truy cập như biểu diễn trong Hình 7-5, sử dụng các cơ chế định nghĩa trong IEC 61131-5.



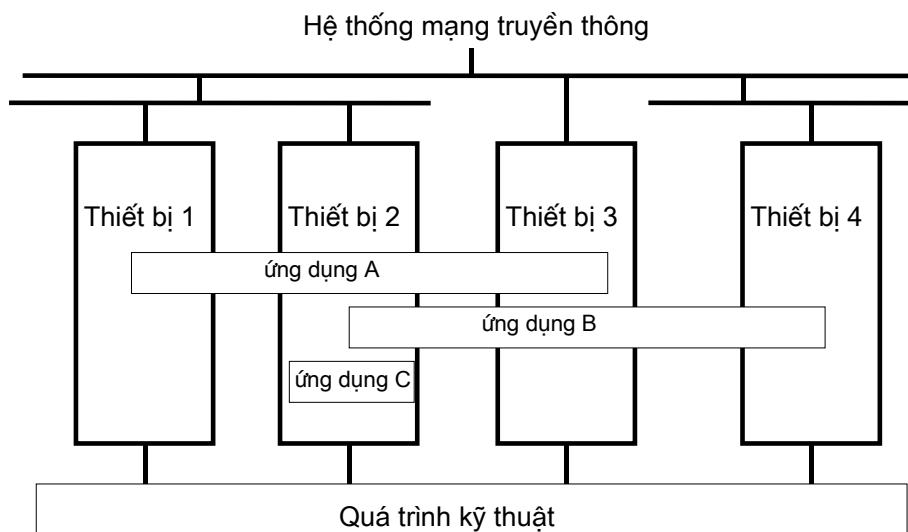
Hình 7-5: Giao tiếp qua đường dẫn truy cập

7.2 IEC-61499

IEC 61499 là tiêu chuẩn liên quan đến việc sử dụng các khối chức năng với tư cách là các module phần mềm trong hệ thống đo lường - điều khiển phân tán. Đứng từ quan điểm hướng đối tượng, mỗi khối chức năng được coi là một đối tượng phân tán với một chức năng trong hệ thống. Chuẩn IEC 61499 bao gồm 3 phần trong đó phần 1 là quan trọng nhất. IEC 61499-1 định nghĩa kiến trúc chung của hệ thống đo lường - điều khiển, ở đó khối chức năng đóng vai trò cốt lõi, dưới dạng các *mô hình* tham khảo. Có tất cả 8 mô hình được định nghĩa như trình bày dưới đây.

7.2.1 Mô hình hệ thống

Chuẩn IEC mô tả hệ thống đo lường - điều khiển quá trình công nghiệp ở dạng một tập hợp các thiết bị kết nối lại và liên lạc với nhau thông qua một hay nhiều mạng truyền thông như trong Hình 7-6 dưới đây. Các mạng này có thể được tổ chức theo thứ bậc.

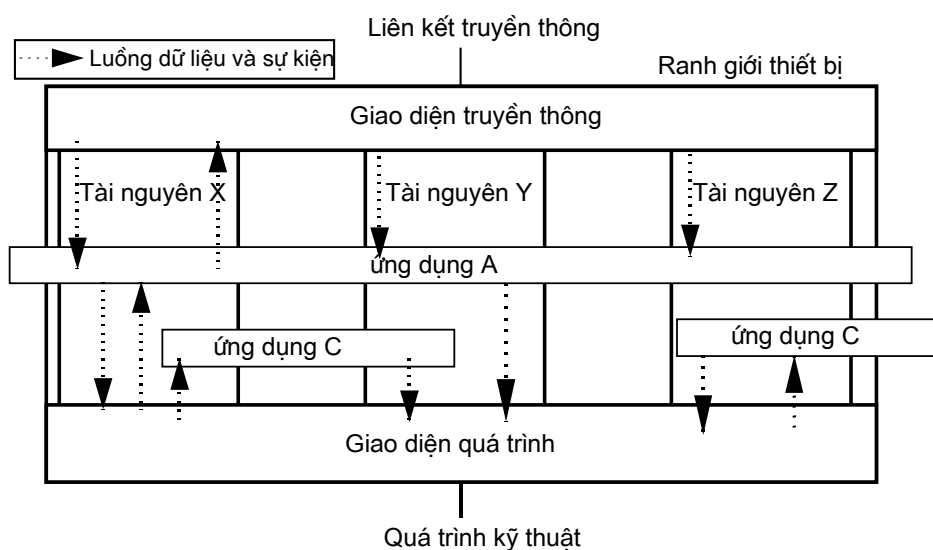


Hình 7-6: Mô hình hệ thống theo IEC 61499

Mỗi chức năng được thực hiện bởi hệ thống đo lường - điều khiển được mô hình hóa bằng một ứng dụng như trong hình vẽ trên. Một ứng dụng có thể nằm trong một thiết bị duy nhất, như ứng dụng C trong hình vẽ, hoặc phân tán trên một số thiết bị, như các ứng dụng A và B. Ví dụ, một ứng dụng có thể chứa một hay nhiều vòng điều khiển trong đó việc lấy mẫu đầu vào được thực hiện bởi một thiết bị, tính toán điều khiển được thực hiện bởi thiết bị khác còn việc chuyển đổi đầu ra lại được tiến hành bởi thiết bị thứ ba.

7.2.2 Mô hình thiết bị

Như biểu diễn trong Hình 7-7, mỗi thiết bị chứa trong nó ít nhất một giao diện, ngoài ra có thể có hoặc không chứa các tài nguyên và mạng khối chức năng.



Hình 7-7: Mô hình thiết bị theo IEC 61499 (ví dụ thiết bị 2 trong Hình 7-6).

Giao diện quá trình là một thành phần của thiết bị có nhiệm vụ ánh xạ giữa cấp điều khiển bên dưới (thiết bị đo tương tự, thiết bị vào/ra phân tán, ...) và các tài nguyên bên trong thiết bị. Thông tin trao đổi giữa cấp điều khiển bên dưới và các tài nguyên thể hiện dưới dạng luồng dữ liệu và sự kiện.

Giao diện truyền thông cũng là một thành phần của thiết bị có nhiệm vụ ánh xạ giữa mạng truyền thông phía trên và các tài nguyên bên trong thiết bị. Các dịch vụ cung cấp bởi giao diện này bao gồm :

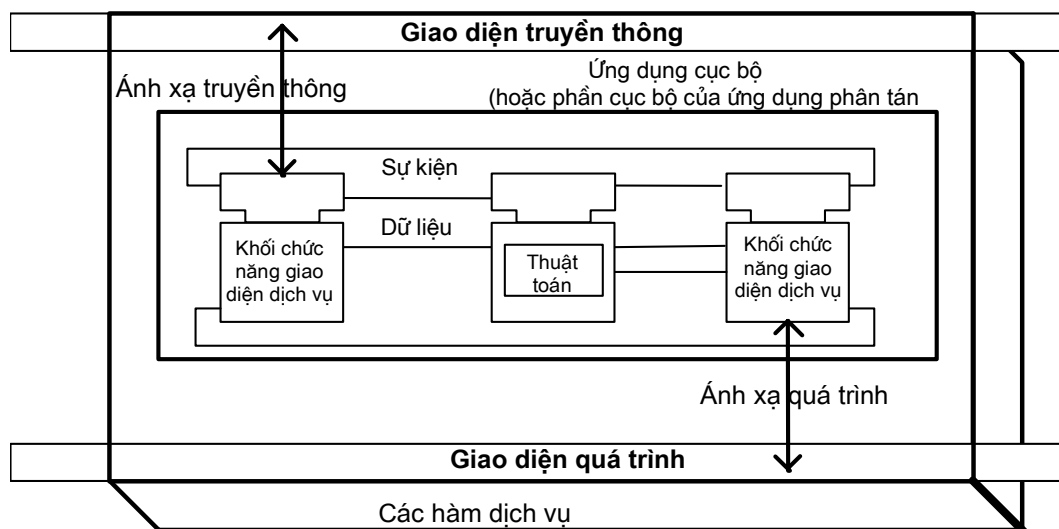
- Biểu diễn các thông tin truyền tới tài nguyên thành dạng dữ liệu và sự kiện
- Cung cấp các dịch vụ khác để hỗ trợ việc lập trình, đặt cấu hình, dò lỗi, ...

7.2.3 Mô hình tài nguyên

Chuẩn IEC coi tài nguyên là một đơn vị chức năng, chứa trong một thiết bị vốn có quyền độc lập điều khiển hành vi của mình. Tài nguyên có thể được

tạo ra, đặt cấu hình, thể hiện bằng tham số, khởi động, xóa, ... mà không ảnh hưởng đến các tài nguyên khác bên trong thiết bị.

Chức năng của tài nguyên là nhận dữ liệu và/hoặc sự kiện từ giao diện truyền thông hoặc giao diện quá trình, xử lý dữ liệu/sự kiện đó rồi gửi tr lại dữ liệu/sự kiện tới giao diện truyền thông/quá trình theo yêu cầu của ứng dụng sử dụng tài nguyên.



Hình 7-8: Mô hình tài nguyên

Như trong , tài nguyên được mô hình hóa bao gồm các thành phần sau :

- Một hoặc nhiều *ứng dụng cục bộ* (hay thành phần cục bộ của một ứng dụng phân tán). Các biến và sự kiện được quản lý trong phần này các biến đầu vào, biến đầu ra, sự kiện đầu vào, sự kiện đầu ra của các khối chức năng thực hiện phép toán cần cho ứng dụng.
- Thành phần *ánh xạ quá trình* có nhiệm vụ ánh xạ dữ liệu và sự kiện giữa ứng dụng và giao diện quá trình. Như biểu diễn trong hình, điều này được thực hiện bởi các khối chức năng phục vụ giao tiếp. Khối chức năng phục vụ giao tiếp cũng là khối chức năng thông thường nhưng được chuyên biệt hóa cho nhiệm vụ này.
- Thành phần *ánh xạ truyền thông* có nhiệm vụ ánh xạ dữ liệu và sự kiện giữa ứng dụng và giao diện truyền thông. Như biểu diễn trong hình, điều này cũng được thực hiện bởi các khối chức năng phục vụ giao tiếp.
- *Bộ phận lập lịch* có nhiệm vụ điều khiển sự thực thi của các khối hàm bên trong ứng dụng cũng như dòng dữ liệu truyền giữa chúng tuân theo những yêu cầu về thời gian và trình tự quyết định bởi sự xuất hiện của các sự kiện, liên kết giữa các khối hàm và thông tin khác như chu kỳ và mức ưu tiên.

7.2.4 Mô hình ứng dụng

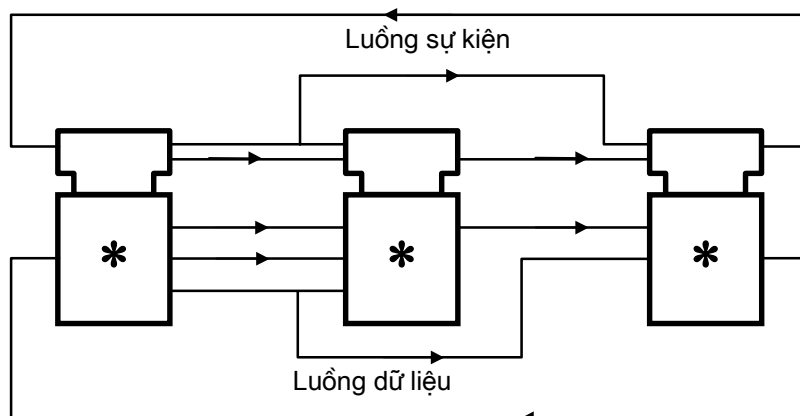
Mỗi ứng dụng được mô hình hóa ở dạng một mạng các khối chức năng (function block network). Mạng này được cấu thành từ các nút mạng là các

khối chức năng và liên kết giữa các nút mạng là liên kết dữ liệu (*data connections*) và liên kết sự kiện (*event connections*).

Một ứng dụng có thể được phân tán trên nhiều tài nguyên thuộc một hay nhiều thiết bị. Mỗi tài nguyên sử dụng các mối liên hệ nhân quả chỉ ra bởi ứng dụng để quyết định phản ứng thích hợp đối với các sự kiện phát sinh từ giao diện truyền thông, giao diện quá trình, từ trong bản thân chính tài nguyên đó hoặc từ tài nguyên khác. Các phản ứng này bao gồm :

- Sự lập lịch và thực thi thuật toán (bên trong tài nguyên)
- Việc thay đổi các biến
- Sản sinh các tín hiệu
- Tương tác với giao diện truyền thông và giao diện quá trình

Ứng dụng được định nghĩa bằng cách chỉ ra mối liên hệ giữa các khối chức năng chứa bên trong nó như minh họa trong hình vẽ, cụ thể là chỉ ra luồng sự kiện (*event flow*) và luồng dữ liệu (*data flow*). Chính luồng sự kiện là cái quyết định việc lập lịch và thực thi của mỗi tài nguyên bên trong ứng dụng.



Hình 7-9: Mô hình ứng dụng

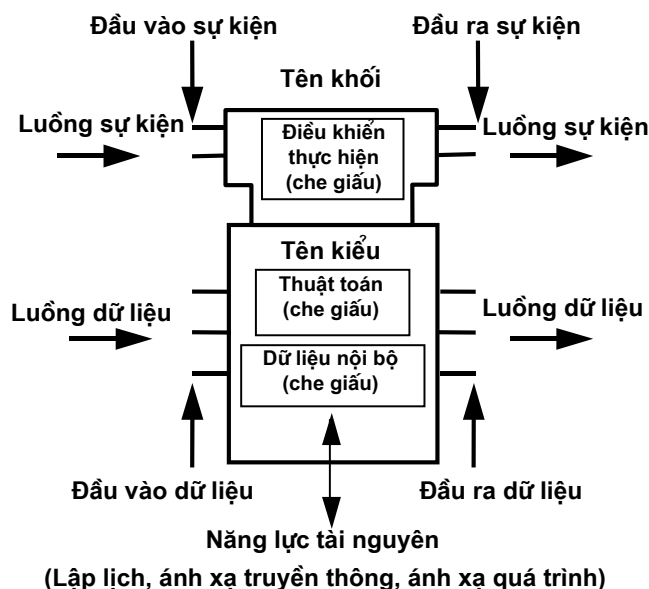
7.2.5 Mô hình khối chức năng

Mỗi khối chức năng thực ra là một biến thể hiện (*function block instance*) của một kiểu khối chức năng nào đó (*function block type*) cũng tương tự như mỗi đối tượng là một biến thể hiện của một lớp nào đó.

Các đặc tính của khối chức năng:

- Tên biến (là tên của chính khối chức năng đó) và tên kiểu (là tên của kiểu khối chức năng mà nó thuộc)
- Một tập các đầu vào sự kiện, dùng để nhận các sự kiện đến từ các liên kết sự kiện với bên ngoài. Chính các sự kiện đầu vào sẽ ảnh hưởng tới việc thực thi thuật toán bên trong khối chức năng.
- Một tập các đầu ra sự kiện, dùng để đẩy các sự kiện ra các liên kết sự kiện với bên ngoài tùy thuộc và sự thực thi của thuật toán và một số yếu tố khác
- Một tập các đầu vào dữ liệu, chúng được ánh xạ tới các biến đầu vào tương ứng

- Một tập các đầu ra dữ liệu, chúng được ánh xạ tới các biến đầu ra tương ứng
- Dữ liệu nội bộ, chúng được ánh xạ tới tập các biến nội bộ
- Các đặc trưng về chức năng, được xác định bằng cách kết hợp giữa dữ liệu nội bộ với thuật toán của khối chức



Hình 7-10: Mô hình khối chức năng

Thuật toán bên trong khối hàm, về nguyên tắc, không nhìn thấy được từ bên ngoài, trừ phi được người cung cấp khối hàm mô tả theo một cách nào đó. Thêm nữa, khối hàm có thể chứa bên trong nó các biến nội bộ hay các thông tin về trạng thái tồn tại không đổi giữa những lần gọi thuật toán của khối hàm nhưng chúng cũng không truy cập được từ bên ngoài.

Mỗi kiểu khối chức năng là một yếu tố phần mềm chỉ rõ đặc tính của tất cả các biến thể hiện thuộc kiểu đó, bao gồm :

- Tên kiểu
- Số lượng, tên, tên kiểu và trật tự của các đầu vào sự kiện và đầu ra sự kiện
- Số lượng, tên, kiểu dữ liệu và trật tự của các biến đầu vào, biến đầu ra và biến nội bộ

Đặc tả kiểu khối chức năng ngoài phần chung kể trên còn có thêm phần định nghĩa chức năng đặc trưng của khối phụ thuộc vào loại khối cụ thể. Có cả thảy 3 loại khối chức năng :

- Khối chức năng cơ bản là đơn vị chức năng nhỏ nhất, không thể phân chia hơn được nữa
- Khối chức năng kết hợp (Siêu khối) là loại khối chức năng hợp thành từ nhiều khối chức năng khác nhỏ hơn
- Khối chức năng dịch vụ giao tiếp là loại khối chức năng cung cấp các dịch vụ giao diện giữa các khối chức năng khác với nhau hoặc giữa tài

nguyên và mạng truyền thông và giữa tài nguyên và quá trình công nghệ được điều khiển.

Đối với kiểu khối chức năng cơ bản, chức năng đặc trưng của nó được diễn đạt bằng cách mô tả thuật toán bên trong khối đó hoạt động dựa trên giá trị của các biến đầu vào, biến đầu ra, biến nội bộ để tạo ra giá trị mới cho các biến đầu ra và biến nội bộ như thế nào, cũng như mối liên hệ giữa sự khởi động, thực thi và kết thúc của thuật toán với sự xuất hiện của các sự kiện tại đầu vào và sự kiện đầu ra của khối.

Đối với kiểu khối chức năng kết hợp, đặc tả kiểu còn bao gồm các liên kết dữ liệu và liên kết sự kiện giữa những khối chức năng thành phần bên trong khối chức năng kết hợp.

Đối với kiểu khối chức năng phục vụ giao tiếp, chức năng đặc trưng được mô tả bằng cách ánh xạ các dịch vụ cơ bản mà nó cung cấp thành các đầu vào sự kiện, đầu ra sự kiện, đầu vào dữ liệu, đầu ra dữ liệu của nó.

7.2.6 Mô hình phân tán

Một ứng dụng cụ thể được phân tán bằng cách phân bố các khối chức năng của nó vào các tài nguyên khác nhau nằm trên một hay nhiều thiết bị. Bởi vì chi tiết bên trong khối chức năng là không nhìn thấy được đối với ứng dụng sử dụng nó cho nên khối chức năng phi là đơn vị nhỏ nhất không phân chia được đối với sự phân tán này. Nghĩa là, không thể chia nhỏ khối chức năng hơn nữa để phân tán trên nhiều tài nguyên; các thành phần chứa bên trong khối chức năng phải nằm trên cùng một tài nguyên.

Các mối liên hệ về chức năng giữa các khối chức năng không được phép bị ảnh hưởng bởi sự phân tán. Tuy nhiên, so với ứng dụng tập trung, các yếu tố về thời gian và độ tin cậy của chức năng truyền thông có ảnh hưởng đáng kể tới các yếu tố về thời gian và độ tin cậy của toàn ứng dụng phân tán.

7.2.7 Mô hình quản lý

Việc quản lý các tài nguyên bên trong một thiết bị được mô hình theo một trong hai cách

- Trong thiết bị, có một tài nguyên được tạo riêng để quản lý tất cả những cái còn lại. Nó cung cấp một tập các dịch vụ quản lý cho các tài nguyên khác dùng chung.
- Mỗi tài nguyên có một bộ phận quản lý của riêng mình. Nói cách khác, các dịch vụ quản lý được phân tán ra khắp tất cả các tài nguyên bên trong thiết bị.

7.2.8 Mô hình trạng thái hoạt động

Bất kỳ hệ thống nào cũng được thiết kế, bàn giao công việc, đưa vào vận hành, duy trì bảo dưỡng và cuối cùng sau khi hoàn thành nhiệm vụ - được hủy bỏ. Điều này được mô hình hóa bằng cái gọi là *vòng đời* của hệ thống.

Đến lượt mình, các đơn vị chức năng cấu thành nên hệ thống như thiết bị, tài nguyên, ứng dụng mỗi cái cũng có vòng đời riêng của mình.

Để hỗ trợ cho đơn vị chức năng tại mỗi thời điểm trong vòng đời của nó, các hành động khác nhau phải được thực hiện. Để phân biệt xem hành động nào có thể thực hiện và để duy trì tính toàn vẹn của đơn vị chức năng, các trạng thái hoạt động phải được định nghĩa, ví dụ “Đang hoạt động”, “Có thể cấu hình”, “Đã nạp”, “Đã dừng”, ...

Mỗi trạng thái hoạt động của một đơn vị chức năng chỉ rõ hành động nào được chấp nhận và hành vi mong muốn tương ứng.

Hệ thống có thể được tổ chức theo cách một khối chức năng nào đó có thể sở hữu hoặc nhận được quyền thay đổi trạng thái của khối chức năng khác.

Một đặc điểm nổi bật của IEC 61499 là nó yêu cầu các công cụ phần mềm tuân theo chuẩn phải có khả năng trao đổi thông tin với nhau. Cụ thể là IEC 61499-2 yêu cầu nhà sản xuất thiết bị phần cứng hay thư viện phần mềm phải cung cấp đủ thông tin cần thiết cho việc sử dụng sản phẩm của họ một cách hiệu quả trong khi vẫn đảm bảo quyền sở hữu trí tuệ. Điều này được thực hiện bằng cách sử dụng ngôn ngữ XML. IEC 61499-2 định nghĩa một tập các thẻ (tag) và cấu trúc tài liệu XML chuẩn, dựa vào đó nhà sản xuất mô tả sản phẩm của mình. Tuân theo một tập thẻ chung, các sản phẩm có thể làm việc trực tiếp với nhau (interoperability).

8 MỘT SỐ CHUẨN GIAO TIẾP CÔNG NGHIỆP

8.1 MMS

MMS (*Manufacturing Message Specification*) là một chuẩn quốc tế cho việc xây dựng lớp ứng dụng theo mô hình qui chiếu OSI. Chuẩn này có ý nghĩa đặc biệt trong các hệ thống truyền thông công nghiệp. Về cơ bản, MMS qui định một tập hợp các dịch vụ chuẩn cho việc trao đổi dữ liệu thời gian thực cũng như thông tin điều khiển giám sát. Các dịch vụ này cũng như các giao thức tương ứng được chuẩn hóa trong ISO/IEC 9506.

MMS có xuất xứ từ MAP (*Manufacturing Automation Protocol*), một giao thức do hãng General Motors khởi xướng phát triển vào đầu những năm 80. Mặc dù MAP không trở thành giao thức truyền thông thống nhất cho công nghiệp tự động hóa, nó đã tác động có tính chất định hướng tới các phát triển sau này. Trên cơ sở của MAP, các dịch vụ truyền thông đã được định nghĩa trong MMS tạo cơ sở quan trọng trong việc xây dựng lớp ứng dụng trong nhiều hệ thống bus trường. Một số ví dụ như FMS (*Fieldbus Message Specification*) của PROFIBUS, PMS (*Peripheral Message Specification*) của Interbus, MPS (*Manufacturing Periodic/aperiodic Services*) của WorldFIP và RAC (*Remote Access Control*) của Bitbus đều là các tập con các dịch vụ MMS.

Các dịch vụ được định nghĩa trong MMS có tính chất thông dụng và đa dạng, có thể thích hợp với rất nhiều loại thiết bị, nhiều ứng dụng và ngành công nghiệp khác nhau. Ví dụ, dịch vụ *Read* cho phép một chương trình ứng dụng hoặc một thiết bị đọc một hoặc nhiều biến một lúc từ một chương trình ứng dụng hoặc một thiết bị khác. Bất kể một thiết bị là PLC hay robot, một chương trình điều khiển tự động hay chương trình điều khiển giám sát, các dịch vụ và thông báo MMS đều như nhau.

Về cơ bản, MMS định nghĩa:

- Hơn 80 dịch vụ truyền thông thông dụng cho các hệ thống bus, trong đó có kiểm soát đường nối, truy nhập biến, điều khiển sự kiện, cài đặt và can thiệp chương trình
- Một giao thức qui định cấu trúc dữ liệu cho việc chuyển giao tham số của các dịch vụ
- Mô hình một số đối tượng “ảo”, đại diện cho các đối tượng vật lý (máy móc, robot,...)
- Một cơ chế Client/Server trong quan hệ giữa các đối tác truyền thông.

Chuẩn ISO/IEC 9506 bao gồm hai phần cốt lõi sau:

- Phần 1: Đặc tả dịch vụ, định nghĩa mô hình thiết bị sản xuất ảo VMD (*Virtual Manufacturing Device*), các dịch vụ trao đổi giữa các nút mạng, các thuộc tính cũng như tham số tương ứng với VMD và các dịch vụ.
- Phần 2: Đặc tả giao thức, định nghĩa trình tự các thông báo được gửi đi

trong mạng, cấu trúc và kiểu mã hóa các thông báo, tương tác giữa MMS với các lớp khác trong mô hình OSI. MMS sử dụng chuẩn lớp biểu diễn dữ liệu ASN.1 (Abstract Notation Number One - ISO 8824) để đặc tả cấu trúc các thông báo.

Ngoài ra, bốn phần phụ tiếp theo - được gọi là các chuẩn đi kèm (*Companion Standard*) - mở rộng phần cốt lõi nhằm thích ứng cho các lĩnh vực ứng dụng điều khiển khác nhau như *Robot Control* (phần 3), *Numeric Control* (phần 4), *Programmable Controller* (phần 5) và *Process Control* (phần 6).

Một trong những điểm đặc trưng của MMS là mô hình đối tượng VMD. Trên quan điểm hướng đối tượng, mô hình VMD cho phép các thiết bị đóng vai trò một server, cung cấp các dịch vụ cho các client thông qua các đối tượng ảo. Các đối tượng ảo này đại diện cho những đối tượng khác nhau trong một hệ thống kỹ thuật, trong đó có cả các biến, chương trình, sự kiện, v.v... Mỗi chương trình ứng dụng có thể đồng thời đóng vai trò server và client. Mô hình VMD định nghĩa các đối tượng sau:

- *VMD*: Bản thân VMD được coi là một đối tượng hợp thành từ các đối tượng khác, đại diện cho toàn bộ một thiết bị.
- *Domain*: Đại diện một phần nhớ có liên kết trong một VMD, ví dụ phần nhớ cho một chương trình có thể nạp xuống (*download*) và nạp lên (*upload*) được.
- *Program Invocation*: Một chương trình chạy trong bộ nhớ chính được hợp thành bởi một hoặc nhiều *domain*.
- *Variable*: Một biến dữ liệu có kiểu (ví dụ số nguyên, số thực dấu phẩy động, mảng).
- *Kiểu*: Mô tả cấu trúc và ý nghĩa của dữ liệu chứa trong một biến.
- *Named Variable List*: Một danh sách nhiều biến có tên.
- *Semaphore*: Một đối tượng dùng để kiểm soát việc truy nhập cạnh tranh một tài nguyên chung (ví dụ bộ nhớ, CPU, cổng vào/ra)
- *Operator Station*: Một trạm có màn hình và bàn phím dùng cho thao tác viên vận hành quá trình.
- *Event Condition*: Một đối tượng đại diện cho trạng thái của một sự kiện
- *Event Action*: Một đối tượng đại diện hành động được thực hiện khi trạng thái của một sự kiện thay đổi
- *Event Enrollment*: Một đối tượng đại diện cho một chương trình ứng dụng mạng được thông báo khi trạng thái của một sự kiện thay đổi.
- *Journal*: Một đối tượng ghi lại diễn biến của các sự kiện và biến theo thời gian.
- *File*: Một file trong một trạm server.
- *Transaction*: Đại diện một yêu cầu dịch vụ MMS riêng biệt.

Các loại dịch vụ tương ứng với các đối tượng cơ bản được giới thiệu tóm tắt trong bảng 8.1.

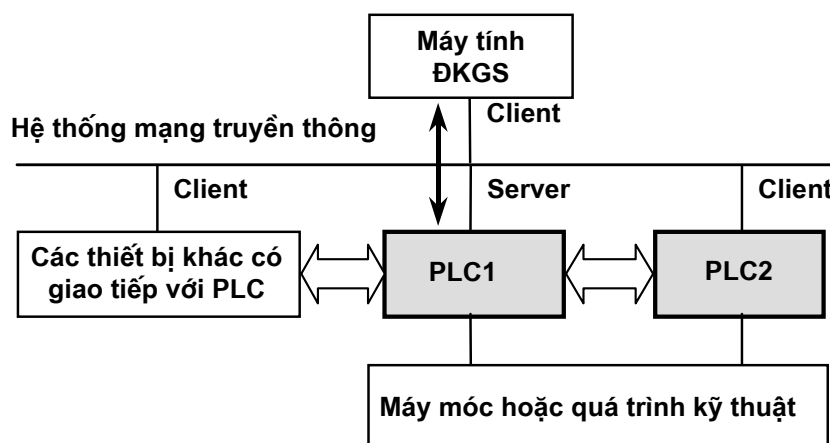
Bảng 8.1: Tổng quan các dịch vụ MMS cơ bản

Nhóm dịch vụ	Mô tả tóm tắt các dịch vụ	Ví dụ
Environment and General Management Services	Khởi tạo/Kết thúc các quan hệ truyền thông	- Initiate - Abort
VMD Support Services	Thông tin về trạng thái của một thiết bị ảo, về các đặc tính, thông số và đối tượng của nó	- Status - UnsolicitedStatus - Identity
Domain Management Services	Tạo lập, nạp lên/nạp xuống và xóa Domain	- InitiateDownloadSequence - DownloadSegment - InitiateUploadSequence - UploadSegment - RequestDomainDownload - RequestDomainUpload
Program Invocation Management Services	Tạo lập, xóa và kiểm soát Program Invocation	- CreateProgramInvocation - Start - Stop - Resume - Reset
Variable Access Services	Truy nhập các biến của một VMD	- Read - Write
Semaphore Management Services	Đồng bộ hóa việc truy nhập cạnh tranh tài nguyên của VMD	- DefineSemaphore - DeleteSemaphore - ReportSemaphoreStatus
Operator Communication Services	Hỗ trợ việc giao tiếp với một trạm thao tác	- Input - Output
Event Management Services	Xử lý sự kiện, sự cố	- AlterEventCondition Monitoring - EventNotification - AcknowledgeEvent Notification
Journal Management Services	Ghi biên bản các sự kiện và thông tin	- InitiateJournal - ReportJournalStatus

8.2 IEC-61131-5

8.2.1 Mô hình giao tiếp mạng

Đối tượng của chuẩn IEC 61131-5 là các dịch vụ do các thiết bị điều khiển khả trình (PLC) thực hiện, hoặc các dịch vụ các PLC có thể yêu cầu từ các thiết bị khác, thể hiện qua các hàm và khối chức năng sử dụng khi lập trình với IEC 61131-5. Phạm vi của chuẩn này vì vậy bó hẹp ở việc giao tiếp giữa các PLC hoặc giữa PLC và một thiết bị khác, theo kiến trúc Client/Server.



Hình 8.1: Mô hình giao tiếp mạng theo IEC 61131-5

8.2.2 Dịch vụ giao tiếp

Các dịch vụ giao tiếp cung cấp thông tin trạng thái và chỉ thị sự cố của các thành phần:

- Thiết bị điều khiển khả trình (tổng thể)
- Vào/ra
- Bộ xử lý trung tâm
- Cung cấp nguồn
- Bộ nhớ
- Hệ thống truyền thông

Lưu ý rằng, status cung cấp thông tin về trạng thái của thiết bị điều khiển và các thành phần phần cứng, phần rắn của nó, không quan tâm tới thông tin cấu hình. Dữ liệu trạng thái cũng không cung cấp thông tin về quá trình được điều khiển cũng như chương trình ứng dụng trên PLC.

Các dịch vụ giao tiếp được liệt kê trong bảng dưới đây. Tuy nhiên, một PLC không bắt buộc phải cung cấp tất cả các dịch vụ này.

Bảng 8.2: Các dịch vụ giao tiếp

STT	Các dịch vụ giao tiếp	PLC yêu cầu	PLC đáp ứng	Khởi hàm có sẵn
1	Kiểm tra thiết bị	X	X	X
2	Thu thập dữ liệu	X	X	X
3	Điều khiển	X	X	X
4	Đồng bộ hóa giữa các chương trình ứng dụng	X	X	X
5	Báo động	X	0	X
6	Thực hiện chương trình và điều khiển vào/ra	0	X	0
7	Truyền nạp chương trình ứng dụng	0	X	0
8	Quản lý nối	X	X	X

8.2.3 Các khối chức năng giao tiếp

Các dịch vụ giao tiếp được thực hiện qua các khối chức năng giao tiếp (*communication function block, CFB*), như được liệt kê trong bảng dưới đây.

Bảng 8.3: Các khối chức năng giao tiếp (CFB)

STT	Chức năng	Tên khối chức năng
1	Định địa chỉ các biến từ xa	REMOTE_VAR
2 3	Kiểm tra thiết bị	STATUS, USTATUS
4	Thu thập dữ liệu kiểu hỏi tuần tự	READ,
5 6	Thu thập dữ liệu kiểu lập trình	USEND, URCV
7	Điều khiển tham số	WRITE,
8 9	Điều khiển liên động	SEND, RCV
10 11	Báo động được lập trình	NOTIFY, ALARM
12	Quản lý nối	CONNECT

Lưu ý: Các khối hàm UXXX thể hiện các hàm dịch vụ không cần yêu cầu (*unsolicited services*).

Kiểm tra thiết bị

Các khối hàm STATUS và USTATUS hỗ trợ việc PLC kiểm tra trạng thái của các thiết bị tự động hóa khác.

Thu thập dữ liệu

Dữ liệu trong các thiết bị khác có thể được biểu diễn qua các biến. Có hai phương pháp để PLC truy nhập các dữ liệu này sử dụng các CFB:

- Hỏi tuần tự (*polled*): PLC sử dụng khối hàm READ để đọc giá trị của một hoặc nhiều biến tại thời điểm được chương trình ứng dụng trong PLC xác định. Việc truy nhập các biến này có thể do các thiết bị kiểm soát.
- Lập trình: Thời điểm dữ liệu cung cấp cho PLC được quyết định bởi các thiết bị khác. Các khối URCV/USEND được sử dụng trong các chương trình ứng dụng PLC để nhận dữ liệu từ và gửi dữ liệu đến các thiết bị khác.

Điều khiển

Hai phương pháp điều khiển cần được PLC hỗ trợ: điều khiển tham số (*parametric*) và điều khiển khóa liên động (*interlocked*)

- Điều khiển tham số: Hoạt động của các thiết bị được điều khiển bằng cách thay đổi các tham số của chúng. PLC sử dụng khối WRITE để thực hiện hoạt động này từ một chương trình ứng dụng.
- Điều khiển khóa liên động: Một client yêu cầu server thực hiện một phép toán ứng dụng và thông báo kết quả cho client. PLC sử dụng các khối SEND và RCV để thực hiện vai trò client and server.

Báo động

Một PLC có thể gửi các báo động tới các client khi một sự kiện xảy ra. Client có thể thông báo lại đã xác nhận tới bộ điều khiển. PLC sử dụng các khối ALARM và NOTIFY trong các chương trình ứng dụng để gửi các thông báo cần xác nhận và không cần xác nhận.

Quản lý các mối liên kết

Các chương trình ứng dụng trong PLC sử dụng khối CONNECT để quản lý các mối liên kết.

8.3 OPC

Tiến bộ của các hệ thống bus trường cùng với sự phổ biến của các thiết bị cận trường thông minh là hai yếu tố quyết định tới sự chuyển hướng sang cấu trúc phân tán trong các giải pháp tự động hóa. Sự phân tán hóa này một mặt mang lại nhiều ưu thế so với cấu trúc xử lý thông tin tập trung cổ điển, như độ tin cậy và tính linh hoạt của hệ thống, nhưng mặt khác cũng tạo ra hàng loạt thách thức mới cho giới sản xuất cũng như cho người sử dụng. Một trong những vấn đề thường gặp phải là việc tích hợp hệ thống. Tích hợp theo chiều ngang đòi hỏi khả năng tương tác giữa các thiết bị tự động hóa của nhiều nhà sản xuất khác nhau. Bên cạnh đó, tích hợp theo chiều dọc đòi hỏi khả năng kết nối giữa các ứng dụng cơ sở như đo lường, điều khiển với các ứng dụng cao cấp hơn như điều khiển giám sát và thu thập dữ liệu (*supervisory control and data acquisition*, SCADA), giao diện người-máy (*human-machine interface*, HMI) và hệ thống điều hành sản xuất (*manufacturing execution system*, MES). Việc sử dụng một chuẩn giao diện vì vậy trở thành một điều kiện tiên quyết. Tiêu biểu cho hướng đi này là chuẩn OPC, được chấp nhận rộng rãi trong các ứng dụng tự động hóa quá trình công nghiệp.

8.3.1 Tổng quan về kiến trúc OPC

OPC (*OLE for Process Control*) là một chuẩn giao diện được hiệp hội OPC Foundation xây dựng và phát triển. Dựa trên mô hình đối tượng thành phần (D)COM của hãng Microsoft, OPC định nghĩa thêm một số giao diện cho khai thác dữ liệu từ các quá trình kỹ thuật, tạo cơ sở cho việc xây dựng các ứng dụng điều khiển phân tán mà không bị phụ thuộc vào mạng công nghiệp cụ thể. Trong thời điểm hiện nay, OPC cũng như COM tuy mới được thực hiện trên nền Windows, song đã có nhiều cố gắng để phổ biến sang các hệ điều hành thông dụng khác.

Với mục đích ban đầu là thay thế cho các dạng phần mềm kết nối như I/O-Drivers và DDE, OPC qui định một số giao diện chuẩn cho các chức năng như:

- Khai thác, truy nhập dữ liệu quá trình (*Data Access*) từ nhiều nguồn khác nhau (PLC, các thiết bị trường, bus trường, cơ sở dữ liệu,...)
- Xử lý sự kiện và sự cố (*Event and Alarm*)

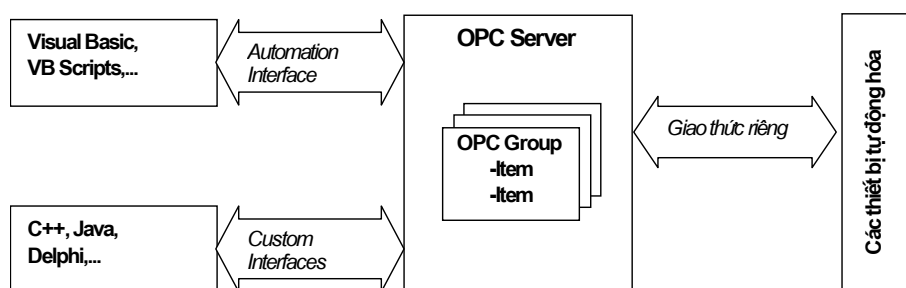
- Truy nhập dữ liệu quá khứ (*Historical Access*)

Trong tương lai OPC sẽ hỗ trợ các chức năng khác như an toàn hệ thống (*Security*) và điều khiển mẻ (*Batch*). OPC sử dụng cơ chế COM/COM để cung cấp các dịch vụ truyền thông cho tất cả các ứng dụng hỗ trợ COM. Có thể kể ra hàng loạt các ưu điểm của việc sử dụng OPC như:

- Cho phép các ứng dụng khai thác, truy nhập dữ liệu theo một cách đơn giản, thống nhất
- Hỗ trợ truy nhập dữ liệu theo cơ chế hỏi tuần tự (*polling*) hoặc theo sự kiện (*event-driven*)
- Được tối ưu cho việc sử dụng trong mạng công nghiệp
- Kiến trúc không phụ thuộc vào nhà cung cấp thiết bị
- Linh hoạt và hiệu suất cao
- Sử dụng được từ hầu hết các công cụ phần mềm SCADA thông dụng, hoặc bằng một ngôn ngữ bậc cao (C++, Visual Basic, Delphi,...).

Cốt lõi của OPC là một chương trình phần mềm phục vụ gọi là *OPC-Server*, trong đó chứa các mục dữ liệu (*OPC-Item*) được tổ chức thành các nhóm (*OPC-Group*). Thông thường, một OPC-Server đại diện một thiết bị thu thập dữ liệu như PLC, RTU, I/O hoặc một cấu hình mạng truyền thông. Các OPC-Items sẽ đại diện cho các biến quá trình, các tham số điều khiển, v.v...

OPC được xây dựng dựa trên ý tưởng ứng dụng công nghệ COM nhằm đơn giản hóa, chuẩn hóa việc khai thác dữ liệu từ các thiết bị cận trường và thiết bị điều khiển, tương tự như việc khai thác một hệ thống cơ sở dữ liệu thông thường. Giống như COM, OPC không qui định việc thực hiện khai thác cụ thể, mà chỉ định nghĩa một số giao diện chuẩn. Thay cho việc dùng C/C++ dùng để định nghĩa một giao diện lập trình như thông thường, ngôn ngữ dùng ở đây (gọi là *interface definition language* hay IDL) không phụ thuộc vào nền cài đặt hay ngôn ngữ lập trình.



Hình 8.2: Kiến trúc sơ lược của OPC

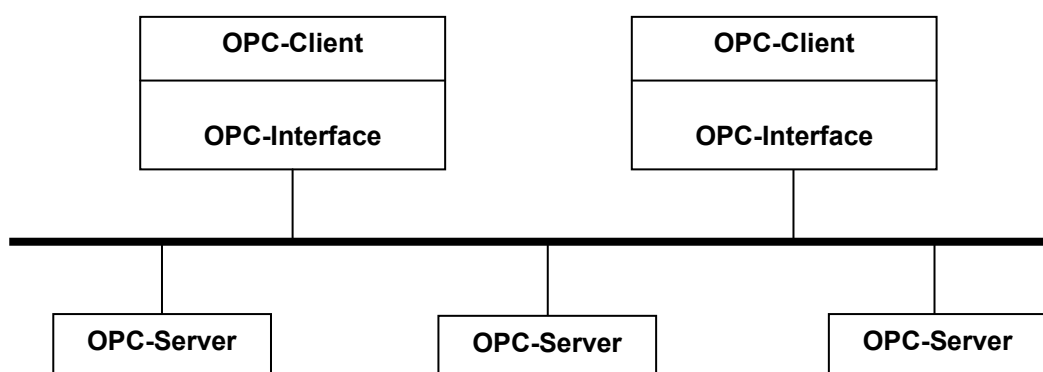
Như được minh họa trên Hình 8.2, hai kiểu đối tượng thành phần quan trọng nhất trong kiến trúc OPC là OPC-Server và OPC-Group. Trong khi OPC-Server có nhiệm vụ quản lý toàn bộ việc sử dụng và khai thác các dữ liệu, thì các đối tượng OPC-Group có chức năng tổ chức các phần tử dữ liệu (*items*) thành từng nhóm để tiện cho việc truy nhập. Thông thường, mỗi item ứng với một biến trong một quá trình kỹ thuật hay trong một thiết bị điều khiển.

OPC Server

OPC Server là một đối tượng phân tán, cung cấp giao diện OPC chuẩn cho các ứng dụng. Việc giao tiếp qua các mạng công nghiệp được thực hiện bằng các lời gọi đơn giản, thống nhất không phụ thuộc vào mạng truyền thông và giao thức được sử dụng.

OPC Server hỗ trợ hai phương pháp truy cập dữ liệu :

- *Polling*: Client chủ động yêu cầu Server cung cấp dữ liệu mỗi khi cần
- *Publisher/Subscriber*: Client chỉ cần một lần yêu cầu Server, sau đó tùy theo cách đặt (Theo chu kỳ, theo sự thay đổi của dữ liệu hoặc theo một sự kiện nào đó). Phương pháp này còn được gọi là truy cập không đồng bộ.



Hình 8.3: Kiến trúc Client/Server trong OPC

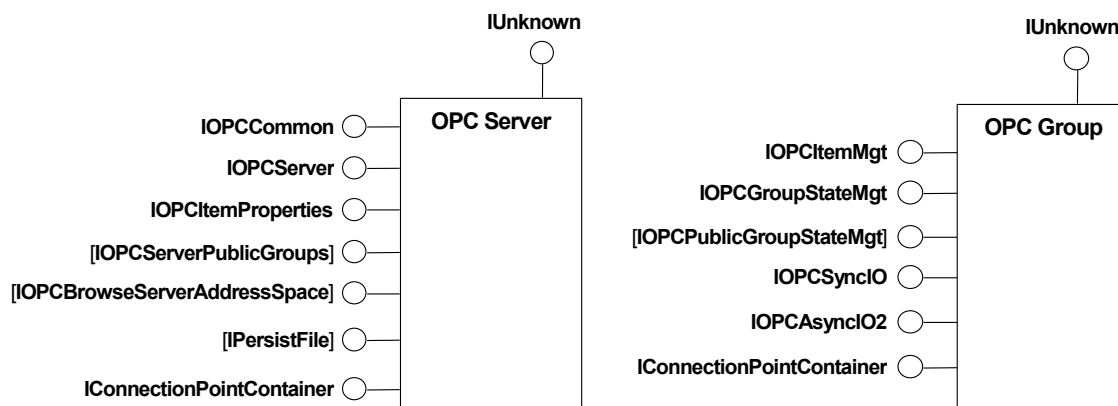
Chuẩn OPC hiện nay qui định hai kiểu giao diện là *Custom Interfaces* (OPC Taskforce, 1998b) và *Automation Interface* (OPC Taskforce, 1998c)². Kiểu thứ nhất bao gồm một số giao diện theo mô hình COM thuần túy, còn kiểu thứ hai dựa trên công nghệ mở rộng *OLE-Automation*. Sự khác nhau giữa hai kiểu giao diện này không những nằm ở mô hình đối tượng, ở các ngôn ngữ lập trình hỗ trợ mà cũng còn ở tính năng, hiệu suất sử dụng. *Custom Interface* dùng các ngôn ngữ như C/C++ phức tạp hơn nhưng hiệu suất cao, dựa trực tiếp trên các đối tượng COM. *Automation Interface* dùng các ngôn ngữ đơn giản, phương pháp lập trình đơn giản, hiệu quả thấp, dựa trên công nghệ *COM automation*.

8.3.2 OPC Custom Interfaces

Giống như các đối tượng COM khác, hai loại đối tượng thành phần quan trọng nhất của OPC là OPC-Server và OPC-Group cung cấp các dịch vụ qua các giao diện của chúng, được gọi là *OPC Custom Interfaces*, như được minh họa trên Hình 3. Tham khảo (OPC Taskforce, 1988b) để tìm hiểu ý nghĩa cụ thể của từng giao diện. Chính vì những giao diện này chỉ là các giao diện theo mô hình COM thuần túy, việc lập trình với chúng đòi hỏi một ngôn ngữ biên

² Lưu ý khái niệm “Automation” được dùng ở đây hoàn toàn không có liên quan tới kỹ thuật tự động hóa

dịch. Trong thực tế, C++ là ngôn ngữ chiếm ưu thế tuyệt đối phục vụ mục đích này. Bên cạnh đó, các công cụ khác nhau cũng cung cấp một số phần mềm khung (*frameworks*) thích hợp để hỗ trợ người lập trình.



Hình 8.4: OPC Custom Interfaces

Để truy nhập dữ liệu dùng OPC Custom Interfaces, ta cần thực hiện hàng loạt các bước sau:

- Tạo một (bản sao) đối tượng OPC-Server
- Tìm và lưu trữ con trỏ (địa chỉ) của các giao diện cần dùng, trong đó có IOPCServer
- Dùng các phương pháp thích hợp của giao diện IOPCServer để tạo một số đối tượng OPC-Group như cần thiết
- Tìm và lưu trữ con trỏ (địa chỉ) của các giao diện cần dùng của các đối tượng OPC-Group
- Sử dụng các giao diện thích hợp của OPC-Group để tổ chức và cấu hình cho các đối tượng này, kể cả việc xây dựng mối liên hệ với các phần tử dữ liệu thực
- Sử dụng IOPCSyncIO và IOPCAsyncIO2 của các đối tượng OPC-Group để đọc hoặc viết dữ liệu theo cơ chế đồng bộ hoặc không đồng bộ (tùy ý hoặc định kỳ)
- Giải phóng các giao diện không sử dụng nữa
- Xử lý các lỗi trong từng bước nêu trên.

8.3.3 OPC Automation Interface

Giống như đối với các đối tượng OLE-Automation khác, việc sử dụng các đối tượng của OPC Automation Interface được đơn giản hóa nhiều. Cụ thể, nhiều thủ tục phức tạp trong lập trình với COM được loại bỏ. Người lập trình không cần hiểu biết sâu sắc về COM cũng như C++, mà chỉ cần sử dụng thành thạo một công cụ tạo dựng ứng dụng RAD (*rapid application development*) như Visual Basic.

Mặt trái của vấn đề lại là, sự đơn giản hóa của phương pháp này phải trả giá bằng sự hạn chế trong phạm vi chức năng, hiệu suất sử dụng và tốc độ

trao đổi dữ liệu. Nhất là trong một giải pháp tự động hóa phân tán, có sự tham gia của các mạng truyền thông công nghiệp, thì hai điểm yếu nói sau trở nên rất đáng quan tâm. Lý do nằm chính trong mô hình giao tiếp của OLE-Automation và các công cụ hỗ trợ, đó là:

- Dùng kiểu dữ liệu đa năng (VARIANT) một mặt sẽ lãng phí khi trao đổi dữ liệu nhỏ, một mặt hạn chế kiểu dữ liệu sử dụng được
- Cơ chế tập trung hóa việc đón nhận và chuyển giao thông tin dùng giao diện IDispatch làm giảm thời gian phản ứng của một ứng dụng đối với một sự kiện một cách đáng kể.

OPC nhóm các giao diện và các *methods* của chúng theo thứ tự 3 lớp:

- OPC Server class
- OPC Group class
- OPC Item class

8.4 Ngôn ngữ đánh dấu khả mở XML

8.4.1 Giới thiệu chung

XML (*eXtensible Markup Language*) là một tập con của ngôn ngữ SGML (*Standard Generalized Markup Language*). SGML là một chuẩn quốc tế (ISO 8879) về một loại "siêu ngôn ngữ" có khả năng tạo ra các loại ngôn ngữ đánh dấu khác và XML chỉ đơn giản là một phiên bản đơn giản hoá của SGML, được xây dựng và quản lý bởi tổ chức W3C (*World Wide Web Consortium*). Tuy nhiên, đằng sau nó là cả một triết lý về cách lưu trữ và trao đổi thông tin.

Trong tin học, cách lưu trữ thông tin là một yếu tố quan trọng. Thông tin được lưu trữ dưới một trong hai dạng: dạng nhị phân (binary) và dạng văn bản (text). Dạng lưu trữ nhị phân thuận tiện hơn cho các chương trình xử lý, trong khi con người không thể đọc được dạng này còn dạng văn bản thuận lợi đối với con người nhưng lại không hiệu quả trong lưu trữ, xử lý. Mặt khác, khi lựa chọn khuôn dạng lưu trữ thông tin, người ta cũng phải cân nhắc xem nên sử dụng khuôn dạng chuẩn đã có sẵn hay tự xây dựng một khuôn dạng cho riêng mình. Các loại ngôn ngữ đánh dấu chính là cách lưu trữ thông tin dạng văn bản, ở đó ý nghĩa của văn bản được thể hiện bởi các *thẻ đánh dấu* (*markup tag*).

HTML là một ngôn ngữ đánh dấu điển hình, là nền tảng của công nghệ Web. Một trang web về cơ bản là một tập tin văn bản có bổ sung các thẻ HTML, những thẻ này mô tả cách trình duyệt sẽ hiển thị văn bản như thế nào. Ví dụ, khi đoạn văn bản nào đó cần in nghiêng trong trình duyệt thì nó sẽ được đặt giữa cặp thẻ <I> ... </I>, đoạn văn bản cần in đậm được đặt giữa cặp thẻ ý nghĩa của các thẻ HTML đã được quy định sẵn, trình duyệt dựa vào đó để hiển thị trang Web.

Tài liệu XML cũng có cấu trúc tương tự, tuy nhiên sở dĩ XML được gọi là ngôn ngữ khả mở (*extensible*) là bởi người viết tài liệu có thể tự tạo ra các thẻ

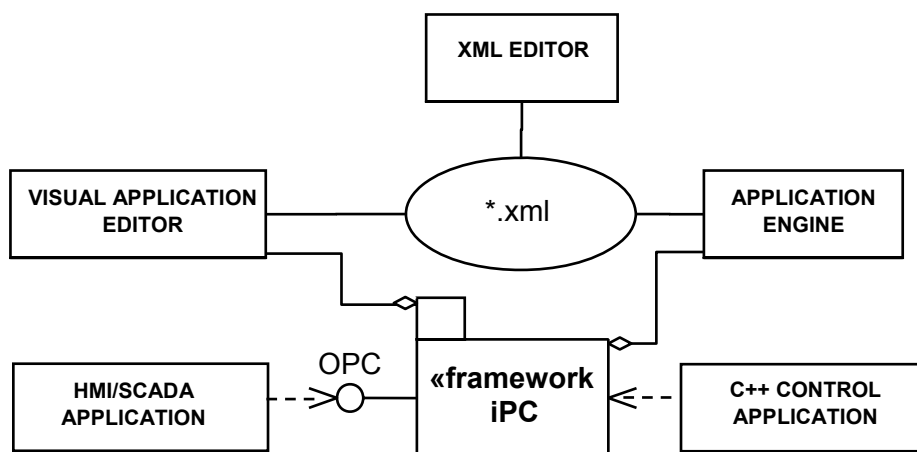
cho riêng mình với ý nghĩa xác định. Vì vậy ta có thể sử dụng XML vào nhiều mục đích khác nhau như:

- Ngôn ngữ mô tả một hệ thống điều khiển hoặc một ứng dụng điều khiển
- Ngôn ngữ mô tả tính năng các thiết bị (*Device Description Language*)
- Xây dựng các trang Web nhúng trong các thiết bị

Lưu trữ thông tin dưới dạng XML cho phép tài liệu vừa dễ đọc với con người, vừa dễ dàng xử lý bởi chương trình máy tính và cho phép các phần mềm khác nhau có thể trao đổi thông tin được với nhau miễn là chúng cùng thống nhất một bộ thẻ đánh dấu. Sự linh hoạt, đa năng và khả chuyển của XML có ý nghĩa đặc biệt đối với các hệ thống phân tán.

8.4.2 Ứng dụng XML trong phần mềm khung iPC

iPC [9] là một phần mềm khung cho các giải pháp điều khiển trên nền máy tính cá nhân, được phát triển trên cơ sở các chuẩn quốc tế IEC 61131, IEC 61499, XML và OPC. Với các thư viện khối chức năng trong phần mềm khung, người sử dụng có thể xây dựng một ứng dụng điều khiển bằng phương pháp *mô tả* hay *cấu hình* thay vì *lập trình*. Thư viện vào/ra cho phép ứng dụng điều khiển giao tiếp với thiết bị cấp trường một cách trong suốt, độc lập với mạng truyền thông vật lý. Bước chuyển một ứng dụng từ chế độ mô phỏng sang điều khiển thời gian thực có thể thực hiện hoàn toàn đơn giản, tự nhiên. Hơn nữa, người sử dụng có thể dễ dàng mở rộng thư viện khối hàm theo nhu cầu, cũng như ghép nối chương trình điều khiển với các phần mềm điều khiển giám sát cấp trên.



Hình 8.5: Mô hình phát triển iPC

Trong kiến trúc phần mềm khung iPC, một ứng dụng điều khiển được mô tả bởi một tập tin XML. Toàn bộ đoạn văn bản mô tả ứng dụng điều khiển được đặt trong cặp thẻ `<controller> ... </controller>`. Người phát triển ứng dụng tạo tập tin này bằng một công cụ soạn thảo ứng dụng (*application editor*) dưới dạng một chương trình soạn thảo XML thông thường hoặc một chương trình soạn thảo đồ họa khối. Một chương trình chạy ứng dụng (*application*

engine) sẽ đọc tập tin XML để vận hành hệ thống. Ví dụ, việc khai báo một bộ điều khiển mờ dạng Mamdani với các số liệu thiết kế trong tập tin LC103.mam được thể hiện như sau:

```
<dll filename="FuzzyFB.dll">
  <FB_list>
    <FB type="MAMDANI" id="LC103"> <init> filename=LC103.mam; </init> </FB>
  </FB_list>
</dll>
```

Tương tự như vậy, các khối chức năng vào/ra tương tự qua mạng Profibus-DP (địa chỉ slave=3, module 3) sử dụng trong mạch vòng điều khiển được khai báo như sau:

```
<dll filename="DPIO.dll">
  <FB_list>
    <FB type="AI" id="AI103"> <init> slave=3; module=3; </init> </FB>
    <FB type="AO" id="AO103"> <init> slave=3; module=3; </init> </FB>
  </FB_list>
</dll>
```

Bước thiết lập một TASK với chu kỳ 100ms để thực hiện mạch vòng điều khiển gồm ba khối hàm AI103, LC103 và AO103 được mô tả như sau:

```
<Task id="ta1" intval="100" res="5" initmode="auto">
  <FB_list> AI103; LC103; AO103; </FB_list>
  <Aconnect> AI103.out -> LC103.in; LC103.out -> AO103.in; </Aconnect>
</Task>
```

Sự kết hợp giữa phương pháp sử dụng các khối hàm như các đối tượng phần mềm độc lập và ngôn ngữ mô tả XML mang lại nhiều ưu điểm quan trọng. Thứ nhất, việc lưu trữ cấu hình hệ thống dưới dạng XML vừa dễ đọc đối với con người, vừa dễ dàng xử lý bởi máy tính. Thứ hai, với một thư viện khối hàm có sẵn, người phát triển ứng dụng *không cần phải biết lập trình*. Mỗi khi thay đổi hoặc bổ sung phần nào trong ứng dụng, hay thay đổi một khối hàm ta chỉ cần sửa các dòng XML tương ứng. Hơn nữa, việc sử dụng các khối hàm vào/ra mô phỏng cho phép chuyển một ứng dụng giữa chế độ *mô phỏng thời gian thực* và chế độ *điều khiển thời gian thực* một cách đơn giản như thay tên thư viện khối hàm tương ứng trong tập tin XML. Trong ví dụ trên đây, để chuyển giữa hai chế độ điều khiển và chế độ mô phỏng ta chỉ cần thay đổi tên thư viện khối hàm vào/ra Profibus-DP "DPIO.DLL" bằng "DPSimIO.DLL".

9 MÔ TẢ HỆ THỐNG ĐIỀU KHIỂN PHÂN TÁN

Mô tả hệ thống là một công việc không thể thiếu được trong thiết kế, xây dựng và phát triển một hệ thống điều khiển. Mô tả hệ thống yêu cầu xây dựng các mô hình và tài liệu kỹ thuật đi kèm. Các mô hình có thể dưới dạng toán học hoặc đồ họa. Nội dung trong phần này chỉ đề cập tới việc mô tả hệ thống ở mức vĩ mô bằng các phương pháp đồ họa, không đi vào mô tả toán học chi tiết từng mạch vòng điều khiển. Qua các tài liệu mô tả hệ thống, các kỹ sư điều khiển và các nhà công nghệ có một ngôn ngữ chung để bàn bạc, trao đổi trước khi tiến hành triển khai một dự án. Cũng qua việc mô tả hệ thống, bản thân các kỹ sư điều khiển cũng đã xây dựng được các mô hình chi tiết cho việc thiết kế cấu hình phần cứng, phát triển ứng dụng điều khiển và giao diện người máy.

9.1 Các phương pháp mô tả đồ họa

Các phương pháp mô tả đồ họa sau đây được xem như chuẩn trong công nghiệp:

- Lưu đồ công nghệ (*process flow diagram*) miêu tả quá trình công nghệ không có các thiết bị đo lường và điều khiển. Hiện nay có nhiều công cụ phần mềm khác nhau hỗ trợ xây dựng lưu đồ công nghệ, ví dụ Microsoft's Visio.
- Lưu đồ ống dẫn và thiết bị (*piping and instrumentation diagram, P&ID*) miêu tả chi tiết quá trình công nghệ kèm theo các thiết bị đo lường và điều khiển cùng các đường liên hệ giữa các thành phần. Tài liệu hình thành thành phục vụ thực hiện chức năng điều khiển quá trình. Hai chuẩn quan trọng liên quan tới các biểu tượng lưu đồ P&ID là ANSI/ISA S5.1 và ANSI/ISA S5.3. Microsoft's Visio cũng là một công cụ thích hợp để xây dựng các lưu đồ P&ID.
- Biểu đồ trình tự chức năng (*sequential function chart, SFC*) biểu diễn các bước thực hiện chức năng của qui trình công nghệ. Tài liệu hình thành phục vụ bài toán điều khiển trình tự và điều khiển logic. Chính vì sự gần gũi với việc mô tả thuật toán điều khiển, SFC cũng được coi là một ngôn ngữ lập trình. SFC được giới thiệu chi tiết trong bài 10.

Gần đây, phương pháp mô hình hóa hướng đối tượng cũng được sử dụng để mô tả toàn bộ hệ thống, sự tương tác giữa các thành phần phần cứng và phần mềm trong hệ thống. Các mô hình hướng đối tượng cũng giúp ích cho việc phát triển các phần mềm ứng dụng. Ngôn ngữ mô hình hóa được sử dụng thống nhất hiện nay là UML.

9.2 Lưu đồ P&ID

Các biểu tượng lưu đồ P&ID được sử dụng tương đối thống nhất trên toàn thế giới. Hai chuẩn do ISA (*Instrument Society of America*) phát hành được chấp nhận rộng rãi trên toàn thế giới là:

- ANSI/ISA S5.1: *Instrumentation Symbols and Identification*.
- ANSI/ISA S5.3: *Graphic Symbols for Distributed Control/Shared Display Instrumentation, Logic and Computer Systems*.

9.2.1 Chuẩn ISA S5.1

Biểu tượng thiết bị













Bảng 9.1 liệt kê các biểu tượng thiết bị trên lưu đồ P&ID. Cần lưu ý một biểu tượng có thể biểu diễn một *thiết bị* hoặc một *chức năng* trong một thiết bị chia sẻ (ví dụ một bộ điều khiển hoặc một màn hình chia sẻ). Ý nghĩa của khái niệm “chia sẻ” ở đây là nhằm phân biệt với các thiết bị đơn lẻ (*discrete instrument*), chuyên dụng cho một mục đích duy nhất, ví dụ một bộ điều chỉnh số đơn lẻ, một đèn hiển thị đơn lẻ. Tùy theo mục đích mô tả mà lưu đồ có thể chứa chi tiết biểu tượng cho từng thiết bị/chức năng, hoặc bỏ qua một số trong trường hợp tương đối hiển nhiên.

Các biểu tượng trên bảng 9.1 mô tả các thiết bị cũng như chức năng mà người vận hành có thể trực tiếp thao tác, sử dụng (*accessible*). Trong trường hợp các thiết bị/chức năng được đặt sau bảng, hoặc người vận hành không được phép can thiệp, đường gạch giữa các biểu tượng cần được vẽ bằng nét đứt.

Biểu tượng các đường tín hiệu và đường nối

Để phân biệt rõ ràng với các đường ống dẫn, tất cả các đường tín hiệu và đường nối khác cần được vẽ nét thanh. Các đường nối được thể hiện bằng các biểu tượng trong bảng 9.2.

Bảng 9.1: Biểu diễn các thiết bị trên lưu đồ P&ID

	Phòng điều khiển trung tâm (Remote)	Vị trí mở rộng (Auxiliary Location)	Hiện trường (Local)
Thiết bị phân cứng đơn lẻ			
Phân cứng chia sẻ - Hiển thị chia sẻ - Điều khiển chia sẻ			
Phần mềm Chức năng máy tính			
Logic chia sẻ Điều khiển logic khả trình			

Thiết bị cho hai biến hoặc
một biến với hai hoặc
nhiều chức năng



Bảng 9.2: Biểu diễn các thiết bị trên lưu đồ P&ID

Tín hiệu không định nghĩa	
Đường nối tới quá trình kỹ thuật, hoặc đường cấp năng lượng cho thiết bị	
Tín hiệu khí nén	
Tín hiệu điện	
Tín hiệu thủy lực	
Tín hiệu điện từ hoặc âm thanh (có dẫn định) *	
Tín hiệu điện từ hoặc âm thanh (không dẫn định)*	
Đường nối nội bộ hệ thống (liên kết phần mềm hoặc dữ liệu)	
Đường nối cơ học	
Ống mao dẫn	

* Các hiện tượng điện từ gồm cả nhiệt, sóng vô tuyến, phóng xạ nguyên tử và ánh sáng.

Thông thường, một đường tính hiệu đủ để biểu diễn liên kết giữa các thiết bị trên lưu đồ P&ID ngay cả khi tồn tại nhiều đường vật lý trong thực tế. Các mũi tên có thể sử dụng bổ sung để làm rõ chiều của luồng thông tin.

Các chữ viết tắt sau đây được dùng để ký hiệu các đường cấp năng lượng:

- AS (*Air supply*): cấp không khí
- ES (*Electric supply*): cấp điện
- GS (*Gas supply*): cấp gas
- HS (*Hydraulic supply*): cấp thủy lực
- NS (*Nitrogen supply*): cấp nito
- SS (*Steam supply*): cấp hơi nước
- WS (*Water supply*): cấp nước

Mức tín hiệu có thể ghi kèm theo ký hiệu các đường cấp, ví dụ ES 24DC ký hiệu đường cấp nguồn 24V một chiều.

Nhãn thiết bị và ký hiệu chức năng

Mỗi thiết bị hoặc chức năng biểu diễn trên lưu đồ cần được phân biệt bởi một nhãn (*tag*). Một nhãn bao gồm phần chữ biểu diễn chức năng và phần số phân biệt vòng kín (*loop*), trong đó phần số có thể mang thông tin về khu vực sản xuất hoặc số thứ tự bản lưu đồ. Ví dụ nhãn FIC-1103 biểu diễn chức năng điều khiển (C) và hiển thị (I) lưu tốc (F), có thể cho vòng điều khiển số 03 trong lưu đồ số 11.

Bảng 9.3 liệt kê ý nghĩa của các chữ cái phân biệt chức năng. Phần biểu diễn chức năng bắt đầu bằng một chữ cái ký hiệu biến đo được hoặc một biến khởi tạo, sau đến các chữ cái ký hiệu chức năng chỉ thị hoặc chức năng bị động. Tiếp nữa là các chữ cái thể hiện chức năng đầu ra theo một thứ tự tùy ý, trừ trường hợp chữ C (control) phải đứng trước V (valve). Các chữ cái phụ nếu có thể sử dụng ngay đằng sau một chữ cái chính để thay đổi ý nghĩa chức năng, ví dụ PD biểu diễn chênh lệch (D) áp suất (P).

Lưu ý rằng, các ký hiệu sử dụng tuân theo chức năng chứ không theo nguyên tắc kết cấu hay nguyên tắc làm việc của thiết bị. Ví dụ, một thiết bị đo lưu lượng theo nguyên tắc chênh lệch áp suất được ký hiệu là FT chứ không phải PDT. Chữ cái đầu tiên ký hiệu biến được đo (đầu ra của quá trình) hoặc biến khởi tạo chứ không phải biến điều khiển (đầu ra điều khiển). Ví dụ, một van điều chỉnh lưu lượng được điều khiển bởi một bộ điều khiển áp suất được ký hiệu là PV chứ không phải FV.

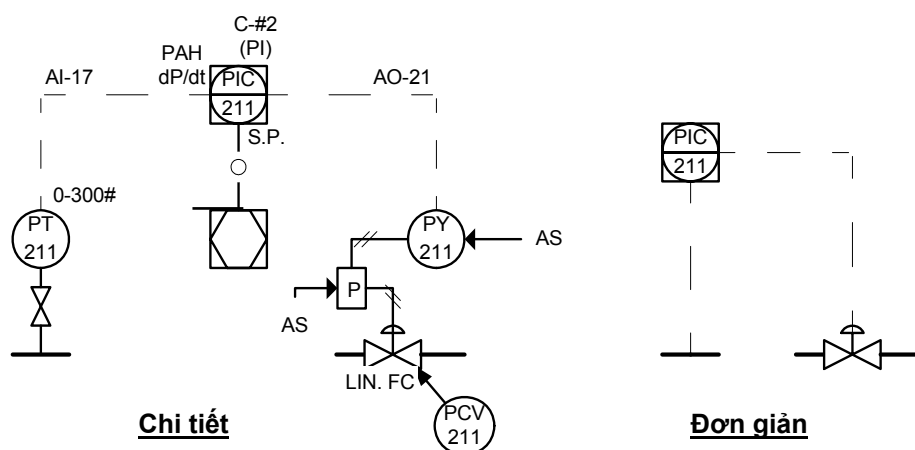
Bảng 9.3: Các chữ cái ký hiệu nhãn thiết bị

Chữ cái đầu			Các chữ cái đứng sau		
	Biến đo được hoặc biến khởi tạo	Thay đổi	Chức năng hiển thị hoặc bị động	Chức năng đầu ra	Thay đổi
A	Phân tích (Analysis)		Báo động (Alarm)		
B	Đốt nóng (Burner, combustion)		Tùy sử dụng	Tùy sử dụng	Tùy sử dụng
C	Tùy sử dụng		Điều khiển (Control)		
D	Tùy sử dụng	Chênh lệch (Differential)			
E	Điện áp		Phần tử sensor		
F	Lưu tốc (Flow rate)	Tỉ lệ (Fraction)			
G	Tùy sử dụng		Kính (Glass), thiết bị nhìn		
H	Bằng tay (Hand)		Cao (High)		
I	Dòng điện		Hiện thị (Indication)		
J	Công suất	Quét			
K	Thời gian, lịch trình	Tần suất thay đổi	Trạm điều khiển		
L	Mức (Level)		Ánh sáng (Light)	Thấp (Low)	
M	Tùy sử dụng	Nhất thời (Momentary)	Trung bình (Middle)		
N	Tùy sử dụng		Tùy sử dụng	Tùy sử dụng	Tùy sử dụng
O	Tùy sử dụng		Hạn chế (Orifice)		
P	Áp suất (Pressure)		Điểm thử (Point)		
Q	Số lượng(Quantity)	Tích phân, tổng số			
R	Bức xạ, phóng xạ (Radiation)		Ghi chép (Record)		
S	Tốc độ, tần số (Speed)	An toàn (Safety)	Chuyển mạch (Switch)		
T	Nhiệt độ (Temperature)		Truyền, phát (Transmit)		
U	Nhiều biến		Đa chức năng	Đa chức năng	Đa chức năng
V	Độ rung (Vibration), phân tích cơ học		Van (Valve), giảm chấn		
W	Trọng lượng (Weight), lực		Giếng (Well), phun		
X	Không xếp loại	Trục X	Không xếp loại	Không xếp loại	Không xếp loại
Y	Sự kiện, trạng thái hoặc sự có mặt	Trục Y	Role, tính toán, biến đổi		
Z	Vị trí, kích thước	Trục Z	Truyền động, chấp hành		

Ví dụ minh họa

Hình 9.1 minh họa một ví dụ mạch vòng điều khiển áp suất. Lưu đồ chi tiết bên trái là cơ sở cho thiết kế hệ thống và phát triển phần mềm, trong khi lưu đồ đơn giản hóa bên phải có thể xuất hiện trên các lưu đồ công nghệ.

Mạch vòng điều khiển áp suất được điều khiển bởi trạm điều khiển DCS. Giá trị đặt được đưa từ một máy tính thông qua đường truyền dữ liệu. Mạch vòng điều khiển có số nhân 211, ký hiệu vòng điều khiển số 11 trên lưu đồ số 2. Thiết bị đo áp suất PT-211 được nối với ống dẫn qua một van khóa và phạm vi làm việc 0-300 PSIG. Tín hiệu ra là dòng điện 4-20mA, được ký hiệu đầu vào AI-17 trong hệ DCS. Bộ điều khiển PIC-211 được thực hiện thuật toán PI trên trạm số 2 (C-2). Đầu ra của bộ điều khiển được ký hiệu AO-21 được đưa vào một bộ chuyển đổi dòng-áp suất (PY-211) gắn trên van điều chỉnh PCV-211. Van điều chỉnh là loại tuyến tính, đóng an toàn, được trang bị bộ định vị (P). Cả bộ định vị và bộ biến đổi được cấp khí nén.



Hình 9.1: Lưu đồ chi tiết một vòng điều khiển áp suất (bên trái) và lưu đồ đơn giản hóa (bên phải)

9.2.2 Chuẩn ISA S5.3

Chuẩn mở rộng ISA S5.1 cho các chức năng trong một hệ điều khiển phân tán. Thực ra, S5.1 vẫn giữ nguyên tập hợp các biểu tượng, nhưng mở rộng và chi tiết hóa ý nghĩa của một số biểu tượng.

Các biểu tượng cho hiển thị/điều khiển chia sẻ

Thông thường người vận hành có thể can thiệp:

- 1) Hiển thị chia sẻ
- 2) Hiển thị chia sẻ và điều khiển chia sẻ
- 3) Truy nhập chỉ qua đường truyền thông
- 4) Giao diện vận hành trên đường truyền thông

Thiết bị lắp tại vị trí mở rộng:





- 1) Lắp trên panel tại vị trí mở rộng, thông thường có giao diện sử dụng tương tự, không lắp tại trạm vận hành trung tâm
- 2) Có thể là một bộ điều khiển dự phòng hoặc một trạm thao tác bằng tay
- 3) Truy nhập có thể chỉ qua đường truyền thông
- 4) Giao diện vận hành qua đường truyền thông.

Thông thường người vận hành không thể can thiệp:



- 1) Bộ điều khiển chia sẻ cam (không có giao diện)
- 2) Hiện thị chia sẻ lắp tại hiện trường
- 3) Tính toán, điều hòa tín hiệu trong bộ điều khiển chia sẻ
- 4) Có thể được nối mạng truyền thông
- 5) Thông thường vận hành không cần giao diện
- 6) Có thể thay đổi khi cấu hình

Các biểu tượng máy tính

Các biểu tượng dưới đây được sử dụng để chỉ các “máy tính” và chức năng máy tính tách biệt với các thành phần cơ bản tích hợp trong hệ DCS. Các máy tính này có thể được nối mạng với các thành phần còn lại, hoặc đứng độc lập.

Thông thường người vận hành có thể can thiệp:



- 1) Màn hình chỉ thị
- 2) Máy tính điều khiển
- 3) Máy tính ghi chép

Bình thường người vận hành không thể can thiệp:



- 1) Giao diện vào/ra
- 2) Tính toán, điều hòa tín hiệu trong máy tính
- 3) Có thể sử dụng như một bộ điều khiển cam hoặc một module phần mềm tính toán.

Các biểu tượng điều khiển trình tự và logic

Biểu tượng cho điều khiển logic hoặc điều khiển trình tự phức tạp không định nghĩa:



Bộ điều khiển logic trong hệ ĐKPT với chức năng điều khiển logic hoặc điều khiển trình tự:



- 1) Bộ điều khiển logic khả trình, hoặc điều khiển logic số tích hợp trong các thiết bị điều khiển phân tán
- 2) Bình thường người vận hành không can thiệp được.

Ý nghĩa như trên, nhưng người vận hành có thể can thiệp:



Tính toán và điều hòa tín hiệu

Ngoài các biểu tượng đã định nghĩa trong S5.1, chuẩn S5.3 còn bổ sung biểu tượng riêng cho tính toán và điều hòa tín hiệu:



9.3 Mô hình hóa hướng đối tượng

Sử dụng ngôn ngữ UML để mô hình hóa hệ thống:

- Từ biểu đồ lớp, biểu đồ triển khai có thể xây dựng cấu hình hệ thống
- Từ biểu đồ trạng thái có thể chuyển sang SFC
- Từ biểu đồ tương tác có thể chuyển sang FBD (CFC)

(Chi tiết về sử dụng UML xem [2])

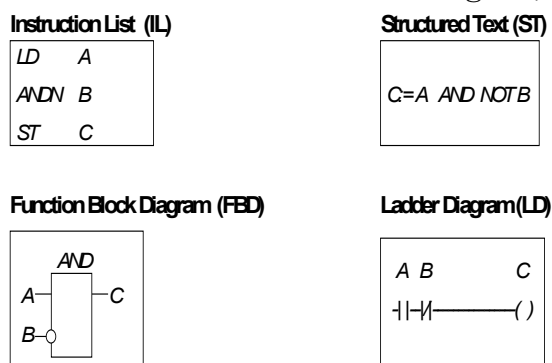
10 LẬP TRÌNH ĐIỀU KHIỂN PHÂN TÁN

10.1 Lập trình theo chuẩn IEC 61131-3

IEC 61131-3 là chuẩn quốc tế duy nhất về ngôn ngữ lập trình cho các thiết bị điều khiển, được chấp nhận rộng rãi trong công nghiệp. Các tiến bộ so với các phương pháp lập trình PLC cổ điển là:

- Các ngôn ngữ lập trình thống nhất
- Một mô hình Task và Resource thích hợp cho nhiều hệ thống khác nhau
- Các kiểu dữ liệu đa dạng, khả mở
- Một thư viện các hàm và khối chức năng chuẩn
- Bước đầu có ý tưởng hướng đối tượng
- Một mô hình giao tiếp thống nhất.

Năm ngôn ngữ được chuẩn hóa là *liệt kê lệnh* (IL, *Instruction List*), sơ đồ tiếp điểm (LD, *Ladder Diagram*), sơ đồ khối chức năng (FBD, *Function Block Diagram*), *văn bản có cấu trúc* (ST, *Structured Text*) và *biểu đồ hoạt động tuần tự* (SFC, *Sequential Function Chart*). Thực ra, trọng tâm của IEC 61131-3 lúc đầu hướng tới các bộ điều khiển khả trình (PLC) đơn lẻ, song một số ngôn ngữ chuẩn hóa ở đây cũng phản ánh các ngôn ngữ lập trình tiêu biểu được dùng trong các hệ điều khiển phân tán. Những ngôn ngữ đó là FBD, ST và SFC, mặc dù trong thực tế chúng có những biến thể khác nhau và được gọi với những cái tên khác nhau. Hai ngôn ngữ còn lại là IL và LD thường chỉ được sử dụng cho lập trình các PLC đơn lẻ, nên không được đề cập ở đây.



Hình 10-1: Minh họa một số ngôn ngữ lập trình PLC

Mỗi ngôn ngữ lập trình như FBD, ST và SFC thích hợp cho việc thể hiện các khía cạnh khác nhau trong một chương trình điều khiển. Cụ thể là, FBD thích hợp cho bài toán điều khiển quá trình và điều khiển logic trên cơ sở các khối chức năng có sẵn, ST thích hợp cho việc thực hiện các khối chức năng, SFC phù hợp cho bài toán điều khiển trình tự và cho biểu diễn trình tự hoạt động trong một chương trình ở mức cao hơn FBD.

10.1.1 Kiểu dữ liệu

Các kiểu dữ liệu cơ sở

Các kiểu dữ liệu cơ sở chuẩn hóa trong IEC 61131-3 được liệt kê trong bảng 10.1, kèm theo qui định về chiều dài ô nhớ và giá trị khởi tạo mặc định. Phạm vi biểu diễn của các số có thể suy ra từ chiều dài ô nhớ. Cách biểu diễn các số thực tuân theo chuẩn IEC 559.

Bảng 10-1: Các kiểu dữ liệu cơ sở trong IEC 61131-3

STT	Từ khóa	Kiểu dữ liệu	Chiều dài (bit)	Giá trị khởi tạo mặc định
1	BOOL	Kiểu Bool	1	0/FALSE
2	SINT	Kiểu nguyên ngắn	8	0
3	INT	Kiểu nguyên	16	0
4	DINT	Kiểu nguyên đúp	32	0
5	LINT	Kiểu nguyên dài	64	0
6	USINT	Kiểu nguyên dương ngắn	8	0
7	UINT	Kiểu nguyên dương	16	0
8	UDINT	Kiểu nguyên dương đúp	32	0
9	ULINT	Kiểu nguyên dương dài	64	0
10	REAL	Số thực	32	0.0
11	LREAL	Số thực dài	64	0.0
12	TIME	Khoảng thời gian	--	T#0S
13	DATE	Ngày tháng	--	D#0001-01-01
14	TIME_OF_DAY TOD	Thời gian trong ngày	--	TOD#00:00:00
15	DATE_AND_TIME DT	Ngày tháng và thời gian	--	DT#0001-01-01- 00:00:00
16	STRING	Chuỗi ký tự 8bit	8 x N	' '
17	BYTE	Chuỗi 8 bit	8	0
18	WORD	Chuỗi 16 bit	16	0
19	DWORD	Chuỗi 32 bit	32	0
20	LWORD	Chuỗi 64 bit	64	0
21	WSTRING	Chuỗi ký tự 16bit	16 x N	" "

Các kiểu dữ liệu tổng quát

Bên cạnh các kiểu dữ liệu cơ sở, chuẩn còn đưa ra khái niệm “kiểu dữ liệu tổng quát”. Thực chất đây là các từ khóa đại diện cho một nhóm kiểu có thể sử dụng trong một số hàm và khối chức năng tổng quát. Việc sử dụng các từ khóa này nhằm đơn giản hóa việc mô tả các hàm và khối chức năng có thể áp dụng cho nhiều kiểu dữ liệu khác nhau. Ngoài ra, các kiểu dữ liệu tổng quát

không có ý nghĩa nào khác trong một chương trình, do đó người sử dụng không thể khai báo một biến thuộc kiểu tổng quát. Quy định về các kiểu dữ liệu tổng quát được minh họa trong bảng 10-2.

Bảng 10-2: Các kiểu dữ liệu tổng quát trong IEC 61131-3

ANY
ANY_DERIVED
ANY_ELEMENTARY
ANY_MAGNITUDE
ANY_NUM
ANY_REAL
LREAL
REAL
ANY_INT
LINT, DINT, INT, SINT
ULINT, UDINT, UINT, USINT
TIME
ANY_BIT
LWORD, DWORD, WORD, BYTE, BOOL
ANY_STRING
STRING
WSTRING
ANY_DATE
DATE_AND_TIME
DATE, TIME OF DAY

Các kiểu dữ liệu dẫn xuất

Các kiểu dẫn xuất được tóm tắt trong bảng 10.3, gồm có:

- Các kiểu dẫn xuất trực tiếp
- Các kiểu liệt kê
- Các kiểu dãy con
- Các kiểu mảng
- Các kiểu cấu trúc

Bảng 10-3: Các kiểu dữ liệu dẫn xuất trong IEC 61131-3

STT	Mô tả/ví dụ
1	Dẫn xuất trực tiếp từ kiểu cơ sở, ví dụ TYPE RU_REAL : REAL ; END_TYPE
2	Các kiểu liệt kê, ví dụ: TYPE ANALOG_SIGNAL_TYPE: (SINGLE_ENDED, DIFFERENTIAL); END_TYPE
3	Các kiểu dãy con, ví dụ: TYPE ANALOG_DATA: INT (-4095..4095); END_TYPE
4	Các kiểu mảng, ví dụ: TYPE ANALOG_16_INPUT_DATA: ARRAY [1..16] OF ANALOG_DATA; END_TYPE

STT	Mô tả/ví dụ
5	<p>Các kiểu cấu trúc, ví dụ:</p> <pre> TYPE ANALOG_CHANNEL_CONFIGURATION: STRUCT RANGE : ANALOG_SIGNAL_RANGE ; MIN_SCALE : ANALOG_DATA ; MAX_SCALE : ANALOG_DATA ; END_STRUCT ; ANALOG_16_INPUT_CONFIGURATION : STRUCT SIGNAL_TYPE : ANALOG_SIGNAL_TYPE ; FILTER_PARAMETER : SINT (0..99) ; CHANNEL: ARRAY [1..16] OF ANALOG_CHANNEL_CONFIGURATION; END_STRUCT ; END TYPE </pre>

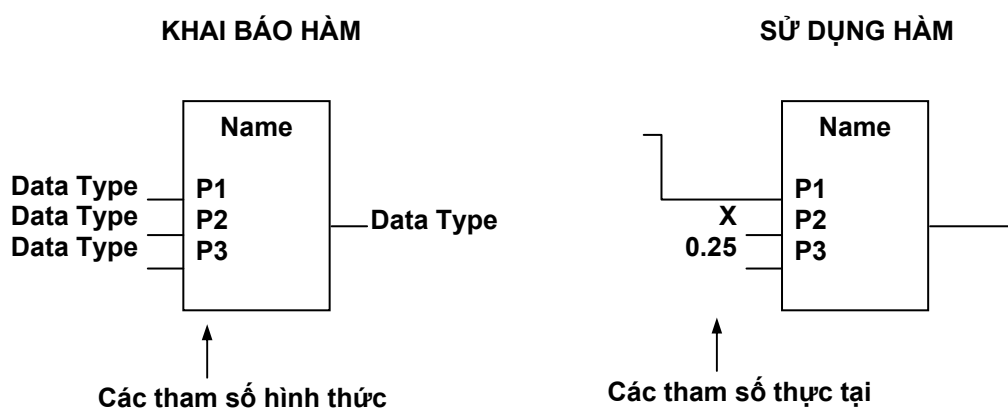
10.1.2 Tổ chức chương trình

Chương trình (PROGRAM), khối chức năng (FUNCTION_BLOCK) và hàm (FUNCTION) là các đơn vị tổ chức chương trình (*Program Organization Unit*), POU).

FUNCTION

Giống như trong PASCAL, một hàm có một hoặc nhiều đầu vào và chính xác một đầu ra. Một hàm không có trạng thái, có nghĩa là gọi hàm với các giá trị đầu vào như nhau sẽ cho kết quả như nhau.

Ví dụ về hàm là các hàm toán học SIN, COS, ADD,... hoặc các hàm logic AND, OR,... Một hàm được biểu diễn đồ họa như trên Hình 10-2.



Hình 10-2: Khai báo và sử dụng một hàm

Mã khai báo/định nghĩa và gọi hàm dưới dạng ST được minh họa dưới đây:

```

(* Khai báo và định nghĩa hàm *)
FUNCTION fct1 : REAL
  VAR_INPUT
    a, b: REAL;
    c : REAL:= 1.0;
  END_VAR
  fct1 := a*b/c;

```

```
END_FUNCTION
```

```
(* Gọi hàm *)
```

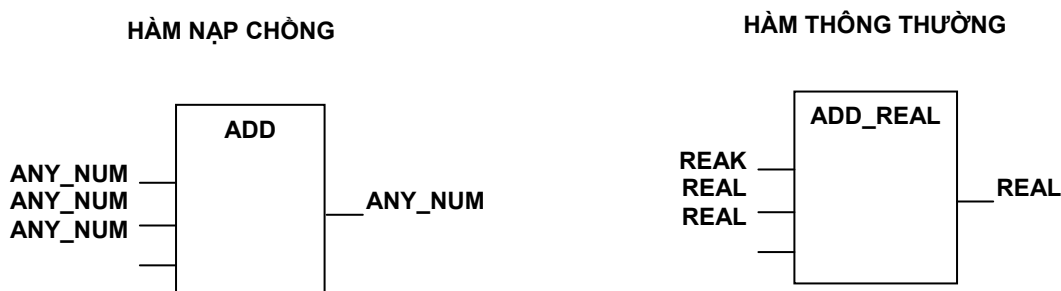
```
...
```

```
y := fct1(a:= x, b:= 2.0);
```

```
...
```

Cần lưu ý:

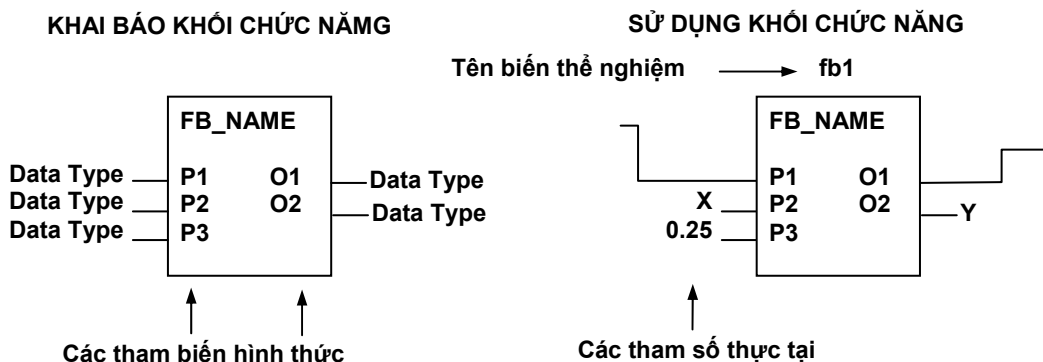
- Khai báo VAR_INPUT .. END_VAR cho các biến đầu vào
- Khai báo VAR .. END_VAR cho các biến cục bộ
- Kết quả trả về được gán cho tên hàm
- Các hàm cũng có thể được nạp chồng để có thể áp dụng với nhiều kiểu dữ liệu khác nhau (Hình 10-3).



Hình 10-3: Khai báo hàm nạp chồng và hàm thông thường

FUNCTION_BLOCK

Hàm và khối chức năng là các khối tổ chức chương trình có giá trị sử dụng lại. Một khối chức năng giống như một đối tượng, có trạng thái và có thể có nhiều đầu ra, ví dụ các bộ điều khiển động, các khâu Flip-Flop, Timer, Counter. Nếu so sánh với một đối tượng thì một khối chức năng chỉ có một hàm thành viên duy nhất, vì thế cách gọi một khối chức năng tương tự như cách gọi một hàm. Mỗi khối chức năng có một trạng thái riêng nên ta phải phân biệt giữa *kiểu khối chức năng* (giống như lớp) và *thể nghiệm khối chức năng* (giống như đối tượng). Tùy theo ngữ cảnh mà khái niệm *khối chức năng* có thể hiểu theo một trong hai nghĩa trên. Việc gọi một khối chức năng cũng được thông qua tên của biến thể nghiệm, chứ không thông qua tên kiểu khối chức năng, như minh họa trên Hình 10-4.



Hình 10-4: Khai báo và sử dụng một khối chức năng

Dưới đây là ví dụ mã một khối chức năng, viết bằng ST.

```
FUNCTION_BLOCK Example
VAR_INPUT
    X :    BOOL;
    Y :    BOOL;
END_VAR
VAR_OUTPUT
    Z :    BOOL;
END_VAR
VAR
    INTERNAL_STATE: BOOL;
END_VAR
(* statements of functionblock body *)
END_FUNCTION_BLOCK
```

Cần lưu ý:

- Khai báo biến đầu vào với VAR_INPUT .. END_VAR
- Khai báo biến đầu ra với VAR_OUTPUT .. END_VAR
- Khai báo biến vào/ra với VAR_IN_OUT .. END_VAR
- Khai báo các biến nội bộ (trạng thái) với VAR .. END_VAR

PROGRAM

Một chương trình có thể coi như một mạng các hàm và khối chức năng nối với nhau. Khác với hàm và khối chức năng, một chương trình có thể sử dụng các biến toàn cục, các biến vào/ra quá trình hoặc các biến định địa chỉ trực tiếp. Một biến có thể thuộc các kiểu dữ liệu cơ bản hoặc dẫn xuất đã nêu, nhưng cũng có thể là một khối chức năng. Như vậy, các khối chức năng có giá trị sử dụng lại, trong khi chương trình thì không. Một chương trình hoặc một khối chức năng có thể được thực hiện tuần hoàn, theo sự kiện hay liên tục nhờ các tác vụ (TASK) tương ứng. Nhiều chương trình và khối chức năng có thể chia sẻ cùng một tác vụ.

10.1.3 Ngôn ngữ FBD

FBD là một trong năm ngôn ngữ được định nghĩa trong IEC 61131-3. Ý tưởng của ngôn ngữ này là xây dựng chương trình từ các *khối chức năng* (*function block*) kết nối với nhau bằng tín hiệu. Mỗi khối chức năng thực hiện một nhiệm vụ xử lý nào đó. Chương trình xây dựng bằng ngôn ngữ FBD có hình thức rất giống sơ đồ khối của hệ thống điều khiển. Thực chất, FBD là sự kết hợp của sơ đồ logic cổ điển (*binary logic diagram*) và biểu đồ hoạt động liên tục (*continuous function chart*, CFC). Mỗi khối trong FBD có thể là một khối logic như AND, OR,... nhưng cũng có thể là một khối có các đầu vào/ra tương tự như ADD, PID, AI, AO,...

FBD thường được sử dụng để thực hiện một phản tương đối độc lập của chương trình điều khiển trên cơ sở các khối chức năng có sẵn, ví dụ lập trình các mạch vòng điều khiển quá trình hoặc các mạch logic điều khiển khóa liên động.

10.1.4 Ngôn ngữ ST

ST (Structured Text) là một ngôn ngữ bậc cao mới được xây dựng trên cơ sở cú pháp PASCAL, nhờ vậy rất thuận tiện cho việc thể hiện các thuật toán phức tạp trong điều khiển quá trình.

Bảng 10-4: Lệnh trong ST

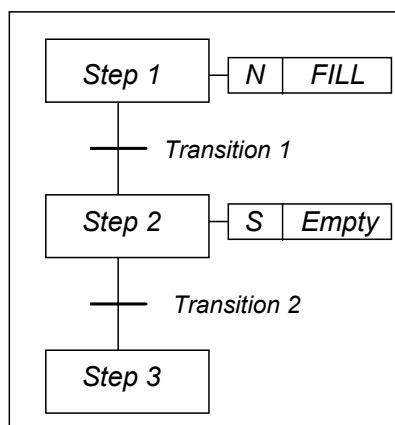
No.	Kiểu lệnh	Ví dụ
1	Gán	<code>A := B; CV := CV+1; C := SIN(X);</code>
2	Sử dụng hàm	<code>CMD_TMR(IN:=IX5, PT:=T#300ms); A := CMD_TMR.Q;</code>
3	Quay về RETURN	<code>RETURN;</code>
4	Điều kiện rẽ nhánh if	<code>D := B*B - 4*A*C; IF D < 0.0 THEN NROOTS := 0; ELSIF D = 0.0 THEN NROOTS := 1; X1 := - B/(2.0*A); ELSE NROOTS := 2; X1 := (- B + SQRT(D))/(2.0*A); X2 := (- B - SQRT(D))/(2.0*A); END IF;</code>
5	Phân nhánh CASE	<code>TW := BCD_TO_INT(THUMBWHEEL); TW_ERROR := 0; CASE TW OF 1,5: DISPLAY := OVEN_TEMP; 2: DISPLAY := MOTOR_SPEED; 3: DISPLAY := GROSS - TARE; 4,6..10: DISPLAY := STATUS(TW - 4); ELSE DISPLAY := 0; TW_ERROR := 1; END_CASE; QW100 := INT_TO_BCD(DISPLAY);</code>
6	Vòng lặp FOR	<code>J := 101; FOR I := 1 TO 100 BY 2 DO IF WORDS[I] = 'KEY' THEN J := I; EXIT; END_IF; END_FOR;</code>
7	Vòng lặp WHILE (3.3.2.4)	<code>J := 1; WHILE J <= 100 & WORDS[J] <> 'KEY' DO J := J+2; END WHILE;</code>
8	Vòng lặp REPEAT (3.3.2.4)	<code>J := -1; REPEAT J := J+2; UNTIL J = 101 OR WORDS[J] = 'KEY' END_REPEAT;</code>
9	Kết thúc EXIT (3.3.2.4) ^a	<code>EXIT;</code>
10	Lệnh rỗng	<code>;</code>

10.1.5 Ngôn ngữ SFC

SFC mô tả tiến trình hoạt động tuần tự của một chương trình điều khiển. SFC được xây dựng trên cơ sở mạng Petri và chuẩn IEC 848 Grafcet, với các thay đổi cần thiết để có thể thực hiện chức năng điều khiển thay vì chỉ là một chuẩn tài liệu mô tả.

SFC bao gồm các bước (*Step*) và các chuyển tiếp (*Transition*). Mỗi bước đại diện cho một trạng thái cụ thể của hệ thống được điều khiển. Một chuyển tiếp được gán một điều kiện logic, khi trở thành “đúng” sẽ kết thúc bước và kích hoạt bước sau. Một bước được liên kết với một khối hành động (*Action Block*) để thực hiện các thao tác điều khiển. Các khối hành động trong SFC đều có thể được lập trình bởi một ngôn ngữ IEC bất kỳ, kể cả chính SFC, trong khi các điều kiện chuyển tiếp có thể được lập trình bằng bốn ngôn ngữ IL, LD, FBD hoặc ST. SFC cũng cho phép mô tả các quá trình phân nhánh cạnh tranh hoặc các quá trình phân nhánh song song.

SFC cho phép sử dụng các bước lớn (*macro step*), trong đó mỗi bước lớn lại có thể được biểu diễn bằng một SFC khác. Nhờ vậy ta có thể mô hình hóa ứng dụng, lập trình ứng dụng ở nhiều mức trừu tượng, tạo điều kiện thuận lợi cho giải quyết các bài toán phức tạp.



Hình 10-5: Một ví dụ SFC đơn giản

10.2 Lập trình với ngôn ngữ bậc cao

Bên cạnh sử dụng các ngôn ngữ lập trình chuyên dụng, việc lập trình với ngôn ngữ bậc cao phục vụ các mục đích sau đây:

- Mở rộng thư viện khối chức năng
- Thể hiện một số chức năng mở rộng
- Lập trình cho hệ DCS trên nền PC

Ngôn ngữ thường được sử dụng nhất là C/C++. Với các công cụ mạnh trong tay, việc lập trình các thuật toán điều khiển trở nên dễ dàng. Bên cạnh các luật điều khiển logic hoặc các bộ điều khiển phản hồi PID, thì khả năng thực

hiện các thuật toán điều khiển cao cấp cũng hoàn toàn không bị giới hạn bởi ngôn ngữ lập trình. Các luật điều khiển này có thể được sử dụng lại nhiều lần, vì vậy có thể xây dựng thành một thư viện lớp. Bên cạnh các yếu tố như tham số bộ điều khiển và chu kỳ lấy mẫu, cần lưu ý giải quyết hợp lý các vấn đề thực tế như giới hạn đầu vào, chuyển đổi chế độ điều khiển bằng tay/tự động, hiệu ứng *reset windup* (ở bộ điều khiển PID).

Để đưa các mã viết bằng C/C++ bổ sung vào thư viện khối chức năng trong một hệ DCS, người lập trình cần tuân thủ theo một qui định nhất định. Thông thường, việc này không dễ dàng và đòi hỏi tính năng mở của hệ thống.

11 CHỨC NĂNG ĐIỀU KHIỂN GIÁM SÁT

11.1 Giới thiệu chung về các hệ điều khiển giám sát

Điều khiển giám sát và thu thập dữ liệu (SCADA) là một phần không thể thiếu được trong một hệ thống tự động hóa hiện đại. Từ những năm gần đây, tiến bộ trong các lĩnh vực truyền thông công nghiệp và công nghệ phần mềm đã thực sự đem lại nhiều khả năng mới, giải pháp mới.

Giống như nhiều từ viết tắt có tính chất truyền thống khác, khái niệm SCADA (*Supervisory Control And Data Acquisition*) cũng được hiểu với những ý nghĩa hơi khác nhau, tùy theo lĩnh vực ứng dụng và theo thời gian. Có thể, khi nói tới SCADA người ta chỉ liên tưởng tới một hệ thống mạng và thiết bị có nhiệm vụ thuần túy là thu thập dữ liệu từ các trạm ở xa và truyền tải về một khu trung tâm để xử lý. Các hệ thống ứng dụng trong công nghiệp khai thác dầu khí và phân phối năng lượng là những ví dụ tiêu biểu. Theo cách hiểu này, vấn đề *truyền thông* được đặt lên hàng đầu. Trong nhiều trường hợp, các khái niệm SCADA và "None-SCADA" lại được dùng để phân biệt các giải pháp điều khiển giám sát dùng công cụ phần mềm chuyên dụng (ví dụ FIX, InTouch, WinCC, Lookout, ...) hay phần mềm phổ thông (Access, Excel, Visual Basic, Delphi, JBuilder, ...). ở đây, *công nghệ phần mềm* là vấn đề được quan tâm chủ yếu.

Nói một cách khái quát, một hệ SCADA không có gì khác là một hệ thống điều khiển giám sát, tức là một hệ thống hỗ trợ con người trong việc quan sát và điều khiển từ xa, ở cấp cao hơn hệ điều khiển tự động thông thường. Đương nhiên, để có thể quan sát và điều khiển từ xa cần phải có hệ thống truy nhập (không chỉ thu thập!) và truyền tải dữ liệu, cũng như cần phải có giao diện người-máy (Human-Machine Interface, HMI). Tùy theo trọng tâm của nhiệm vụ mà người ta có thể có những cách nhìn khác nhau.

Như ta thấy, HMI là một thành phần trong một hệ SCADA, tuy nhiên không phải chỉ ở cấp điều khiển giám sát, mà ngay ở các cấp thấp hơn người ta cũng cần giao diện người-máy phục vụ việc quan sát và thao tác vận hành cục bộ. Vì lý do giá thành, đặc tính kỹ thuật cũng như phạm vi chức năng, ở các cấp gần với quá trình kỹ thuật này các OP chuyên dụng chiếm vai trò quan trọng hơn.

Sự tiến bộ trong công nghệ phần mềm và kỹ thuật máy tính PC, đặc biệt là sự chiếm lĩnh thị trường của hệ điều hành Windows NT cùng với các công nghệ của Microsoft đã thúc đẩy sự phát triển của các công cụ tạo dựng phần mềm SCADA theo một hướng mới, sử dụng PC và Windows NT làm nền phát triển và cài đặt. Từ phạm vi chức năng thuần túy là thu thập dữ liệu cho việc quan sát, theo dõi quá trình, một hệ SCADA ngày nay có thể đảm nhiệm vai trò điều khiển cao cấp, điều khiển phối hợp. Phương pháp điều khiển theo mẻ, điều khiển theo công thức (batch control, recipe control) là những ví dụ tiêu

biểu. Hơn nữa, khả năng tích hợp hệ điều khiển giám sát với các ứng dụng khác như các phần mềm quản lý, tối ưu hóa hệ thống,... của toàn công ty cũng trở nên dễ dàng hơn.

Trong giải pháp điều khiển phân tán, hệ thống truyền thông ở các cấp dưới (bus trường, bus chấp hành-cảm biến) đã có sẵn. Nếu như mạng máy tính của một công ty cũng đã được trang bị (chủ yếu dùng Ethernet), thì cơ sở hạ tầng cho việc truyền thông không còn là vấn đề lớn phải giải quyết. Chính vì vậy, trọng tâm của việc xây dựng các giải pháp SCADA trong thời điểm hiện nay là vấn đề lựa chọn công cụ phần mềm thiết kế giao diện và tích hợp hệ thống.

Trong một hệ điều khiển phân tán, chức năng SCADA là một thành phần không thể thiếu được. Như vậy có thể nói, một hệ DCS bao giờ cũng có chức năng SCADA, trong khi một hệ SCADA theo đúng nghĩa của nó không thể là một hệ DCS.

11.1.1 Các thành phần chức năng cơ bản

Xét một cách tổng quát, một hệ SCADA bao gồm các thành phần chức năng liệt kê dưới đây.

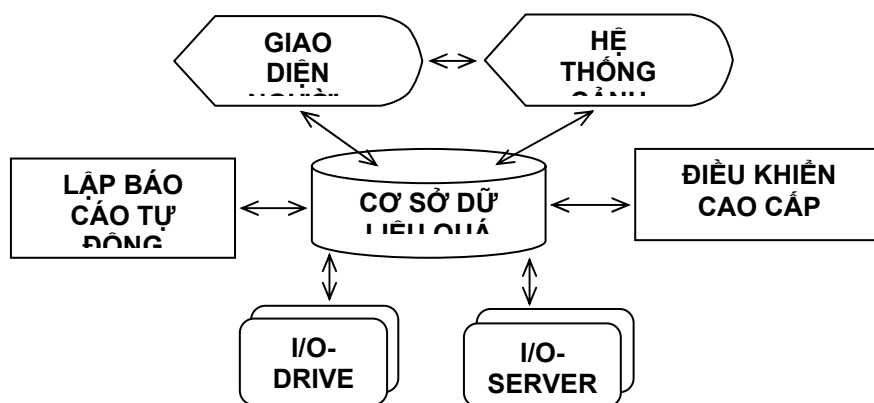
Phần cứng

- Thiết bị thu thập dữ liệu: PLC, RTU, PC, I/O, các đầu đo thông minh
- Hệ thống truyền thông: Mạng truyền thông, các bộ dồn kênh/phân kênh, Modem, các bộ thu phát.
- Trạm quản lý dữ liệu: Máy chủ (PC, Workstation), các bộ tập trung dữ liệu (Data concentrator, PLC, PC)
- Trạm vận hành (Operator Station)

Hình 11-1: Các thành phần phần mềm trong một hệ SCADA

Phần mềm:

- Giao diện vào/ra (phần mềm giao diện quá trình), dưới dạng các I/O-Drivers, I/O-Servers (DDE, OPC,...).
- Giao diện người-máy



- Cơ sở dữ liệu quá trình
- Hệ thống cảnh báo, báo động
- Lập báo cáo tự động
- Điều khiển cao cấp: Điều khiển mẻ, điều khiển trình tự, điều khiển công thức, điều khiển chuyên gia...

Các thành phần nói trên đã được tích hợp trong một hệ điều khiển phân tán. Vì vậy, việc xây dựng các chức năng SCADA ở đây đơn giản hơn nhiều so với trong các hệ khác.

11.1.2 Công cụ phần mềm SCADA/HMI

Phân loại theo phạm vi sử dụng:

- Công cụ lập trình phổ thông
- Công cụ tích hợp trong một hệ DCS
- Công cụ độc lập, có thể sử dụng cho nhiều hệ thống khác nhau: WinCC (Siemens), InTouch (Wonderware), iFIX (Intellution), Genesis (Iconics), LookOut (NI)

Phân loại theo kiến trúc phần mềm:

- Kiến trúc truyền thống
- Kiến trúc hướng đối tượng
- Kiến trúc Web

Kiến trúc hướng đối tượng

Hiện nay, có lẽ không một phần mềm SCADA nào tự nhận là tiên tiến mà không đưa từ khóa *hướng đối tượng* vào danh sách các đặc tính ưu việt để quảng cáo. Mặc dù trong đại đa số các trường hợp, cách sử dụng thuật ngữ như vậy mang tính chất lạm dụng, nhưng qua đó ta cũng thấy ít hay nhiều tầm quan trọng của công nghệ đối tượng. Thực chất, các thư viện thành phần sẵn có trong những sản phẩm thuộc thế hệ mới thường được xây dựng trên cơ sở một mô hình đối tượng, đặc biệt phải nói tới mô hình COM của Microsoft.

Việc sử dụng một mô hình đối tượng thành phần chuẩn công nghiệp như COM mang lại nhiều ưu thế như:

- Nâng cao hiệu suất công việc thiết kế, xây dựng giao diện người-máy bằng cách sử dụng ActiveX-Controls

- Nâng cao khả năng tương tác và khả năng mở rộng, hay nói cách khác là tính năng mở của hệ thống
- Thuận lợi trong việc sử dụng một chuẩn giao diện quá trình như OPC để kết nối với các thiết bị cung cấp dữ liệu.

Quả thật, hầu hết (nếu không nói là tất cả) các phần mềm SCADA tiên tiến nhất hiện nay đều hỗ trợ COM, cụ thể là đều có ba đặc điểm nêu trên. Nếu trước đây để tạo được một màn hình giao diện đồ họa, một kỹ sư có kinh nghiệm cần trung bình một vài ngày, thì nay thời gian có thể giảm xuống tới một vài giờ. Sử dụng một công cụ thích hợp, ta có thể hoàn toàn tập trung vào công việc chính mà không cần kiến thức chuyên sâu về lập trình. Công nghệ đối tượng thành phần và các phương pháp *không lập trình* đã mở ra khả năng này.

Kiến trúc Web

Chỉ trong vòng mấy năm gần đây, Web đã phát triển thành một trong những công nghệ tin học quan trọng bậc nhất. Không chỉ là một phương tiện truyền bá và truy nhập thông tin đại chúng như trong thời gian đầu mới hình thành, bản thân Web hiện đã trở thành một môi trường đa năng cho các ứng dụng phân tán.

Một ứng dụng Web cấu thành bởi sự tương tác giữa Web server, Web browser, trang HTML (*Hypertext Markup Language*) với các ứng dụng và phụ kiện khác. Những ứng dụng đó cũng hay được gọi là các ứng dụng nhiều lớp (*multi-tier applications*).

Sử dụng Web làm nền cho các ứng dụng SCADA không chỉ mang lại hiệu quả về thời gian cài đặt phần mềm, mà trước tiên là mở ra khả năng mới cho việc tích hợp hệ thống tự động hóa trong một hệ thống thông tin thống nhất của công ty. Điều khiển giám sát không còn là chức năng độc quyền của các chuyên viên kỹ thuật. Một giám đốc điều hành sản xuất, hay một tổng giám đốc công ty đều có thể quan sát và tham gia điều hành quá trình sản xuất từ phòng làm việc riêng, chỉ qua màn hình, bàn phím và chuột. Tương tự, các báo cáo về tình hình sản xuất cũng như các chỉ thị không nhất thiết phải đi theo con đường giấy tờ hay truyền miệng, mà trực tiếp diễn ra "on-line".

Đưa SCADA lên Web cũng tạo điều kiện thuận lợi cho các dịch vụ bảo trì hệ thống từ xa. Các nhà cung cấp giải pháp tự động hóa không cần phải đến trực tiếp cơ sở sản xuất, mà có thể theo dõi toàn bộ diễn biến quá trình kỹ thuật qua một trình duyệt như Internet Explorer hay Netscape Navigator, trên cơ sở đó có thể chẩn đoán, xác định lỗi và đưa ra phương hướng giải quyết thích hợp.

Có thể nói, một trong những yếu tố mang lại thành công cho công nghệ Web là mô hình đơn giản, không phụ thuộc vào nền triển khai và sử dụng. Hầu hết các loại máy tính, hầu hết các hệ điều hành thông dụng đều cho

phép cài đặt Web server và Web browser. Tuy nhiên, mô hình Web nguyên sơ thể hiện rõ những yếu điểm sau:

- Giao thức HTTP có tính năng kém, không thích hợp cho các ứng dụng mang tính tương tác và thời gian thực.
- Sự nghèo nàn của HTML.
- Cấu trúc xử lý nhiều lớp mặc dù linh hoạt nhưng hiệu suất kém.
- Độ an toàn và độ tin cậy của Web còn rất thấp.

Để khắc phục những yếu điểm trên, người ta đã đưa ra nhiều giải pháp khác nhau. Trong phạm vi hẹp của báo cáo tác giả không thể đưa ra đầy đủ, cũng như không thể đi sâu vào chi tiết từng giải pháp, nhất là khi các tiến bộ công nghệ Web hiện nay được cập nhật không ngừng. Nhìn một cách tổng quát, ta có thể tóm tắt lại những kỹ thuật chính được áp dụng hiện nay như sau:

- Dùng ActiveX-Controls hoặc JavaApplets kết hợp với các biên ngữ (*scripting language*) như VBScript, JavaScript,... hoặc dùng Dynamic-HTML để nâng cao tính tương tác và khả năng lập trình cho trang Web.
- Sử dụng giao thức riêng biệt kết hợp với ActiveX-Controls hoặc JavaApplets và bỏ qua HTTP nhằm tăng hiệu suất của ứng dụng. Kỹ thuật này thường được các nhà sản xuất ActiveX-Controls hoặc JavaApplets áp dụng trong các sản phẩm của mình.
- Dùng *plug-ins* trong Web server và Web browser để mở rộng, cải tiến chức năng cho các ứng dụng. Kỹ thuật này được dùng chẳng hạn trong các sản phẩm của công ty Netscape Communications.
- Hầu hết các Web server và Web browser đều dành sẵn các giao diện lập trình (APIs) để tạo điều kiện cho người dùng mở rộng, cải tiến chức năng cho các ứng dụng.
- Dùng sản phẩm Web server và Web browser riêng để tối ưu tính năng vận hành của ứng dụng. Kỹ thuật này được dùng chẳng hạn trong Scout - một bộ chương trình quan sát tiến trình dựa Web của công ty Wonderware.

11.2 Xây dựng cấu trúc hệ thống

Đặt vấn đề

- Yêu cầu thực tế của ứng dụng: qui mô hệ thống, giá thành
- Hiệu năng của hệ thống: Khả năng cập nhật dữ liệu, cập nhật màn hình, hiệu suất sử dụng đường truyền
- Độ tin cậy của hệ thống
- Khả năng hỗ trợ bởi các công cụ phần mềm

Cấu trúc một người sử dụng (single-user)

- Ghép nối điểm-điểm
- Ghép nối nhiều điểm: vai trò của mạng truyền thông

Cấu trúc nhiều người sử dụng (multi-user)

- Một màn hình/một trạm
- Nhiều màn hình/một trạm
- Kiến trúc Client/Server: Cấu trúc mạng phẳng và cấu trúc mạng phân cấp

Cấu trúc hệ thống qui mô lớn:

- Số lượng lớn các thiết bị thu thập dữ liệu: Vai trò của các bộ tập trung dữ liệu
- Hệ thống diện rộng: Vai trò các bộ dồn kênh/phân kênh, các bộ thu/phát, modem, mạng viễn thông, Internet.

11.3 Thiết kế giao diện người-máy

Giao diện người-máy là hệ thống phần mềm hỗ trợ con người theo dõi quá trình các diễn biến của kỹ thuật, trạng thái và các thông số làm việc của các thiết bị trong hệ thống, qua đó có thể thực hiện các thao tác vận hành và can thiệp từ xa tới hệ thống điều khiển phía dưới. Ngày nay, các phần mềm giao diện người-máy chủ yếu được xây dựng trên nền máy tính cá nhân, dựa trên các kỹ thuật đồ họa hiện đại. Giao diện người-máy là một trong các thành phần chính của một hệ thống điều khiển giám sát.

11.3.1 Yêu cầu chung

- Đơn giản, dễ sử dụng (*easy-to-use*)
- Bền vững, khó gây lỗi (*robustness*)
- Tính thông tin cao (*informativeness*)
- Nhất quán (*consistency*)
- Đẹp, nhã nhặn (*good-looking, elegant*)

11.3.2 Các phương pháp giao tiếp người-máy

- Đưa lệnh trực tiếp
- Lựa chọn lệnh từ menu
- Giao tiếp qua hộp thoại

11.3.3 Thiết kế cấu trúc màn hình

Yêu cầu cấu trúc các màn hình

- Gắn với các máy móc, thiết bị, công nghệ thực
- Khoa học, kết hợp hợp lý phương pháp sử dụng chuyển cấp hoặc lựa chọn nhanh
-

Phân cấp màn hình

- Tổng quan hệ thống (*system overview*), hệ thống con (*subsystem overview*)
- Tổng quan nhóm (*group overview*)
- Hiển thị nhóm (*group display*)

- Hiển thị chi tiết (*details display*)
- Hình ảnh hệ thống, hình ảnh phạm vi/công đoạn/máy móc dưới dạng lưu đồ công nghệ (*process diagram*) hoặc hình ảnh dây chuyền sản xuất
- Đồ thị (*trends*): Đồ thị thời gian thực, đồ thị quá khứ
- Cửa sổ báo động (*alarm windows*)

11.3.4 Các nguyên tắc thiết kế

Màu sắc

- Chỉ dùng màu sắc khi thật cần thiết
- Nền: màu tối, ví dụ xám sẫm hoặc xanh lam đậm
- Máy móc, thiết bị: Sử dụng hình phẳng, màu và độ sáng khác ít so với nền, cố gắng tránh 3D, tránh các mẫu hoa văn
- Hình tĩnh (đường ống, máy móc): tránh các màu tươi, chói
- Tín hiệu trạng thái, hình động: Chọn các màu tươi, chói

Chữ viết

- Hạn chế số font chữ, kiểu chữ, chên lệch độ lớn
- Chân phương, tránh các hiệu ứng đặc biệt (3D, lượn sóng, đường viền,...)

Các hình ảnh động

- Hỗ trợ phân biệt trạng thái, ví dụ nhấp nháy
- Nhất quán trong tất cả các màn hình
- Các số nên chỉnh căn phải, các biến liên quan trực tiếp để gần nhau và cùng cách biểu diễn
- Biểu diễn các đơn vị vật lý với giá trị số và đơn vị, không dùng %

12 TÍNH SẴN SÀNG VÀ ĐỘ TIN CẬY CỦA CÁC HỆ ĐKPT

12.1 Đặt vấn đề

Tính sẵn sàng và độ tin cậy của hệ thống phụ thuộc vào:

- Độ tin cậy của từng thiết bị
- Cấu trúc hệ thống
- Tính năng hệ thống truyền thông
- Cơ chế dự phòng
- Cơ chế an toàn
- Cơ chế khởi động lại sau sự cố nguồn
- Cơ chế bảo mật
- Khả năng bảo trì
- ...

12.2 Cơ chế dự phòng

Yêu cầu:

- Các thành phần quan trọng cần được dự phòng hoàn toàn để trường hợp lỗi một thành phần đơn (phần cứng & phần mềm) không làm mất đi tính năng do nó cung cấp
- Lỗi mỗi module hoặc card được phép không gây ra tê liệt hơn một trạm vận hành hoặc một vòng điều khiển.

Phân biệt

- Dự phòng lạnh
- Dự phòng nóng: Dự phòng cạnh tranh và dự phòng dự trữ

Các biện pháp dự phòng nóng

- Dự phòng CPU: Mỗi trạm điều khiển cần có CPU dự phòng cạnh tranh, thực hiện song song và đồng bộ với CPU chính và so sánh kết quả
- Dự phòng trạm điều khiển: Dự phòng dự trữ 1:1, chuyển mạch kịp thời, trơn tru
- Dự phòng dự trữ hệ thống mạng: Dự phòng cáp truyền, dự phòng module truyền thông và các thiết bị mạng khác, chuyển mạch kịp thời, trơn tru, thời gian chuyển mạch < 1ms
- Dự phòng vào/ra
- Dự phòng trạm vận hành 1:n
- Dự phòng trạm server 1:1

Dự phòng lạnh:

- Cho phép thay thế trực tuyến các module vào/ra và các card khác
- Cho phép thay thế các trạm điều khiển trong một thời gian nhanh nhất

12.3 Cơ chế an toàn

- Có cơ chế dừng an toàn, dừng khẩn cấp (mạch cứng hoặc qua bus an toàn) khi hệ thống có các cơ cấu chuyển động
- Tín hiệu ra tương tự hỗ trợ chế độ an toàn khi mất liên lạc với trạm điều khiển hoặc khi phát hiện trạm điều khiển có lỗi (giữ giá trị cuối hoặc đưa về giá trị mặc định)

12.4 Cơ chế khởi động lại sau sự cố

- Các trạm điều khiển cần có khả năng tự phát hiện lỗi mất nguồn, thực hiện xử lý và đặt các tín hiệu ra về trạng thái an toàn, sau khi có nguồn trở lại phải có khả năng hồi phục trạng thái cũ
- Các trạm vận hành phải có khả năng tự hồi phục trạng thái làm việc trước khi xảy ra sự cố
- Tất cả các nút mạng phải có khả năng tự khởi động một cách độc lập với các nút khác

12.5 Bảo mật

- Đặt chế độ bảo mật theo trạm hoặc theo người sử dụng để hạn chế, kiểm soát quyền truy nhập dữ liệu và điều khiển
- Đặt chế độ bảo mật dựa trên từng tag riêng rẽ hoặc từng cửa sổ riêng rẽ
- Người vận hành cần sử dụng mã ID và mật khẩu
- Cho phép thực hiện bảo mật theo nhóm

12.6 Bảo trì

- Chế độ bảo trì: Hệ thống cần cho phép người vận hành đưa trực tiếp giá trị biến quá trình trong trường hợp thiết bị trường hỏng, cần sửa chữa hoặc đang được hiệu chỉnh
- Chỉ thị lỗi: mỗi module, mỗi card hoặc bộ nguồn cần được trang bị đèn LED hoặc đèn khác để chỉ thị trạng thái vận hành
- Chẩn đoán: Hệ thống phải hỗ trợ chẩn đoán trực tuyến với các yêu cầu tối thiểu như:
 - CPU của trạm điều khiển cần có biện pháp phát hiện và sửa lỗi trong bộ nhớ
 - Mạng truyền thông thời gian thực cần sử dụng biện pháp bảo toàn dữ liệu để phát hiện lỗi
 - Thông báo lỗi và các thông tin chẩn đoán với người vận hành về cấp nguồn, quạt thông gió/làm mát, các card DCS, máy in, ROM của trạm điều khiển, lỗi thực hiện thuật toán điều khiển, lỗi nạp chương trình lên/xuống,...
- System back-up: Toàn bộ hệ thống phần mềm cần được lưu trữ backup trên các phương tiện phổ thông, tất cả các phần mềm phát triển, chương

trình ứng dụng và các gói phần mềm chuẩn cũng như tùy chọn phải được cung cấp cùng đĩa cứng hoặc đĩa CD.

- Các công cụ đặc biệt do nhà sản xuất cung cấp, phục vụ chẩn đoán và thay thế thiết bị
-

13 ĐÁNH GIÁ VÀ LỰA CHỌN GIẢI PHÁP ĐIỀU KHIỂN PHÂN TÁN

13.1 Đánh giá và lựa chọn các sản phẩm DCS tích hợp trọn vẹn

13.1.1 Phạm vi chức năng

Chức năng điều khiển cơ sở

- Phương pháp điều khiển vòng kín (PID, MPC, Fuzzy) với các yêu cầu công nghiệp như chuyển chế độ Manual/Automatic trơn tru, Anti-Reset-Windup.
- Điều khiển logic, khóa liên động

Chức năng điều khiển cao cấp

- Điều khiển mẻ, điều khiển công thức
- Điều khiển thích nghi, bền vững, tối ưu
- Điều khiển chuyên gia

Chức năng điều khiển giám sát

- Chất lượng giao diện đồ họa
- Khả năng lập báo cáo tự động
- Cơ chế quản lý và xử lý sự kiện, sự cố
- Hỗ trợ ActiveX-Control và OPC
- Hỗ trợ giao diện cơ sở dữ liệu ODBC
- Chức năng Web

13.1.2 Cấu trúc hệ thống và các thiết bị thành phần

- Cấu trúc vào/ra phân tán hay vào/ra tập trung
- Cấu trúc cấp điều khiển
- Cấu trúc cấp điều khiển giám sát
- Các chủng loại thiết bị hỗ trợ
- Các hệ thống mạng truyền thông được hỗ trợ (đặc biệt bus trường liên quan tới các chủng loại thiết bị trường có thể hỗ trợ).

13.1.3 Tính năng mở

- Khả năng tự mở rộng hệ thống
- Lựa chọn các thiết bị của các nhà cung cấp khác
- Hỗ trợ các chuẩn công nghiệp (COM, OPC, ActiveX-Control, MMS, IEC,...)

13.1.4 Phát triển hệ thống

Cấu hình hệ thống

- Đơn giản, hướng đối tượng
- Khả năng phát triển hệ thống một cách xuyên suốt

- Cấu hình và tham số hóa các thiết bị và mạng truyền thông dễ dàng qua phần mềm từ trạm kỹ thuật

Lập trình điều khiển

- Đơn giản, hướng đối tượng
- Các ngôn ngữ lập trình chuyên dụng (FBD, SFC, ST,...)
- Các ngôn ngữ lập trình bậc cao (C/C++, BASIC)
- Lập trình giao tiếp ngầm hay hiện
- Khả năng tự mở rộng thư viện chức năng (thông qua một ngôn ngữ lập trình bậc cao)

13.1.5 Độ tin cậy và tính sẵn sàng

- Cơ chế dự phòng
- Khả năng bảo mật
- ...

13.1.6 Giá thành, chi phí

Chi phí ban đầu

- Chi phí thiết kế hệ thống
- Chi phí phần cứng
- Chi phí phần mềm công cụ
- Chi phí phát triển phần mềm ứng dụng
- Chi phí triển khai, đưa vào vận hành
- Chi phí đào tạo, chuyển giao công nghệ
- ...

Chi phí vận hành

- Chi phí bảo trì, bảo dưỡng thiết bị và phần mềm
- Chi phí thiết bị thay thế
- Chi phí dịch vụ hỗ trợ kỹ thuật
- Chi phí dừng hệ thống khi xảy ra sự cố
- ...

13.2 So sánh giải pháp DCS tích hợp trọn vẹn với các giải pháp khác

So sánh trên cơ sở các tiêu chí:

- Phạm vi chức năng
- Độ tin cậy và tính sẵn sàng
- Tính năng mở
- Phát triển hệ thống
- Giá thành, chi phí

So sánh với giải pháp PLC+SCADA/HMI

Tham khảo [11]

So sánh với giải pháp PC + SCADA/HMI

Tham khảo [10]

14 GIỚI THIỆU MỘT SỐ HỆ ĐIỀU KHIỂN PHÂN TÁN TIÊU BIỂU

14.1 PCS7 của Siemens

14.2 PlantScape của Honeywell

14.3 DeltaV của Fisher Rosermount

14.4 Centum CS1000/CS3000 của Yokogawa

14.5 AdvantOCS của ABB

Tham khảo các tài liệu đi kèm đĩa CD.

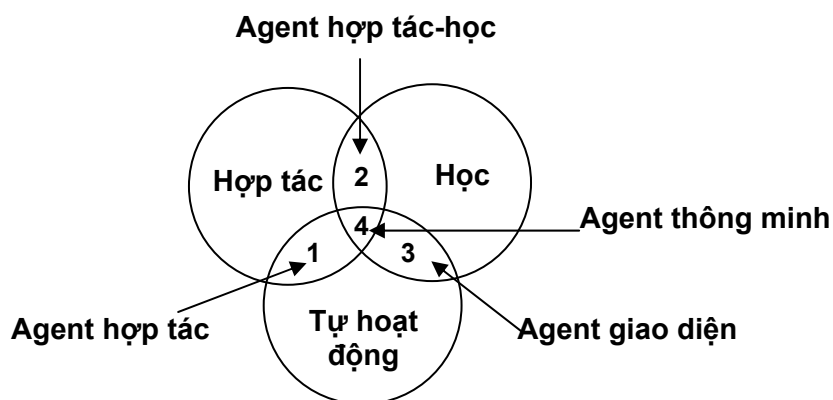
15 MỘT SỐ HƯỚNG NGHIÊN CỨU VÀ ỨNG DỤNG

15.1 Trí tuệ nhân tạo phân tán

Đối với các hệ thống mang đặc thù tính phân tán, việc ứng dụng *trí tuệ nhân tạo phân tán* là một trong những hướng nghiên cứu mới, hứa hẹn nhiều kết quả khả quan. Gần đây, agent (tác tử) và multi-agent (đa tác tử) được coi là các công nghệ trọng tâm của trí tuệ nhân tạo phân tán, thu hút được sự quan tâm của đông đảo giới nghiên cứu trong lĩnh vực công nghệ thông tin.

Agent là một thực thể phần mềm thông minh, có khả năng tự hoạt động với nhiệm vụ xác định để đạt được các mục tiêu đã đề ra. Đáng lưu ý là trong thực tế không có định nghĩa nào cho khái niệm agent được chấp nhận một cách thống nhất. Hầu như người ta chỉ có thể nhất trí rằng *tự hoạt động (autonomy)* là trọng tâm trong khái niệm agent. Có thể nói, chính vì đứng trên các quan điểm ứng dụng khác nhau nên mỗi nhà nghiên cứu tìm cách đưa ra một định nghĩa thích hợp nhất với lĩnh vực ứng dụng cụ thể. Do đó, việc phân loại agent trước định nghĩa có lẽ hợp lý hơn quá trình ngược lại.

Trong thực tế cũng có nhiều quan điểm phân loại agent khác nhau. Ví dụ, một số tác giả phân biệt agent thông minh, agent di động với agent thông thường. Trên Hình 15-1 là mô hình phân loại theo Nwana, được chấp nhận tương đối rộng rãi.



Hình 15-1: Phân loại agent theo Nwana

Các agent có thể thiết kế theo mô hình single-agent hoặc multi-agent. Các single-agent không nhận biết các agent khác để cùng tương tác, mà nó chỉ coi các agent đó là một phần của môi trường xung quanh. Một hệ thống multi-agent có thể phân chia một nhiệm vụ để nhiều agent cùng phối hợp giải quyết. Điều này rất có ý nghĩa trong một hệ thống điều khiển phân tán.

Theo nhiều quan điểm, agent có những đặc điểm chính như sau:

- Thông minh và có khả năng hoạt động độc lập (*Autonomy and Intelligence*): các agent tự kiểm soát và chịu trách nhiệm về những quyết

định và hành vi của mình, tự hoạt động mà không cần đợi những tác động từ ngoài vào.

- Có khả năng học (*Learning*): tồn tại trong một môi trường động, các agent phải có khả năng học để có thể thích nghi và giải quyết những vấn đề nảy sinh.
- Có khả năng giao tiếp (*Communication*): giao tiếp giữa các agent, và giữa agent với con người.
- Có khả năng phối hợp hoạt động (*Co-operation*): với các agent khác và với con người nhằm giải quyết những vấn đề phức tạp mà một agent không thể thực hiện.
- Có khả năng di chuyển (*Mobility*): một agent có thể di chuyển qua hệ thống, từ khu vực này đến khu vực khác nhằm thu thập dữ liệu.

Tuy nhiên, trong các đặc tính trên, chỉ đặc tính thứ nhất và thứ hai được coi là cốt lõi, còn các đặc tính khác chỉ là tiêu biểu trong các ứng dụng thực tế. Ví dụ, khả năng giao tiếp và phối hợp hoạt động là hai đặc tính trong một hệ multi-agent. Hay khả năng di chuyển là một đặc tính tiêu biểu trong các ứng dụng Internet, tuy không thực sự cần thiết trong nhiều hệ thống ứng dụng khác.

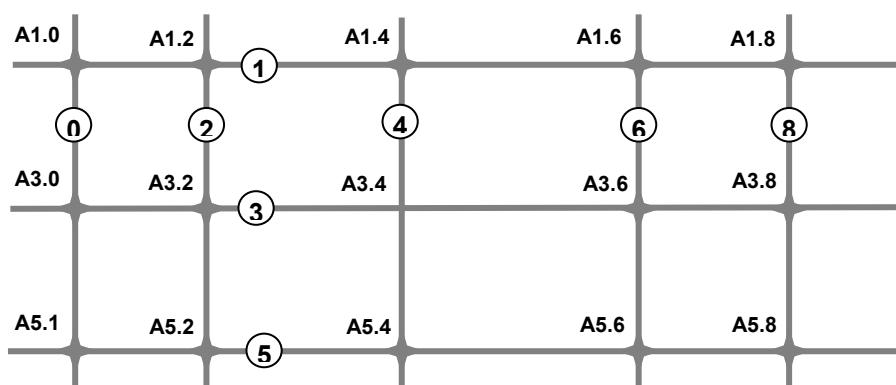
15.2 Điều khiển và giám sát các hệ thống giao thông

15.2.1 Đặt vấn đề

Hệ thống giao thông nói chung và hệ thống điều khiển tín hiệu giao thông đô thị nói riêng là những hệ phân tán tiêu biểu, việc áp dụng phương pháp điều khiển cục bộ hay tập trung đều không thích hợp. Độ phức tạp của cấu trúc phân tán ở đây không những thể hiện qua phân bố địa lý trên phạm vi rộng, mà còn qua sự phân tán chức năng và tính bất định, dễ thay đổi của mô hình. Dựa trên ý tưởng trí tuệ phân tán, người ta có thể xây dựng một mô hình kiến trúc tổng thể cho điều khiển thông minh hệ thống đèn tín hiệu giao thông.

15.2.2 Mô hình hệ thống điều khiển đèn tín hiệu giao thông bằng công nghệ Agent

Hệ thống điều khiển đèn tín hiệu giao thông đô thị đưa ra ở đây có cấu trúc phân tán hoàn toàn, được thực hiện dưới dạng một hệ multi-agent, trong đó việc điều khiển tại mỗi nút giao thông do một agent đảm nhiệm. Thực chất, mỗi agent ở đây là một bộ điều khiển thích nghi, có khả năng nhận biết tình hình giao thông thực tế để đưa ra quyết định điều khiển một cách thông minh. Ví dụ, một agent có thể dựa vào lượng giao thông thực tế tại các làn đường mà đưa ra quyết định về thời gian mở đèn xanh, sử dụng lý thuyết logic mờ.



Hình 15-2: Mô hình hệ thống điều khiển tín hiệu giao thông

Mô hình hệ thống được minh họa đơn giản hóa trên Hình 15-2. Các đường Đông-Tây được đánh số lẻ và các đường Bắc-Nam được đánh số chẵn. Các agent điều khiển nút (Ax.y) thuộc một tuyến đường được nối mạng với nhau thành một nhóm. Như vậy, một agent điều khiển nút thông thường thuộc hai nhóm khác nhau ứng với hai tuyến đường. Trong điều kiện thông thường, giữa các agent lân cận có sự giao tiếp và phối hợp hoạt động để đạt được mục tiêu đề ra là tối ưu khả năng lưu thông trên một tuyến đường. Ví dụ, A3.4 có thể hợp tác với A3.2, A3.6, A1.4 và A5.4.

Trường hợp có sự cố trong giao tiếp xảy ra (ví dụ do sự cố mạng), mỗi agent phải có khả năng chuyển từ chế độ hợp tác sang chế độ hoạt động hoàn toàn độc lập. Khi đó, mỗi agent không có thông tin hỗ trợ từ các agent khác, mà phải tự nhận biết tình huống để phán đoán và khai thác thông tin. Điều khiển làn sóng xanh là một ví dụ tiêu biểu. Trong trường hợp bình thường, các agent trên cùng một tuyến đường một chiều có thể trao đổi thông tin về thời điểm mở đèn xanh để tạo ra khả năng lưu thông tốt nhất. Tuy nhiên, khi có sự cố xảy ra về mặt giao tiếp, mỗi agent sẽ phải tự nhận biết mẫu lưu lượng giao thông thông qua các thiết bị đo để ra quyết định phối hợp mở đèn xanh.

Có thể thấy rằng, việc tối ưu hóa toàn cục cho hệ thống bằng phương pháp tính cũng như phương pháp động nhưng tập trung là một bài toán không thể giải được đối với một hệ phân tán có cấu trúc và tham số thay đổi. Ngược lại, việc tối ưu hóa cục bộ cho từng nút giao thông không thể mang lại hiệu quả cao nhất cho toàn hệ thống. Giống như trong một nền kinh tế thị trường, vấn đề trọng tâm ở đây là khả năng tự học, tự thích nghi và hợp tác giữa các agent điều khiển nút để có thể cùng nhau đạt được mục tiêu chung một cách tốt nhất cho cả hệ thống.

Điều khiển thông minh mang đến khả năng linh hoạt rất lớn cho hệ thống đèn tín hiệu và đem lại sự thuận tiện tối ưu cho hệ thống giao thông. Ứng dụng trí tuệ phân tán và công nghệ agent, multi-agent vào trong các hệ thống

điều khiển phân tán nói chung và các hệ thống giao thông nói riêng là một hướng nghiên cứu có nhiều triển vọng.

15.3 Điều khiển và giám sát các hệ thống sản xuất và cung cấp điện

Hệ thống sản xuất và cung cấp điện cũng là một ví dụ điển hình một hệ thống phân tán qui mô lớn, do đó việc áp dụng các phương pháp điều khiển phân tán sẽ mang lại hiệu quả tốt.

Một số vấn đề lớn được đặt ra:

- Điều khiển cục bộ từng nhà máy điện, từng khu vực và điều khiển phối hợp trong một hệ thống điện lưới quốc gia
- Tối ưu hóa cục bộ và tối ưu hóa toàn cục (chất lượng và hiệu quả kinh tế)
- Truyền thông đường dài
- Tính ngẫu nhiên, tính bất định, tính hỗn hợp của hệ thống.

Một số hướng giải pháp là:

- Sử dụng công nghệ tác tử và đa tác tử cho điều khiển cục bộ và phối hợp hoạt động trong toàn hệ thống
- Công nghệ truyền thông qua đường dây tải điện
- Công nghệ Web cho chức năng điều khiển giám sát.
- ...

TÀI LIỆU THAM KHẢO

- [1] Martin Fowler, Kendall Scott: *UML Distilled: A Brief Guide to the Standard Object Modeling Language* (2nd Edition). Addison-Wesley, 1999.
- [2] James Rumbaugh, Iva Jacobson, Grady Booch: *The unified modeling language reference manual*. Addison Wesley 1999. (*)
- [3] Erich Gamma et. al.: *Design Patterns – Elements of Reuseable Object-Oriented Software*. Addison- Wesley, 1995.
- [4] OMG: *CORBA – Specification*. www.omg.org (*)
- [5] Microsoft Corp.: “DCOM- Technical Overview”. *White Paper*. MSDN-Library. (*)
- [6] Microsoft Corp.: “Dr. GUI on Components, COM, and ATL”. *MSDN Selected Online Column*. (*)
- [7] OPC Foundation: *OPC – Data Access Custom Interfaces Specification 2.0*. www.opcfoundation.org. (*)
- [8] OPC Foundation: *OPC – Data Access Automation Interfaces Specification 2.0*. www.opcfoundation.org (*)
- [9] Đặng Anh Việt, Bùi Quang Việt: *Xây dựng phần mềm khung cho giải pháp điều khiển trên nền PC*. Đồ án tốt nghiệp, BM Điều khiển Tự động, Đại học Bách khoa Hà Nội, 5/2002.
- [10] Tạp chí *Tự động hóa ngày nay*.
- [11] Chuẩn IEC-61131-3 và IEC-61499: www.holobloc.com (*)
- [12] Trang Web về IEC-61131-3: www.plcopen.org
- [13] Các tài liệu sản phẩm DCS của một số hãng (*)

(*): Có trong đĩa CD tài liệu kèm theo bài giảng.