Hệ thống điều khiển phân tán

Chương 1: Mở đầu

8/17/2006

Chương 1: Mở đầu

1.1 Mục đích và phạm vi đề cập của môn học

Các nội dung cơ bản của bài giảng

1.2 Phương pháp học và đánh giá kết quả

Các nguồn tài liệu tham khảo

1.3 Giới thiệu các khái niệm cơ bản

1.4 Tổng quan các lĩnh vực ứng dụng

1.5 Lược sử phát triển các giải pháp điều khiển

Sự tiến hóa tới các hệ thống điều khiển phân tán

1.1 Mục đích và phạm vi đề cập

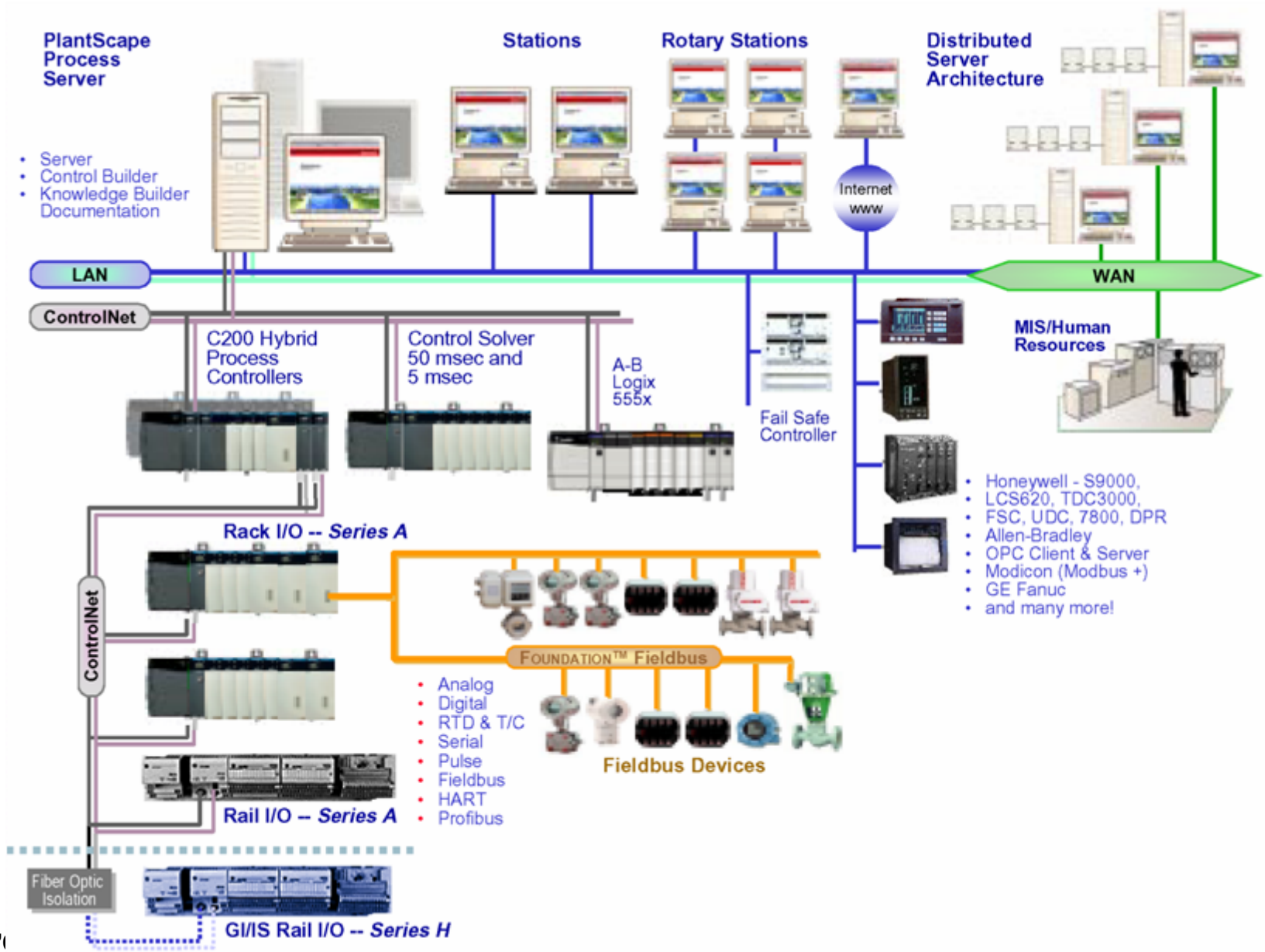
Phạm vi đề cập

- Các hệ thống điều khiển hiện đại có kiến trúc phân tán trong công nghiệp cũng như trong nhiều lĩnh vực khác

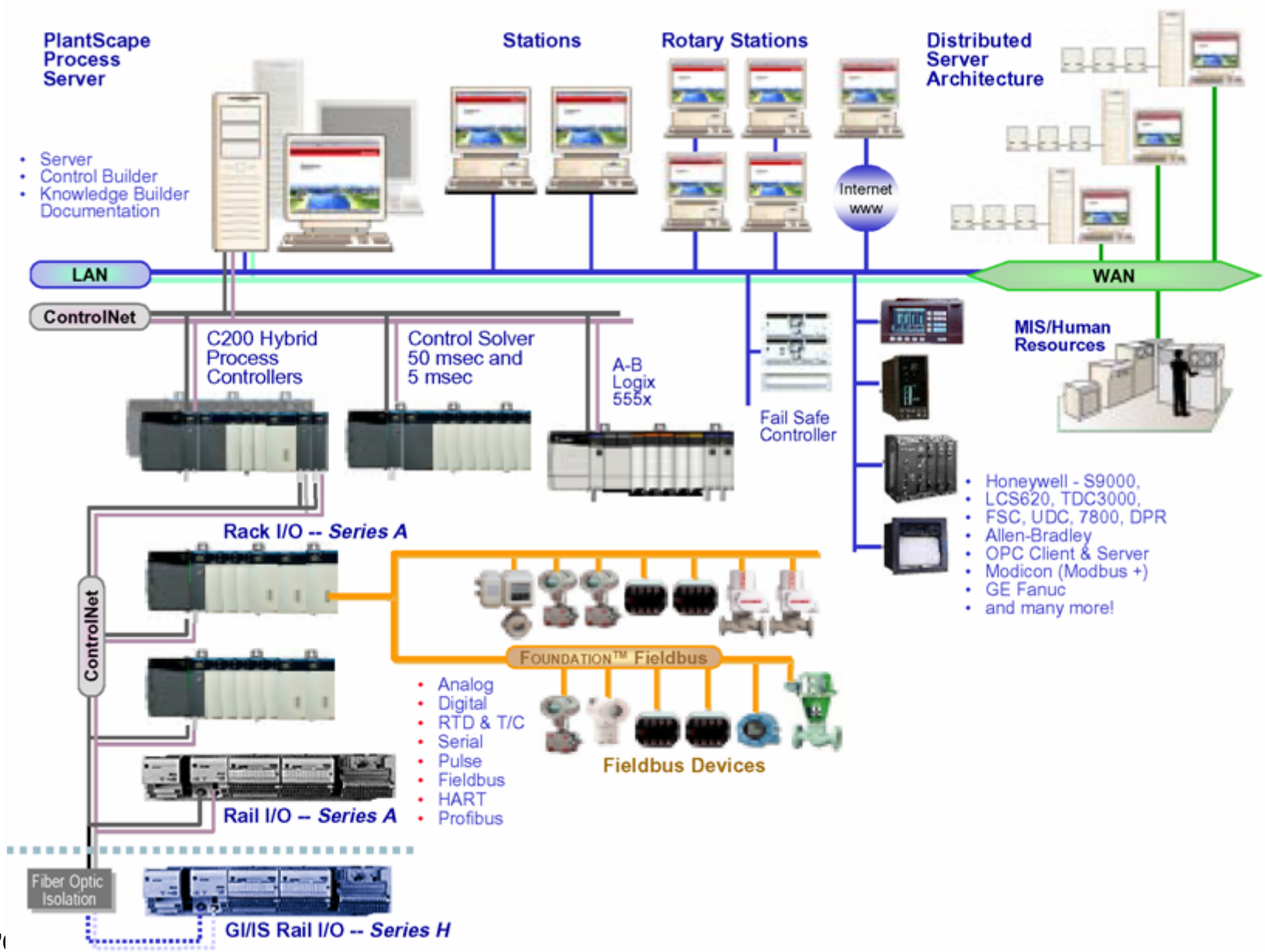
Mục đích: Sinh viên nắm được

- Các khái niệm cơ bản và tổng quan về các HTĐK&GS hiện đại trong công nghiệp
- "state of the art" trong công nghệ HTĐK, kiến trúc giải pháp ĐK, truyền thông CN, công nghệ PM
- Sơ lược về các nhiệm vụ phát triển, các nguyên tắc cơ bản trong ***thiết kế giải pháp hệ thống điều khiển công nghiệp***

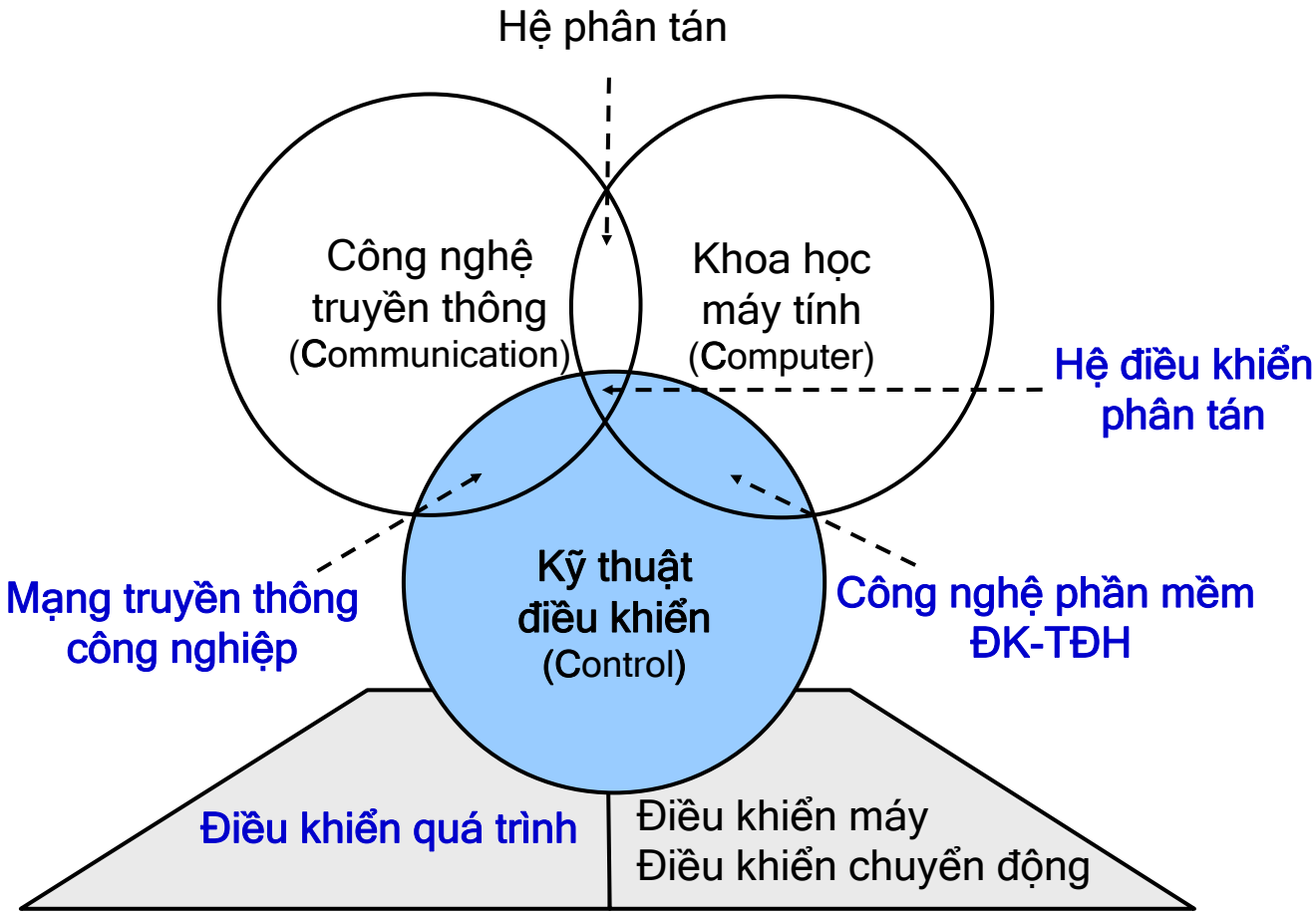
Ví dụ: PlantScape (Honeywell)



Ví dụ: PlantScape (Honeywell)



Tính chất liên ngành của môn học



MÔ HÌNH 3C+

(Control + Computer + Communication + ...)

Nội dung cơ bản

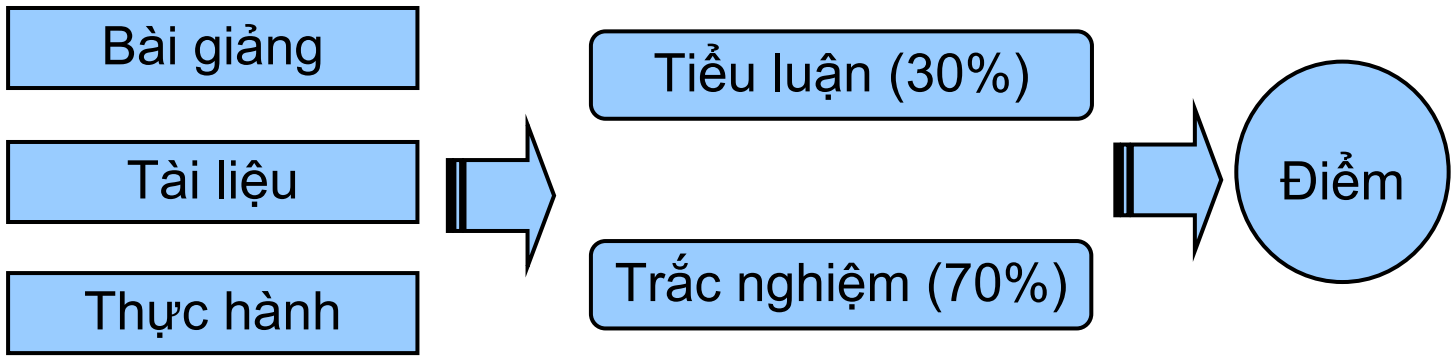
- Cấu trúc các hệ thống điều khiển và giám sát: Mô hình phân cấp, các thành phần chức năng cơ bản
- Các kiến trúc và giải pháp hệ thống điều khiển phân tán (DCS, PLC-based DCS, PC-based DCS, FCS)
- Cơ sở lý luận của điều khiển phân tán
- Cơ sở công nghệ phần mềm: Xử lý phân tán, công nghệ hướng đối tượng, phần mềm thành phần
- Hệ thống điều khiển giám sát và thu thập dữ liệu (SCADA)
- Các chuẩn giao tiếp công nghiệp: MMS, OPC, XML,...
- Độ tin cậy và tính sẵn sàng của hệ thống
- Các hướng nghiên cứu và ứng dụng

Phân bố chương trình

- 14 bài giảng
- 2 hai buổi thực hành: Lập trình phân tán với mô hình COM/DCOM
  - Lập trình COM-Server sử dụng Visual C++
  - Lập trình COM-Client (HMI) sử dụng Visual Basic
- Tiểu luận (nhóm 2/4 người):
  - bài viết 15-20 trang
  - trình bày 15 phút
  - thảo luận 5-10 phút

1.2 Phương pháp học và đánh giá

- Nghe, đọc, hỏi, thảo luận, trình bày
- Thực hành và chủ động liên hệ thực tế
- Đánh giá kết hợp trình bày tiểu luận và thi trắc nghiệm (không sử dụng tài liệu)

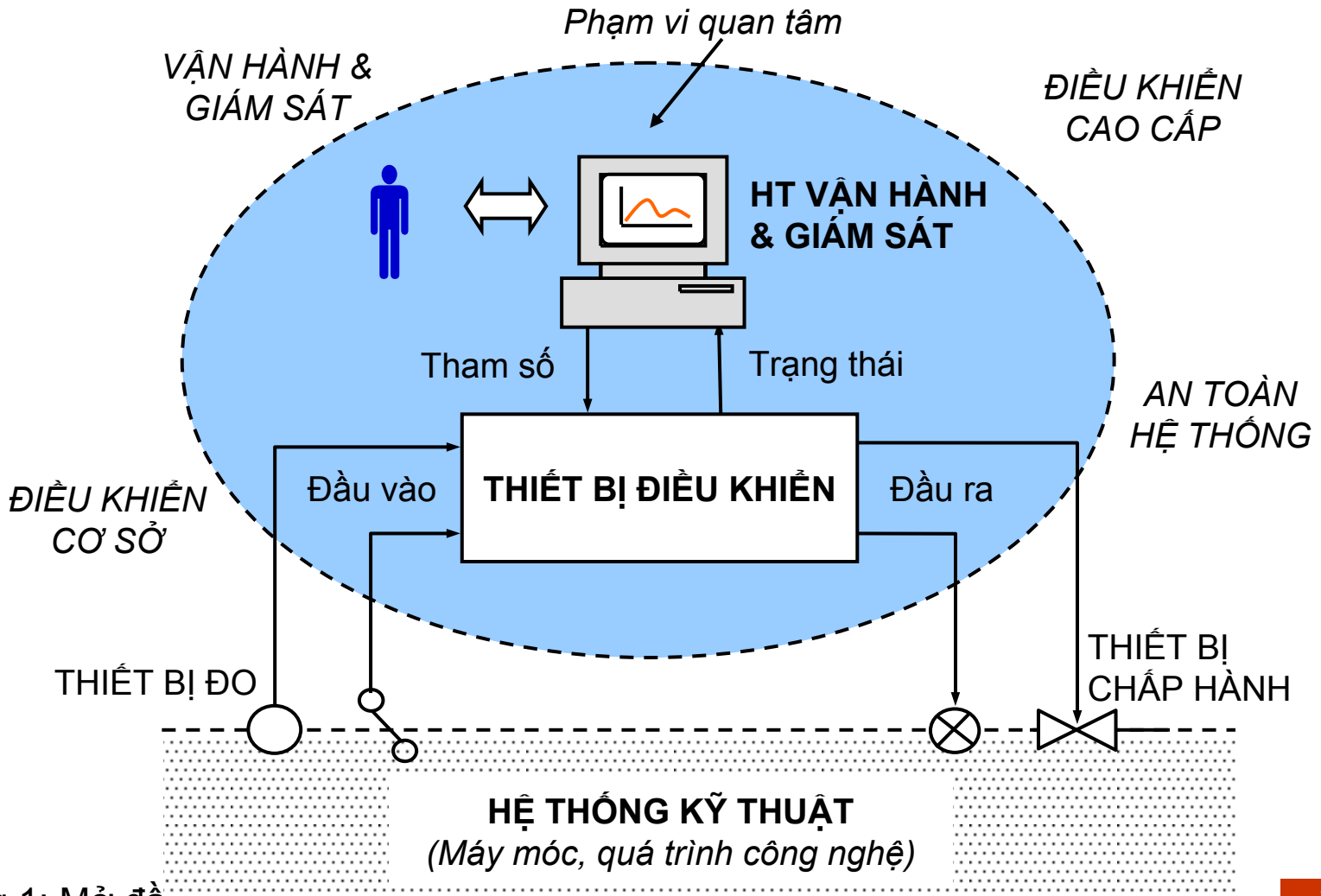


Tài liệu tham khảo

- Đĩa CD tập hợp tài liệu từ nhiều nguồn khác nhau
  - Bài giảng (2003, chưa cập nhật)
  - Tài liệu sản phẩm, giới thiệu sản phẩm của nhiều hãng
  - Một số chuẩn quốc tế quan trọng
- Tạp chí Tự động hóa ngày nay, chuyên san đặc biệt 2004/2006 (Tự động hóa - Đolường - Điều khiển)
- Các trang Web: www.controleng.com, www.automationtechies.com, www.abb.com, www.ad.siemens.de, www.easydeltav.com, ...

1.3 Một số khái niệm cơ bản

- Hệ thống điều khiển và giám sát (HTĐK&GS)



Các chức năng hệ thống

- Điều khiển cơ sở:
  - Điều chỉnh tự động (*regulatory control*)
  - Điều khiển servo, ĐK bám (*servo mechanism, tracking problem*)
  - Điều khiển rời rạc (*discrete control*)
  - Điều khiển trình tự (*sequence control*)
- Điều khiển vận hành & giám sát:
  - Giao diện người máy (*Human-Machine-Interface*, HMI)
  - Quản lý dữ liệu quá trình (*process data management*)
  - Lập báo cáo tự động (*reporting*)
- Điều khiển cao cấp:
  - Điều khiển mẻ (*batch control*)
  - Điều khiển chất lượng (*quality control*)
  - Điều khiển tối ưu, tối ưu hóa quá trình (*process optimisation*)
- An toàn hệ thống:
  - Khóa liên động,
  - Cảnh giới, báo động

Hệ điều khiển phân tán

- HTĐK&GS tích hợp toàn diện có kiến trúc phân tán
- Công nghệ điều khiển số hoàn toàn
- Phân tán về cấu trúc hệ thống và phân tán về chức năng điều khiển
- Các thành phần được tích hợp chặt chẽ qua mạng truyền công nghiệp và các giao thức chuẩn
- Hầu như toàn bộ các hệ thống ĐK&GS hiện đại trong các ngành công nghiệp chế biến là các hệ điều khiển phân tán
- Các HĐKPT cũng ngày càng có vai trò quan trọng, chủ chốt trong các lĩnh vực khác

Hệ ĐKPT <> hệ DCS?

- DCS (Distributed Control System): Một kiến trúc giải pháp điều khiển phân tán (phân biệt với PLC, PC,...)
- Một hệ DCS là một hệ điều khiển phân tán
- Một hệ ĐKPT không bắt buộc phải là một hệ DCS (lưu ý cách viết tắt đã trở thành một tên riêng giống như PLC)
- Khái niệm DCS không còn mới, nhiều hệ điều khiển hiện đại có kiến trúc DCS không được gọi với cái tên "DCS"

Hệ ĐKPT <> Hệ SCADA <> DCS?

- SCADA (*Supervisory Control And Data Acquisition*): *Hệ thống* hoặc *chức năng* Thu thập dữ liệu & Điều khiển giám sát
- Một hệ SCADA thường là một hệ ĐKPT (nghĩa rộng)
- Một hệ DCS có chức năng SCADA
- Một hệ SCADA không có chức năng điều khiển cơ sở => không bao giờ được gọi một hệ DCS, mặc dù nó có thể được xây dựng trên cơ sở các thành phần của một hệ DCS

1.4 Tổng quan các lĩnh vực ứng dụng

- Nghiên cứu đặc điểm các lĩnh vực ứng dụng
- Làm rõ các bài toán điều khiển tiêu biểu
- Tìm ra giải pháp hệ thống điều khiển phù hợp

Tự động hóa quá trình (process automation)

- Phạm vi ứng dụng quan trọng nhất của các HĐKPT
- Công nghiệp chế biến, khai thác, năng lượng (gọi chung là công nghiệp chế biến, process industry): hóa chất, dầu khí, thực phẩm, dược phẩm, sản xuất điện năng,...
- Qui mô lớn, đầu tư chi phí cao, thị trường sản phẩm lớn
- Các quá trình vận hành liên tục hoặc theo mẻ, các biến quá trình có giá trị tương tự
- Yêu cầu cao về độ tin cậy, an toàn hệ thống, chất lượng sản phẩm, hiệu quả sản xuất và bảo vệ môi trường
- Vai trò rất quan trọng của điều khiển quá trình (process control): điều chỉnh, hiển thị, giám sát, ghi chép, lưu trữ,...
- Thị trường lớn nhất của các sản phẩm tự động hóa (DCS, PLC, PC, HMI, SCADA,...)
- Yêu cầu năng lực rất cao của các công ty tích hợp hệ thống

Tự động hóa xí nghiệp (factory automation)

- Công nghiệp chế tạo, lắp ráp (manufactoring): xe hơi, điện tử, máy công cụ, nhựa, đóng bao,...
- Qui mô sản xuất vừa và nhỏ
- Các quá trình rời rạc, vận hành gián đoạn, các quá trình diễn ra rất nhanh (các quá trình cơ điện)
- Yêu cầu cao về tốc độ, độ chính xác, sự linh hoạt, tính tích hợp cao giữa các cấp => CIM (computer integrated manufactoring)
- Vai trò đặc biệt quan trọng của điều khiển rời rạc (discrete control) và điều khiển chuyển động (motion control)
- Các giải pháp điều khiển tiêu biểu: PLC, PC, CNC, Robot
- Các hệ điều khiển phân tán cũng ngày càng được ứng dụng nhiều hơn (trên nền PLC, PC, hệ điều khiển lai,...)

Tự động hóa tòa nhà (building automation)

- Công sở, trung tâm thương mại, khách sạn, nhà ga, sân bay, bệnh viện,...
- Các hệ thống lò sưởi, điều hòa, đóng mở cửa, thang máy, gara, chiếu sáng, cảnh báo cháy, ..
- Phạm vi địa lý tương đối hẹp nhưng mức độ hỗn tạp cao, số lượng thiết bị lớn => ĐKPT là giải pháp lý tưởng
- => **Building Management Systems (BMS)**

ĐK&GS các hệ thống giao thông-vận tải

- Hệ thống đèn tín hiệu giao thông, đèn chiếu sáng đô thị, điều khiển sân bay, không lưu, điều vận bến cảng, nhà ga, điều hành xe buýt, xe lửa, giám sát các trục lộ giao thông
- Qui mô vừa và lớn, phạm vi địa lý rộng, đối tượng hỗn hợp, bản chất phân tán cố hữu
- Ứng dụng HĐKPT trên cơ sở tích hợp các thành phần hỗn hợp (ít khi từ một dòng sản phẩm duy nhất!)

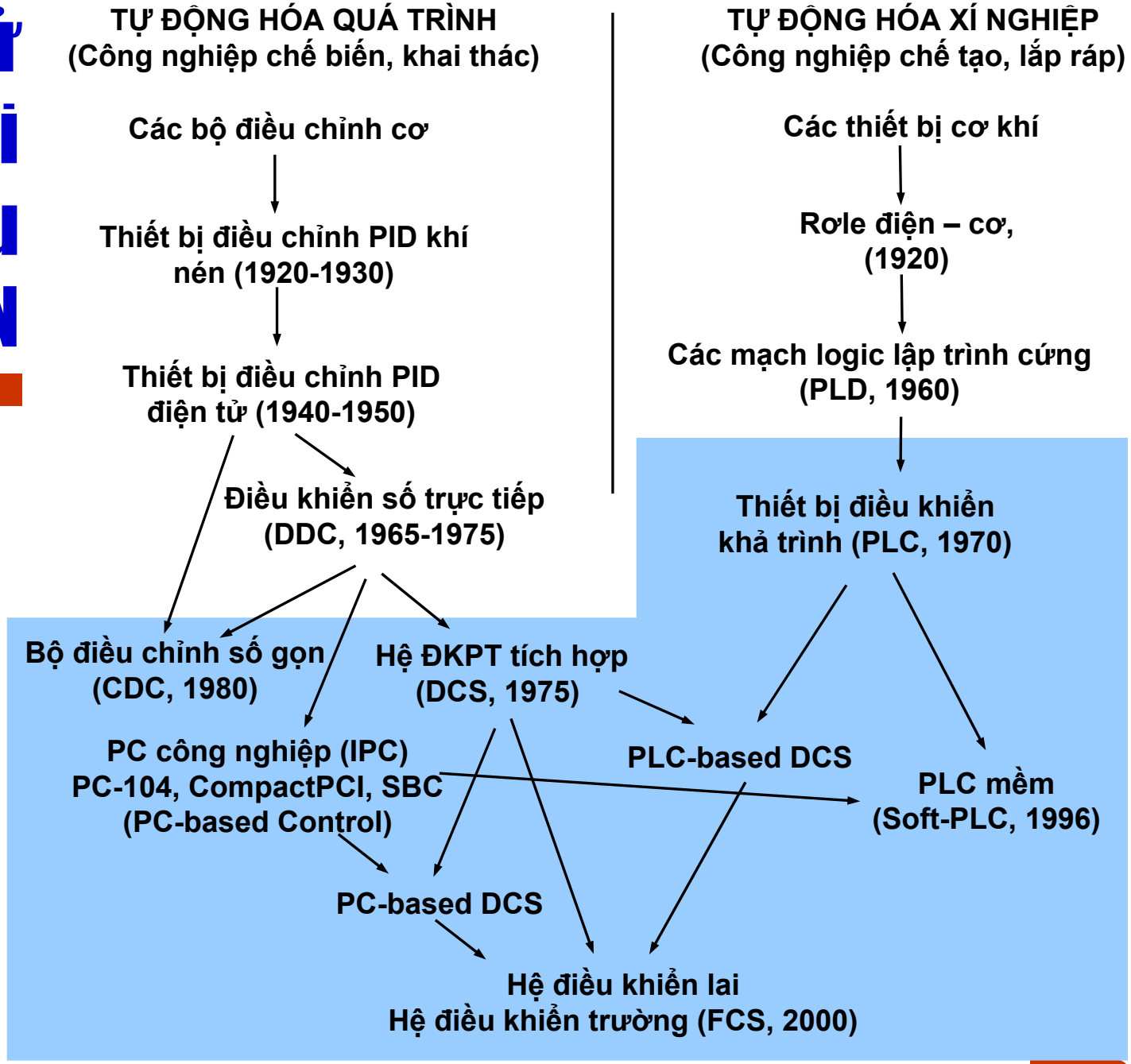
ĐK&GS các hệ thống phân phối

- Hệ thống mạng lưới cung cấp điện, hệ thống đường ống dẫn dầu, khí, hệ thống cung cấp nước sạch
- Qui mô lớn và rất lớn, phạm vi địa lý rất rộng, đối tượng hỗn hợp, bản chất phân tán cố hữu
- HĐKPT phân cấp mạnh, ứng dụng các chuẩn giao tiếp công nghiệp là vấn đề cốt lõi.

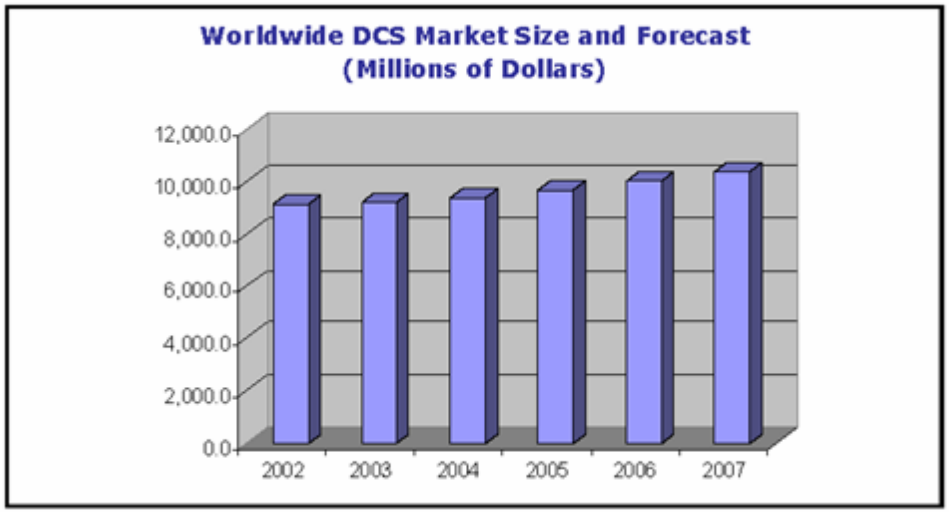
Các lĩnh vực ứng dụng khác

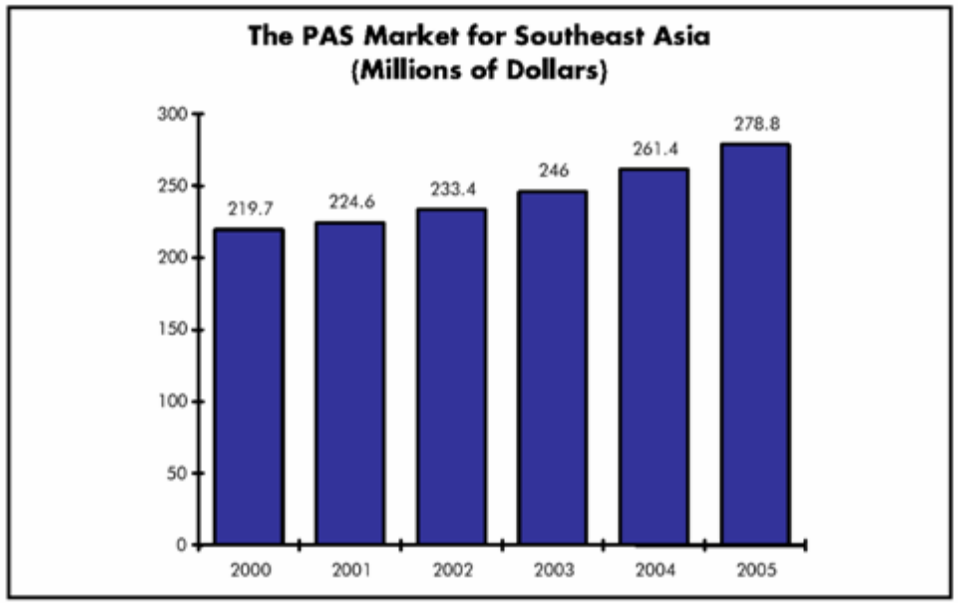
- ĐK&GS các hệ thống viễn thông
- ĐK&GS các hệ thống quốc phòng
- ĐK&GS các hệ thống thủy lợi, môi trường,...

1.5 Lược sử các giải pháp điều khiển CN



Xu hướng thị trường sản phẩm





Tại sao điều khiển phân tán?

Xu hướng tất yếu của sự tiến hóa, tương tự như sự tiến hóa của xã hội loài người:

- Xã hội nguyên thủy: Các bộ lạc riêng lẻ, cục bộ, phân tán (so sánh với thời kỳ điều khiển tương tự)
- Xã hội phong kiến: Nhà nước quân chủ tập trung (so sánh với thời kỳ điều khiển số trực tiếp)
- Xã hội hiện đại: Nền kinh tế thị trường, nhà nước đóng vai trò điều chỉnh, điều phối => kiến trúc phân tán (so sánh với công nghệ điều khiển hiện tại)
- ...

Lý do cơ bản: Công nghệ HT ĐK

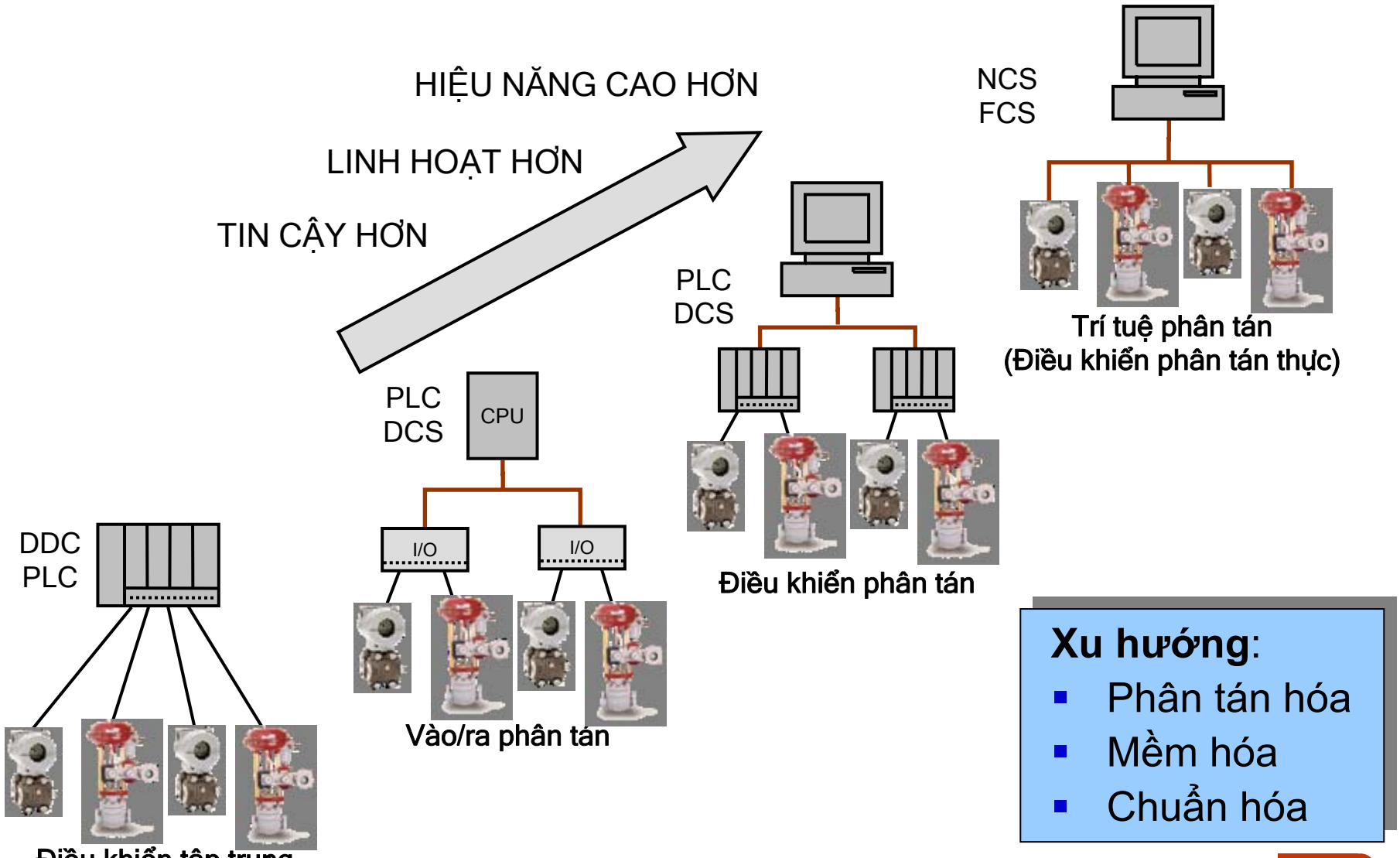
Ưu điểm + khả năng thực hiện + yêu cầu từ người sử dụng:

- Xử lý thông tin, điều khiển tại chỗ => thời gian đáp ứng nhanh hơn, độ tin cậy cao hơn
- Nâng cao hiệu năng hệ thống nhờ xử lý song song và xử lý phân cấp
- Đơn giản hóa các công việc xây dựng và bảo trì hệ thống nhờ cấu trúc module
- Giảm chi phí và thời gian xây dựng hệ thống nhờ quan điểm tích hợp hệ thống dựa trên các công nghệ thành phần

Lý do cơ bản: Công nghệ HT ĐK (tiếp)

- Các tiến bộ (kỹ thuật và giá thành) trong công nghệ vi xử lý cho các thiết bị đo lường, điều khiển và chấp hành
- Các tiến bộ trong công nghệ truyền thông công nghiệp
- Các tiến bộ trong công nghệ phần mềm
- Yêu cầu ngày càng cao về khả năng vận hành, chất lượng sản phẩm, hiệu quả sản xuất, an toàn hệ thống, bảo vệ môi trường trong thị trường cạnh tranh mạnh.

Tiến hóa của các kiến trúc điều khiển



Xu hướng:

- Phân tán hóa
- Mềm hóa
- Chuẩn hóa

Hệ thống điều khiển phân tán

Chương 2: Cấu trúc hệ thống

8/14/2006

Chương 2: Cấu trúc hệ thống

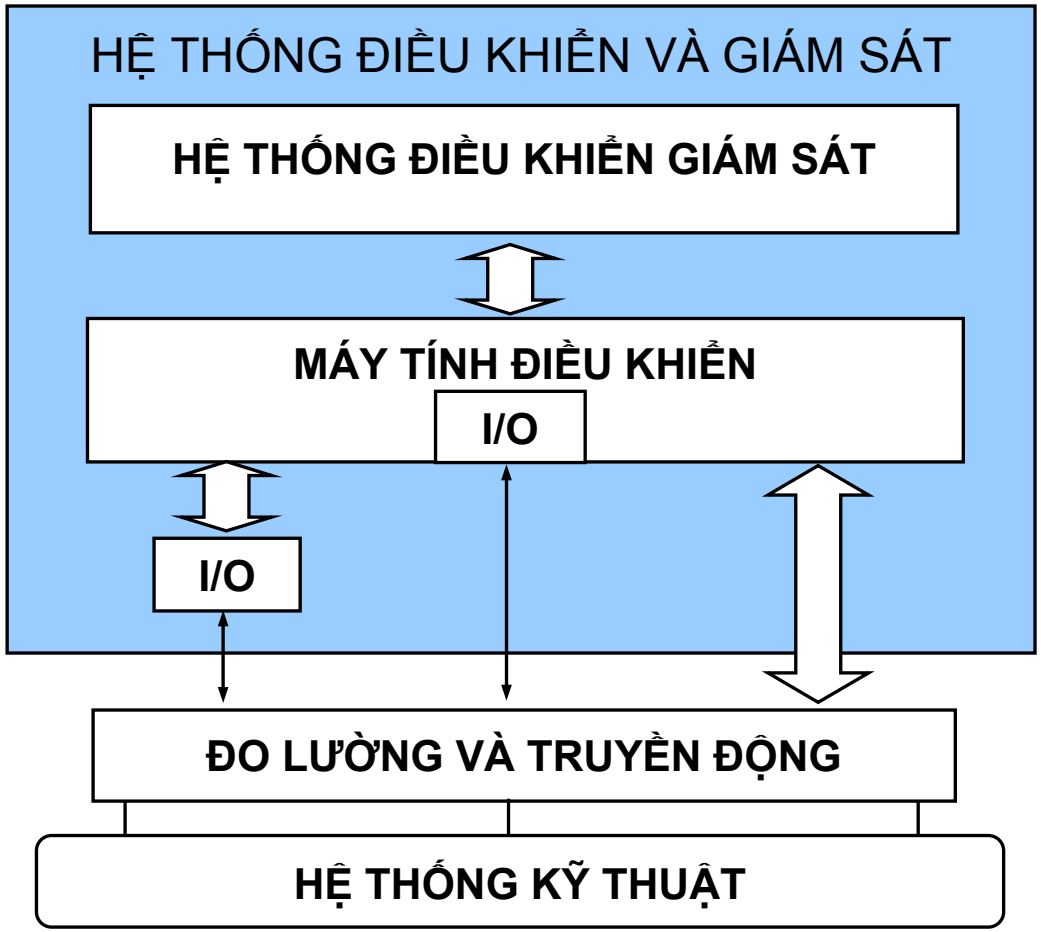
2.1 Cấu trúc cơ bản của một HTĐK&GS

2.2 Mô hình phân cấp chức năng

2.3 Các cấu trúc vào/ra

2.4 Các cấu trúc điều khiển

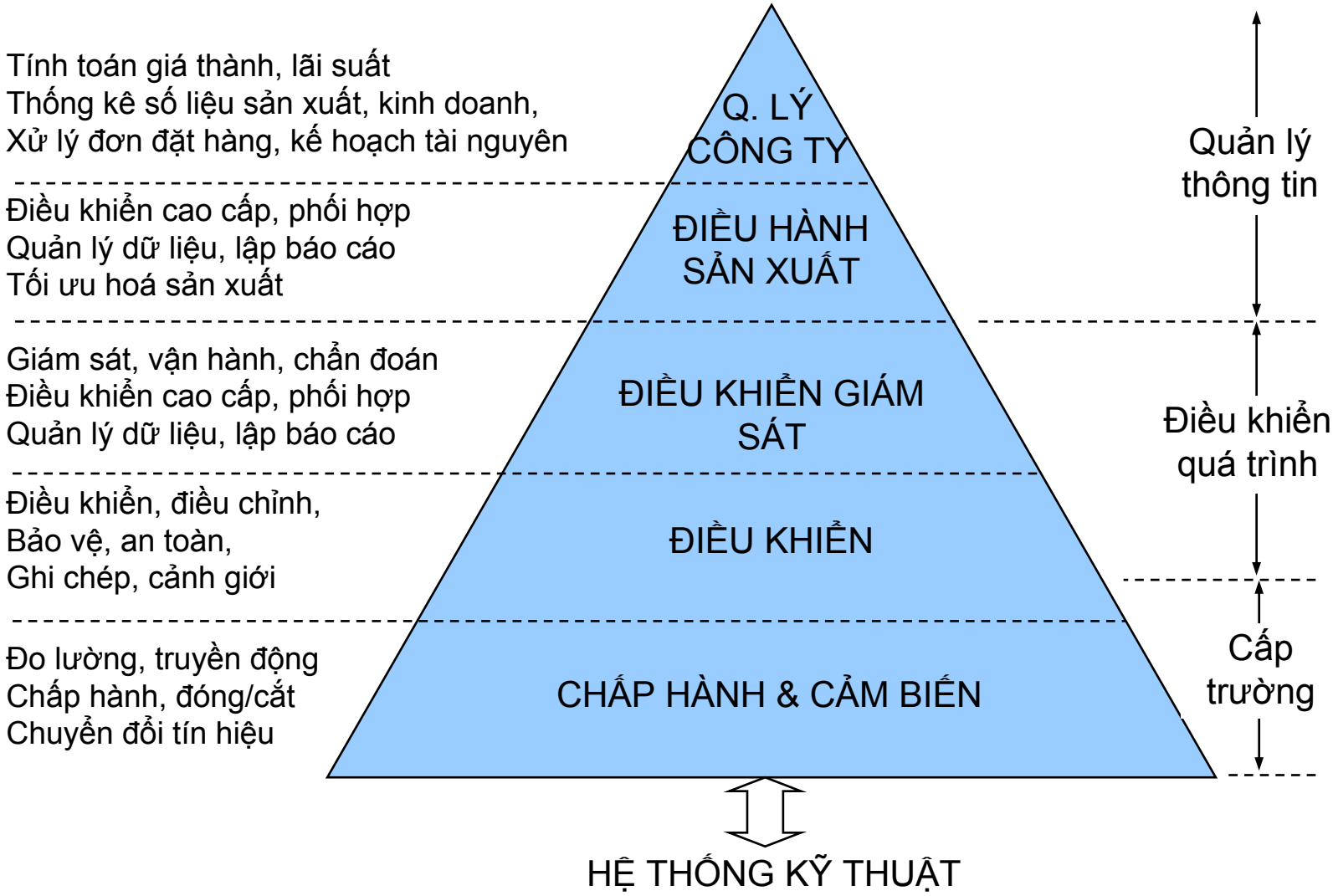
2.1 Cấu trúc cơ bản một HTĐK&GS



Các thành phần cơ bản

- Hệ thống máy tính điều khiển: Các hệ thống máy tính điều khiển chuyên dụng hoặc phổ thông.
- Giao diện quá trình: Giao diện giữa các MTĐK với hệ thống kỹ thuật thông qua các thiết bị đo lường và truyền động.
- 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: Các thiết bị bảo vệ và cơ chế thực hiện chức năng an toàn hệ thống.

2.2 Mô hình phân cấp chức năng

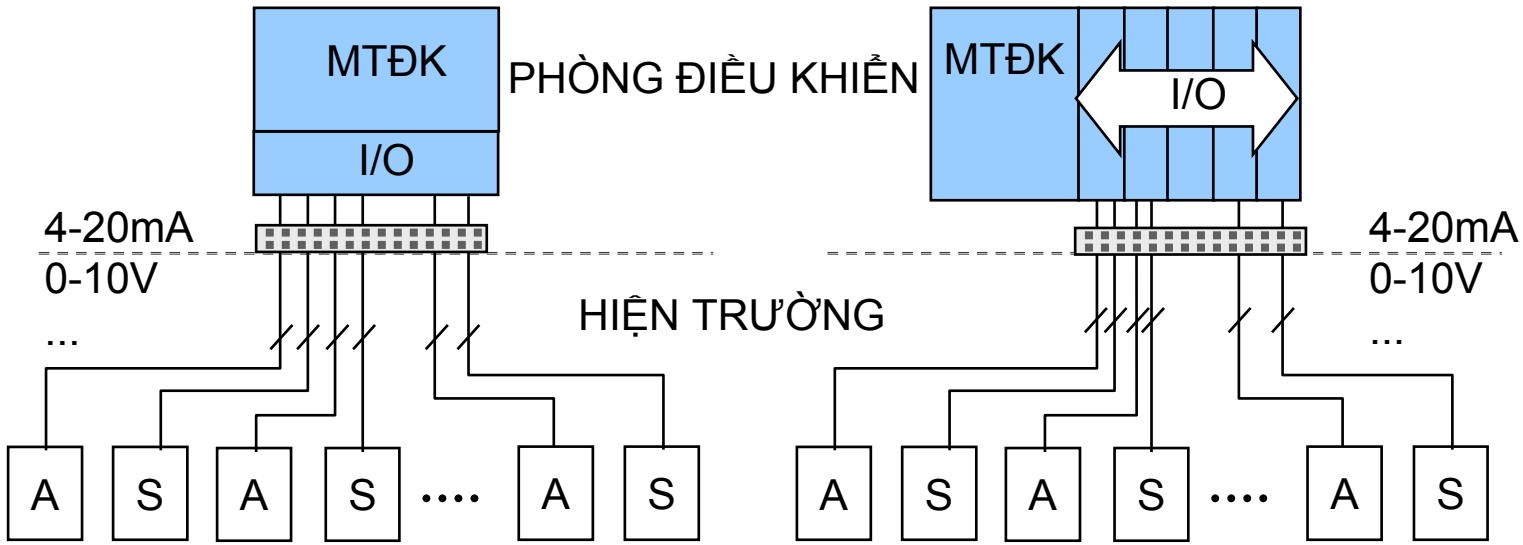


Mục đích phân cấp

- Định nghĩa các cấp theo chức năng, không phụ thuộc lĩnh vực công nghiệp cụ thể. Mỗi cấp có chức năng và đặc thù khác nhau
- Với mỗi ngành công nghiệp, lĩnh vực ứng dụng có thể có các mô hình tương tự với số cấp nhiều hoặc ít hơn
- Ranh giới giữa các cấp không phải bao giờ cũng rõ ràng.
- 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.
- Càng ở cấp trên quyết định càng quan trọng hơn, lượng thông tin cần trao đổi và xử lý càng lớn hơn.
- Phân cấp tiện lợi cho công việc thiết kế hệ thống

2.3 Cấu trúc vào/ra

Vào/ra tập trung (central I/O)



a) Vào/ra tích hợp

b) Vào/ra kiểu module

I/O: input/outputA: actuator S: sensor

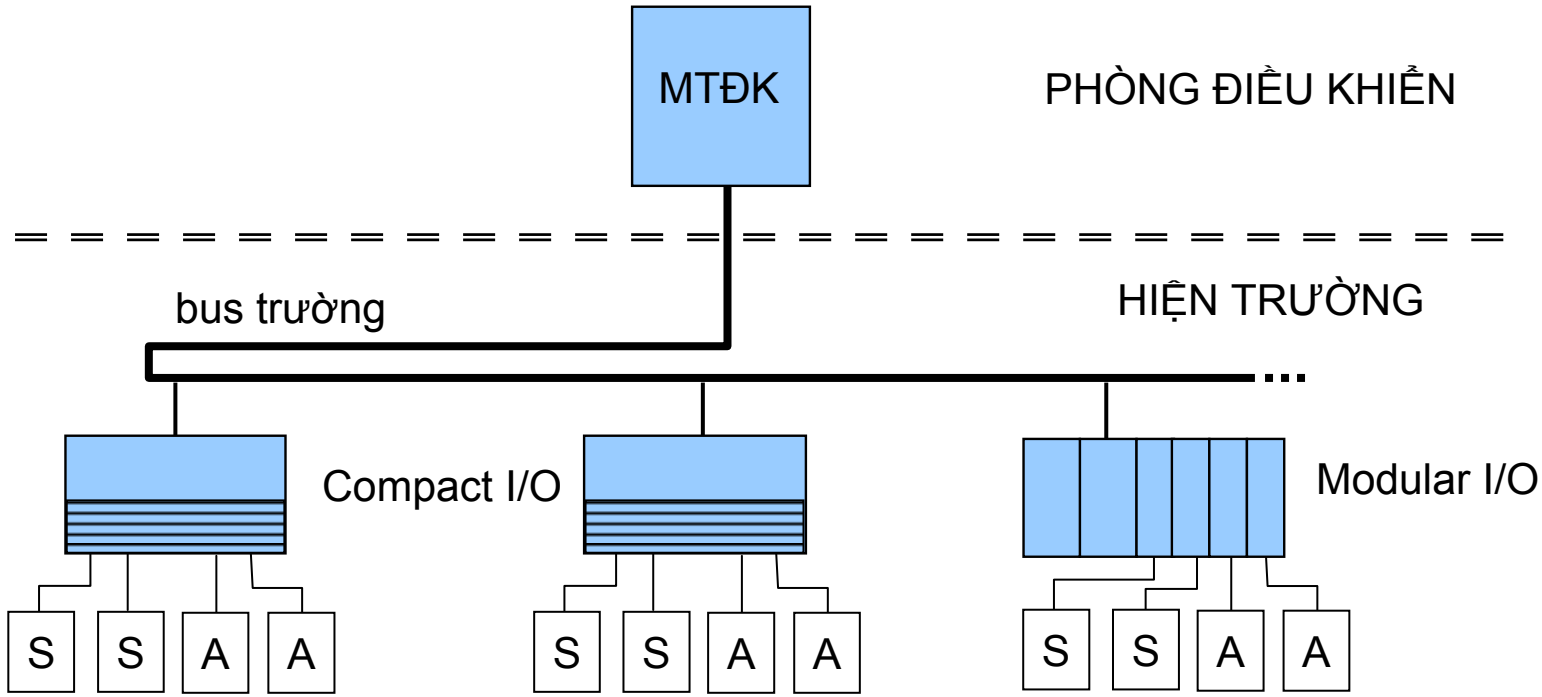
➔ Nối dây truyền thống

Vào/ra tập trung: Ưu và nhược điểm

- Công việc nối dây phức tạp, chi phí cho cáp dẫn cao: số lượng lớn các cáp nối, cấu trúc phức tạp, công thiết kế, lắp đặt lớn.
- Kém tin cậy: Phương pháp truyền dẫn tín hiệu tương tự giữa các thiết bị trường và thiết bị điều khiển dễ chịu ảnh hưởng của nhiễu, gây ra sai số mà không có khả năng phát hiện.
- Kém linh hoạt: Khó mở rộng bởi phải đi lại cáp dẫn, không thể lựa chọn các module vào/ra của hãng khác.
- Khó chẩn đoán lỗi: Một mặt lỗi do truyền tín hiệu khó phát hiện ra, mặt khác lỗi do thiết bị rất khó có thể định vị và đưa ra kết luận chẩn đoán.
- Phù hợp với các hệ thống qui mô nhỏ: Phạm vi địa lý hẹp, một máy tính điều khiển, số lượng vào/ra không lớn

Vào/ra phân tán (distributed I/O)

Còn gọi là vào/ra từ xa (*remote I/O*)

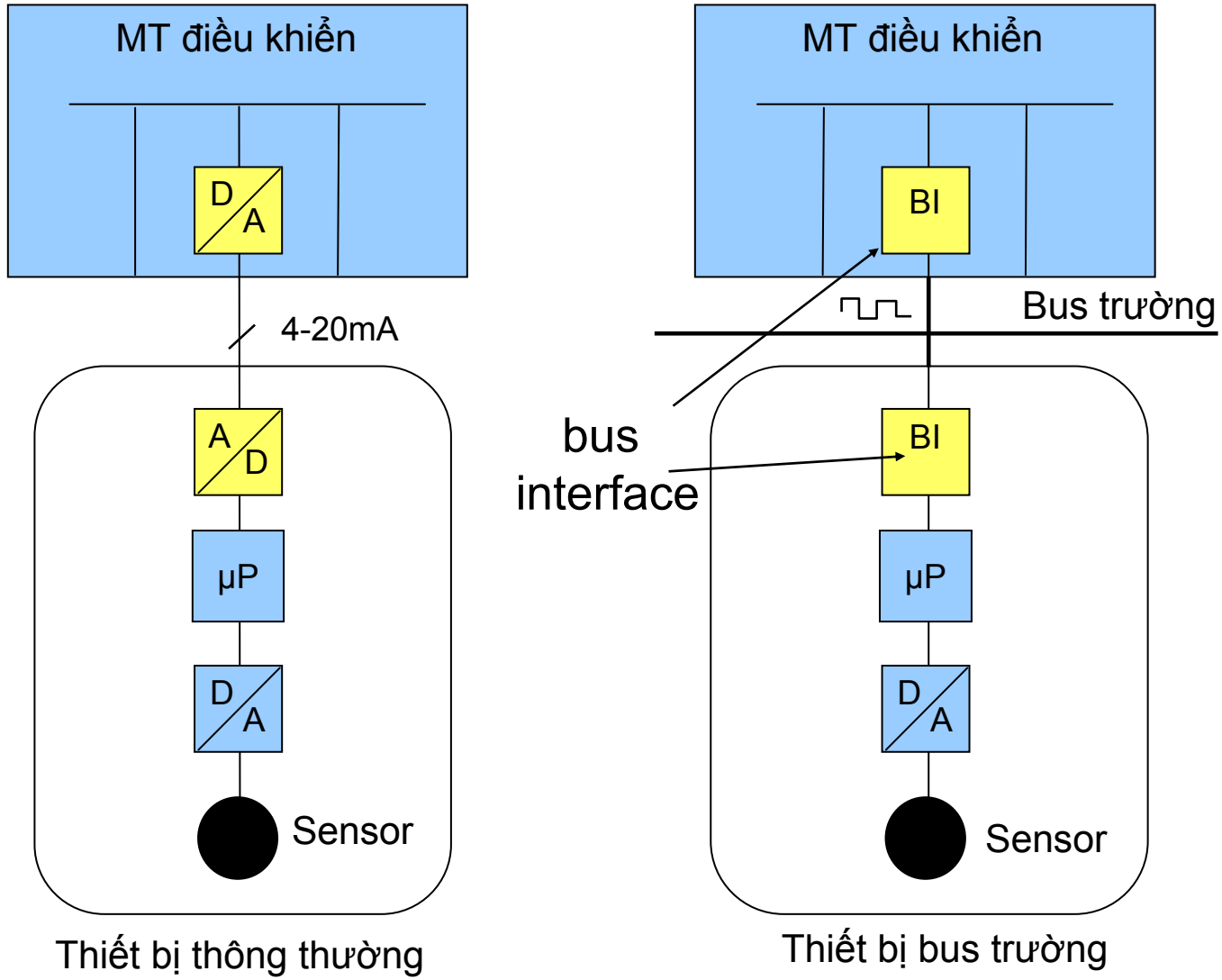


➔ Ưu điểm nhiều, song vẫn còn nối dây truyền thống

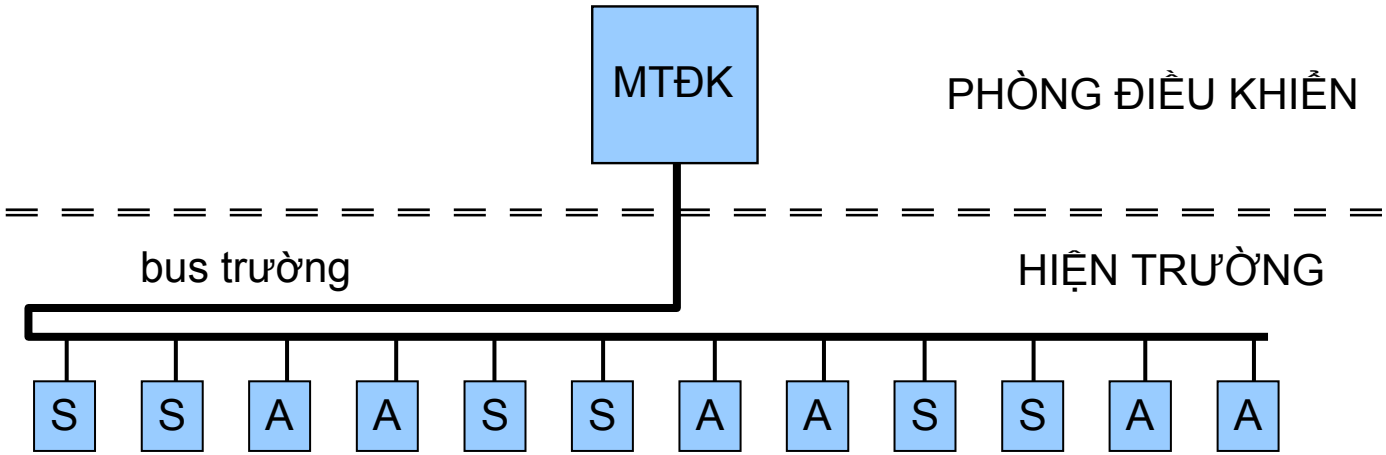
Vào/ra phân tán với bus trường chuẩn

- Tiết kiệm chi phí dây dẫn và công lắp đặt: Từ bộ điều khiển xuống tới các vào/ra phân tán chỉ cần một đường truyền duy nhất.
- Cấu trúc đơn giản: Thiết kế và bảo trì hệ thống dễ dàng hơn.
- Tăng độ tin cậy của hệ thống:
  - Truyền kỹ thuật số => hạn chế lỗi được hạn chế
  - Nếu có lỗi truyền thông cũng dễ dàng phát hiện nhờ các biện pháp bảo toàn dữ liệu của hệ bus.
- Tăng độ linh hoạt của hệ thống:
  - Tự do hơn trong lựa chọn các thiết bị vào/ra
  - Tự do hơn trong thiết kế cấu trúc hệ thống.
  - Khả năng mở rộng dễ dàng hơn
- Vào/ra phân tán không nhất thiết phải đặt gần tại hiện trường (chỉ lợi dụng ưu điểm cuối cùng)

Thiết bị thường và thiết bị bus trường



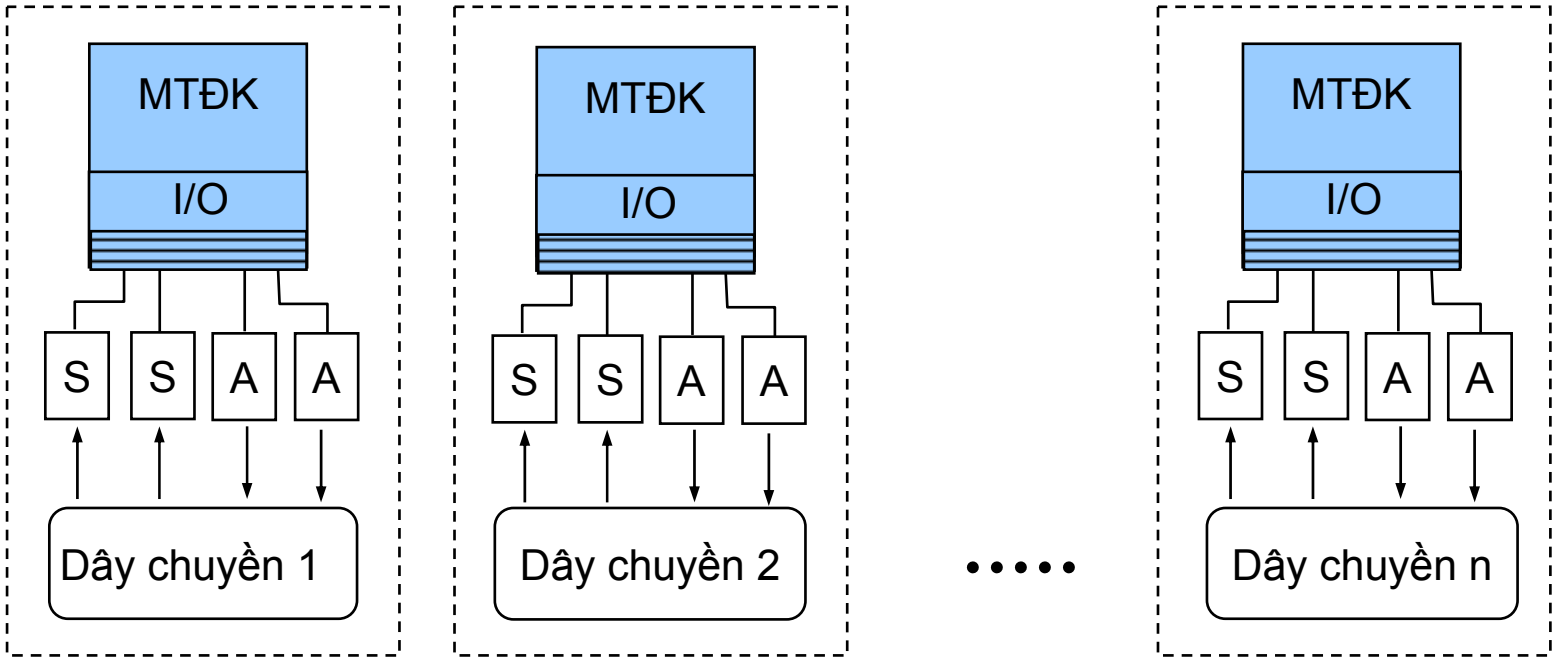
Vào/ra trực tiếp với thiết bị bus trường



- Cấu trúc đơn giản, dễ thiết kế và lắp đặt
- Giảm chi phí cáp truyền, các khối vào/ra và các phụ kiện khác
- Giảm kích thước tủ điều khiển
- Đưa vào vận hành và khả năng chẩn đoán các thiết bị trường qua mạng một cách dễ dàng.
- Khả năng tích hợp các chức năng điều khiển tự động xuống các thiết bị trường => *trí tuệ phân tán (distributed intelligence)*

2.4 Cấu trúc điều khiển

Điều khiển cục bộ/điều khiển song song

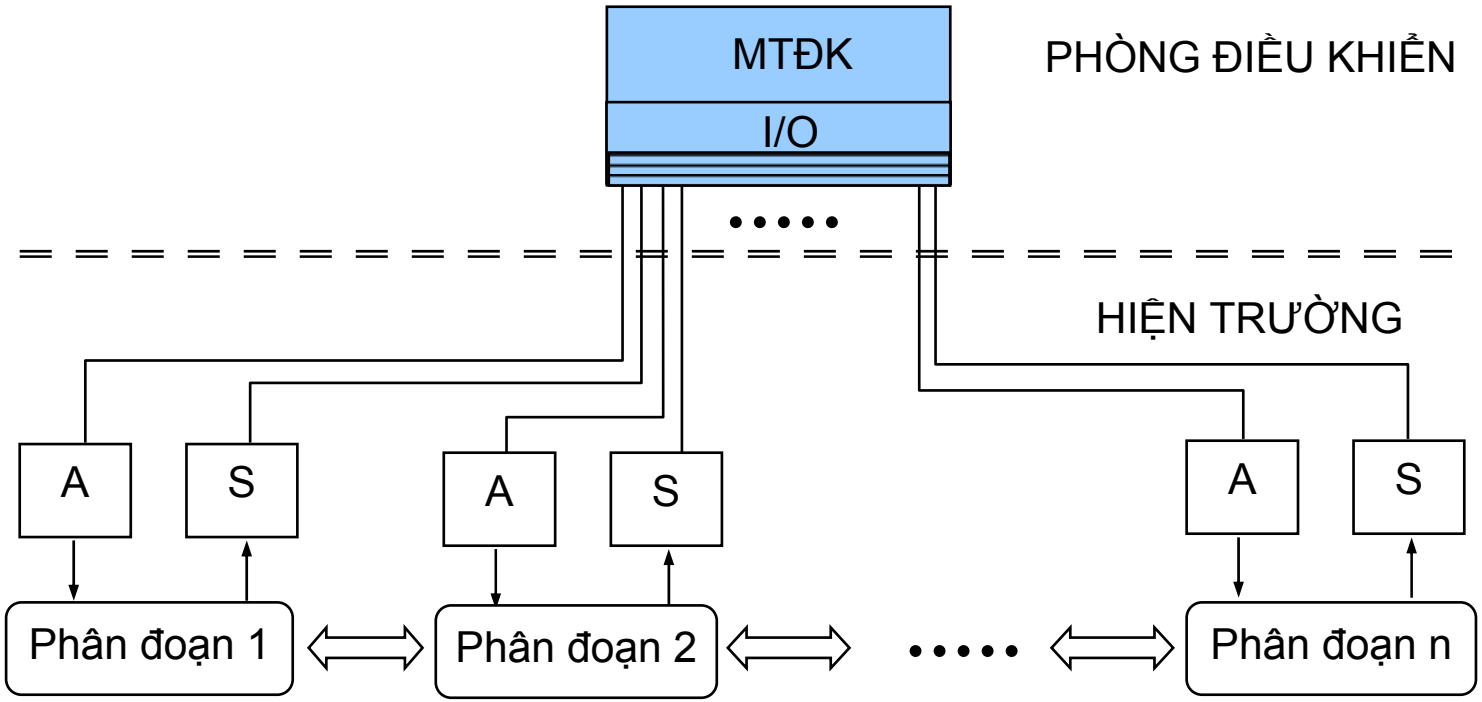


Điều khiển song song: ưu và nhược điểm

- Cấu trúc cổ điển nhất
- Thường được sử dụng cho các hệ thống có qui mô vừa và nhỏ, đặc biệt tiêu biểu trong các ngành công nghiệp chế tạo, lắp ráp (các dây chuyền song song độc lập với nhau)
- Các thiết bị điều khiển được đặt tại hiện trường
- Có thể sử dụng kết hợp cấu trúc vào/ra tập trung hoặc vào/ra trực tiếp với bus trường.
- Các máy tính điều khiển làm việc hoàn toàn độc lập với nhau => độ tin cậy cao
- Hoàn toàn không có sự phối hợp giữa chúng để cùng chia sẻ giải quyết cùng một nhiệm vụ.
- Một số môi trường công nghiệp không cho phép lắp đặt các thiết bị điều khiển tại hiện trường.

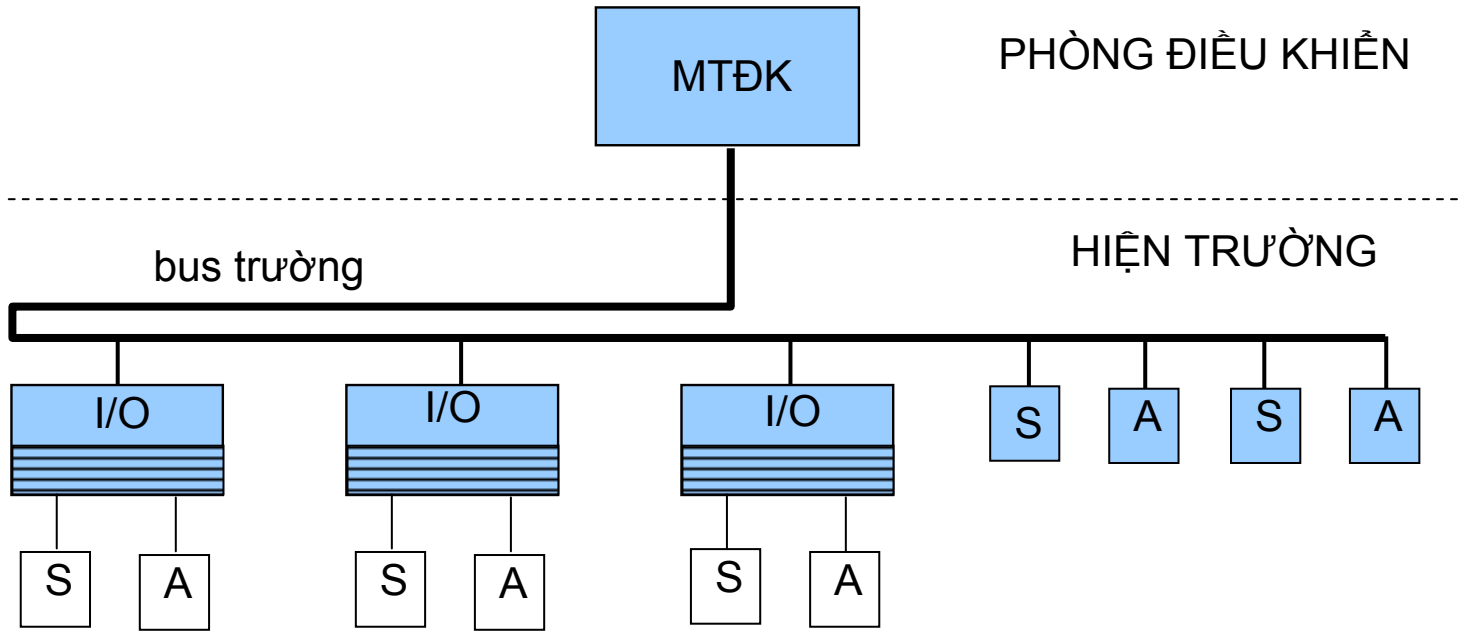
Điều khiển tập trung (centralized control)

Nối dây truyền thống



Điều khiển tập trung

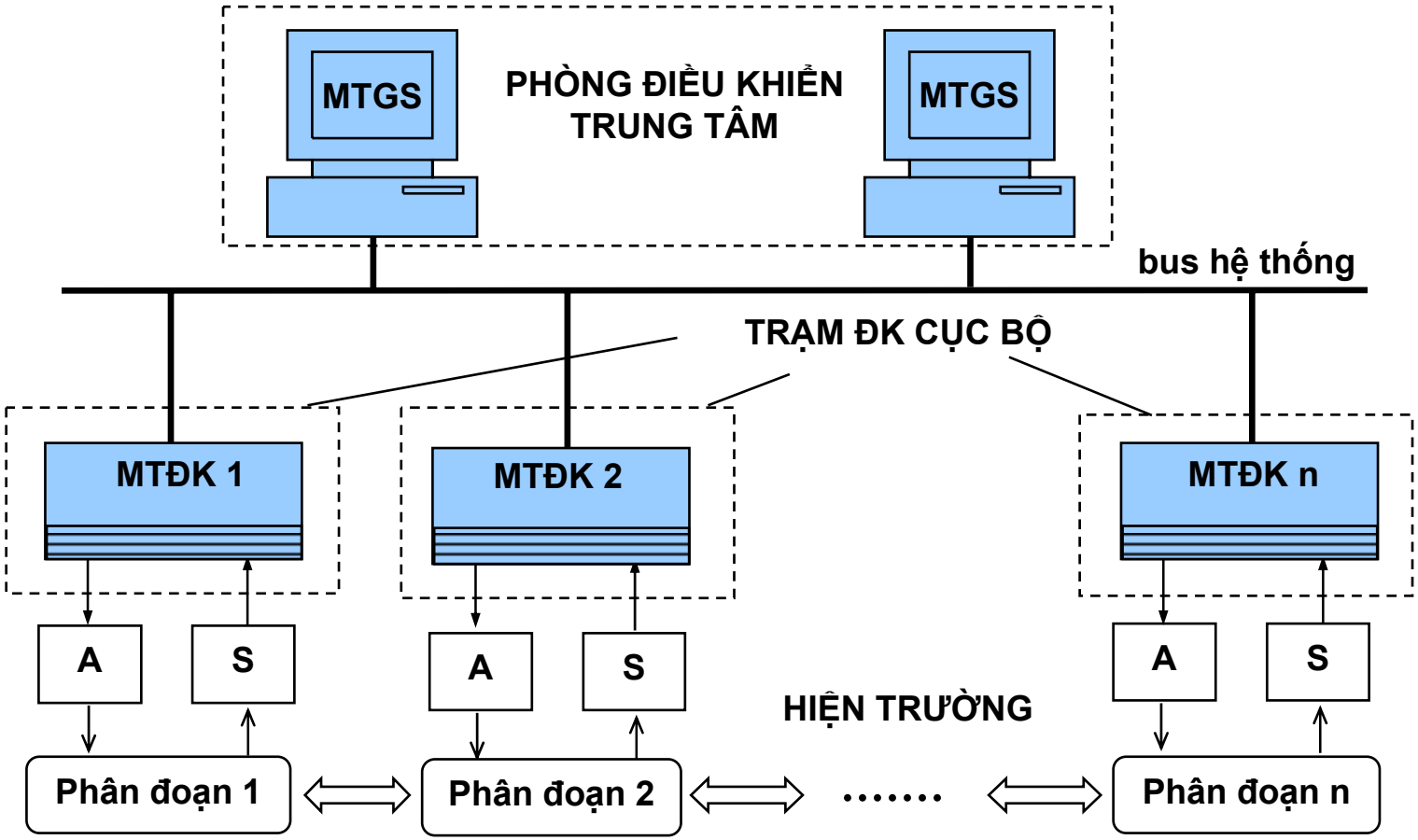
Sử dụng bus trường



Điều khiển tập trung: ưu và nhược điểm

- Độ tin cậy thấp: Tập trung chức năng điều khiển và xử lý thông tin tại một máy tính duy nhất
- Độ linh hoạt thấp: Mở rộng cũng như thay đổi một phần trong hệ thống đòi hỏi phải dừng toàn bộ hệ thống.
- Hiệu năng kém: Toàn bộ thông tin đều phải đưa về trung tâm, chậm trễ do thời gian truyền dẫn và xử lý tập trung
- Chỉ phù hợp với các ứng dụng qui mô nhỏ

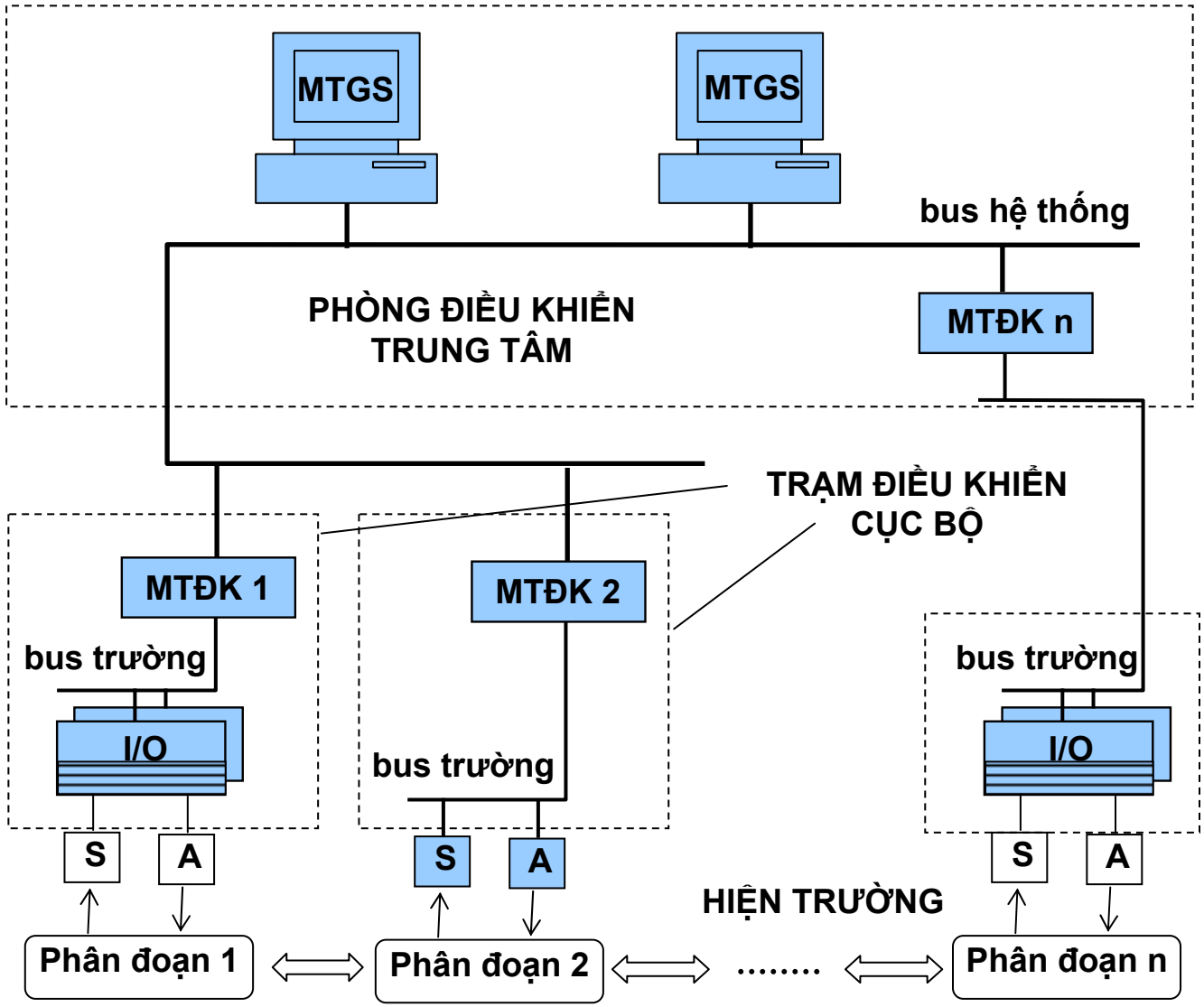
Điều khiển phân tán (distributed control



Điều khiển phân tán: ưu điểm

- Phân chia chức năng điều khiển xuống các máy tính điều khiển tại các trạm cục bộ (ở vị trí không xa với quá trình kỹ thuật).
- Điều khiển phối hợp giữa các máy tính điều khiển có thể diễn ra trực tiếp hoặc thông qua các máy tính giám sát trung tâm (MTGS).
- Độ linh hoạt cao hơn hẳn 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.
- 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.

Điều khiển phân tán sử dụng bus trường



Hệ thống điều khiển phân tán

Chương 3: Kiến trúc PLC/HMI

8/14/2006

Chương 3: Kiến trúc PLC/HMI

3.1 Giới thiệu sơ lược về PLC
Lịch sử phát triển của PLC
Các ưu nhược điểm chính
3.2 Cấu hình cơ bản một hệ PLC/HMI
Cấu trúc máy tính PLC
Thiết kế phần cứng PLC
3.3 Phương pháp lập trình PLC
Chuẩn IEC 61131-3
3.4 SCADA/HMI cho giải pháp PLC
3.5 Các điểm mấu chốt trong kiến trúc PLC/HMI
So sánh DCS và PLC/HMI

3.1 Giới thiệu sơ lược về PLC

- PLC (*Programmable Logic Controller*):
  - Thiết bị điều khiển có thể „lập trình mềm“, làm việc theo chương trình lưu trong bộ nhớ => máy tính điều khiển chuyên dụng
  - Thích hợp nhất cho điều khiển logic (thay thế các rơle), song cũng có thể chức năng điều chỉnh (PID, mờ,...) và các chức năng tính toán khác
  - Ngày nay khái niệm „Programmable Controller“ được sử dụng nhiều hơn, mặc dù từ viết tắt „PLC“ vẫn thông dụng
- Phạm vi ứng dụng:
  - Lúc đầu chủ yếu trong các ngành công nghiệp chế tạo, điều khiển các quá trình rời rạc
  - Ngày nay cả trong điều khiển trình tự và điều khiển quá trình liên tục -> cạnh tranh với Compact Digital Controllers và các hệ DCS trong các ứng dụng “lai”
  - Thiết bị thu thập dữ liệu trong các hệ SCADA

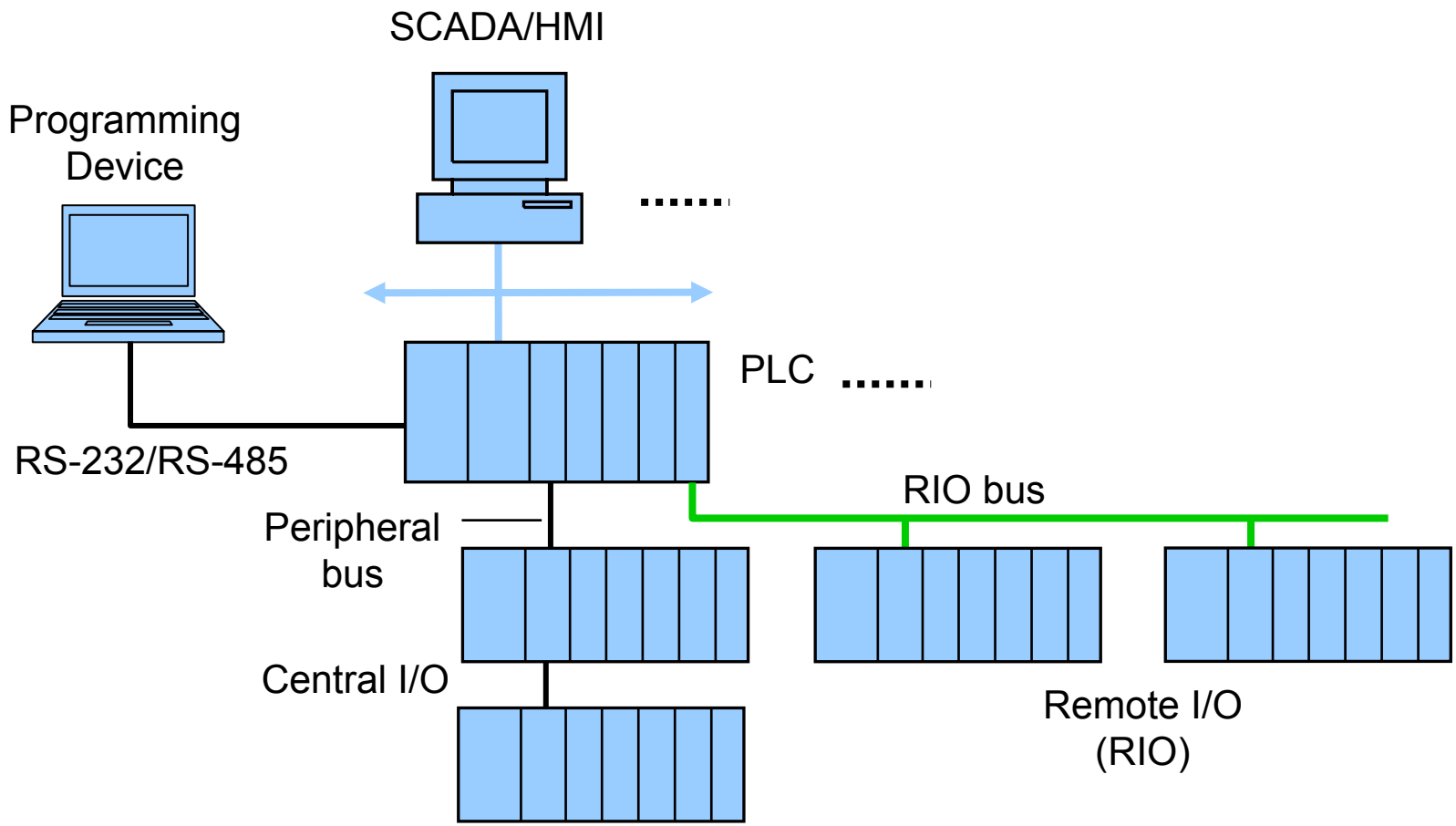
Lịch sử phát triển

- 1968: Richard Morley sáng tạo ý tưởng PLC cho General Motors
- 1969: PLC đầu tiên (Allen Bradley và Bedford), được GM sử dụng trong công nghiệp ô-tô (128 DI/DO, 1kByte bộ nhớ)
- 1971: Ứng dụng PLC đầu tiên ngoài CN ô-tô
- 1973: PLC „thông minh“ với khả năng tính toán, điều khiển máy in, xử lý dữ liệu, giao diện màn hình
- 1975: PLC với bộ điều khiển PID
- 1976: Lần đầu tiên sử dụng trong hệ thống phân cấp điều khiển dây chuyền sản xuất
- 1977: mP-based PLC
- 1980: Các module vào/ra thông minh
- 1981: PLC nối mạng, 16-bit PLC, các màn hình CRT màu
- 1982: PLC lớn với 8192 I/O ra đời
- 1992: Chuẩn IEC 61131 (phần 1-5)
- 1996: Slot-PLC, Soft-PLC,...

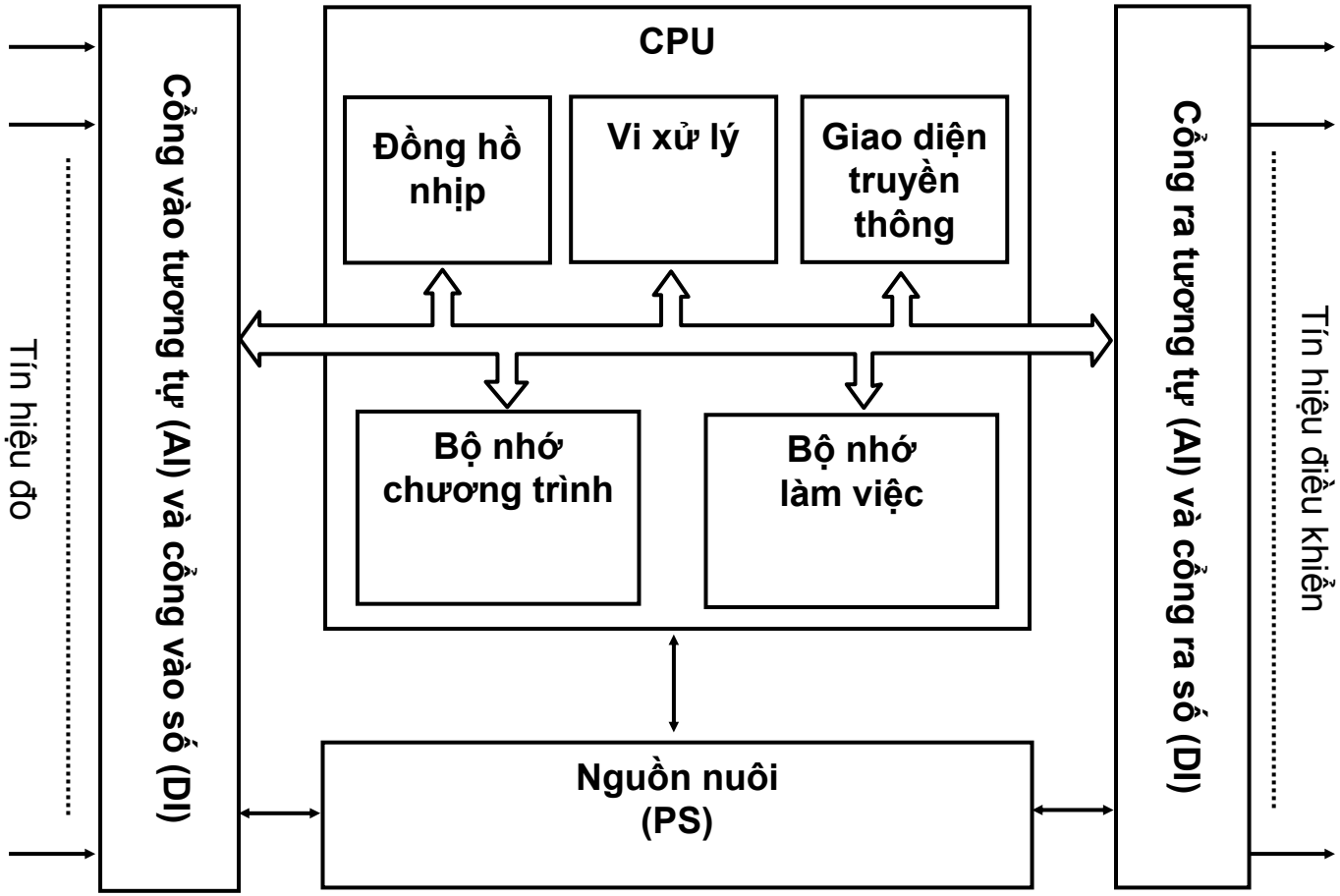
Các ưu nhược điểm chính

- Ưu điểm:
  - Phần cứng gọn nhẹ, thiết kế bền chắc, độ tin cậy cao, thích hợp với môi trường làm việc công nghiệp
  - Khả năng xử lý tín hiệu logic (24VDC-240VAC) và tín hiệu tương tự
  - Khả năng mở rộng số đầu vào/ra đơn giản
  - Lập trình và thay đổi chương trình đơn giản với kỹ sư điện
  - Khả năng giám sát hoạt động của dây chuyền SX, khả năng phát hiện lỗi thiết bị trường từ máy tính điều khiển
  - Tính năng thời gian thực
- Nhược điểm
  - Giải pháp đơn lẻ, cần tích hợp giao diện người-máy (HMI)
  - Kiến trúc đóng kín, khó tích hợp sản phẩm ngoài
  - Năng lực tính toán tương đối yếu

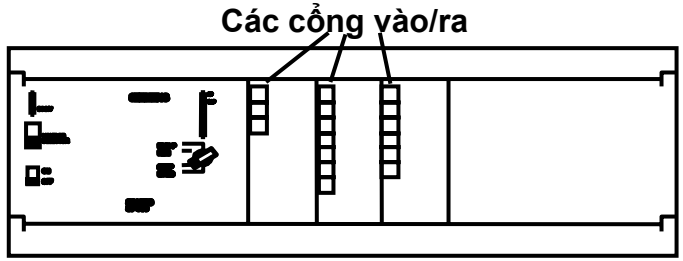
3.2 Cấu hình cơ bản một hệ PLC+HMI



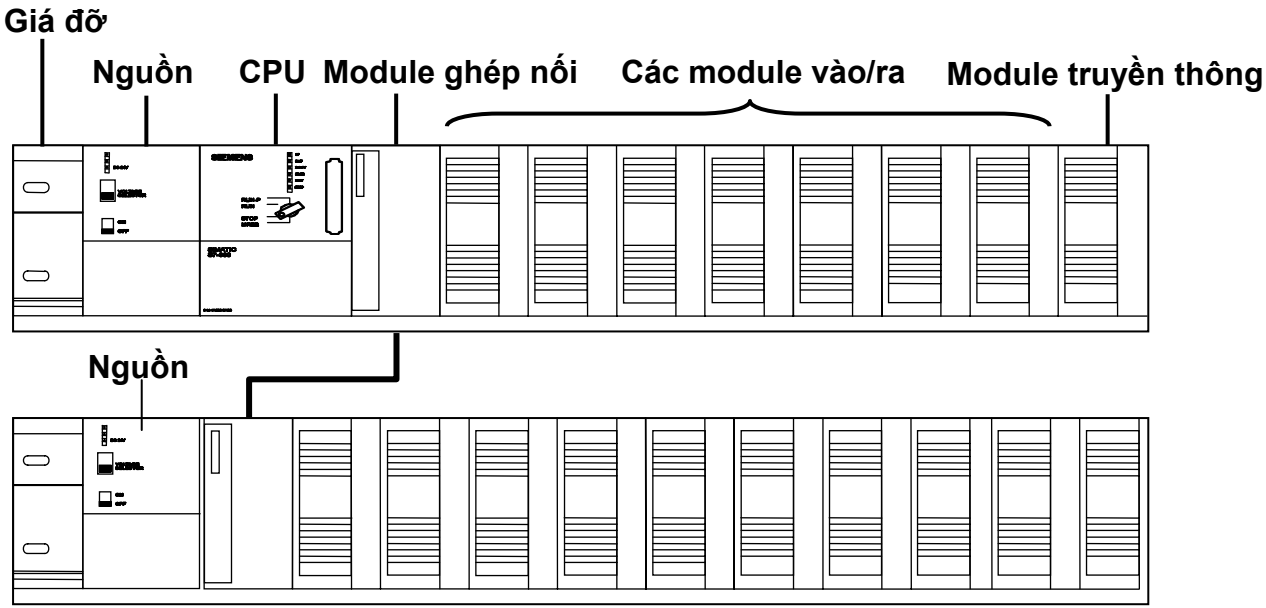
Cấu trúc máy tính PLC



Thiết kế phần cứng PLC

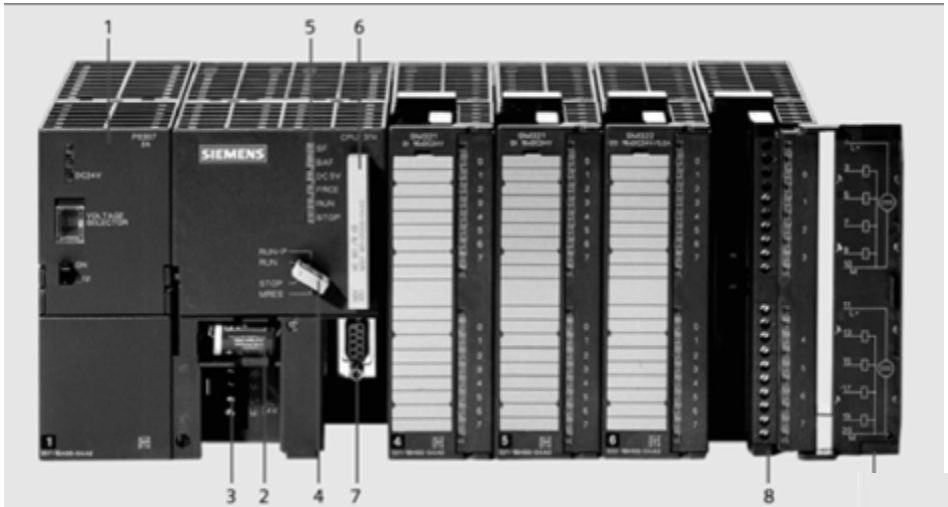


a) Thiết kế gọn

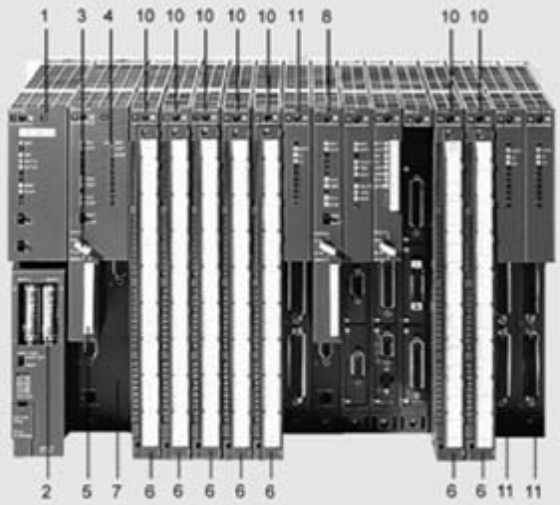


b) Thiết kế module

Ví dụ: SIMATIC S7-300/S7-400

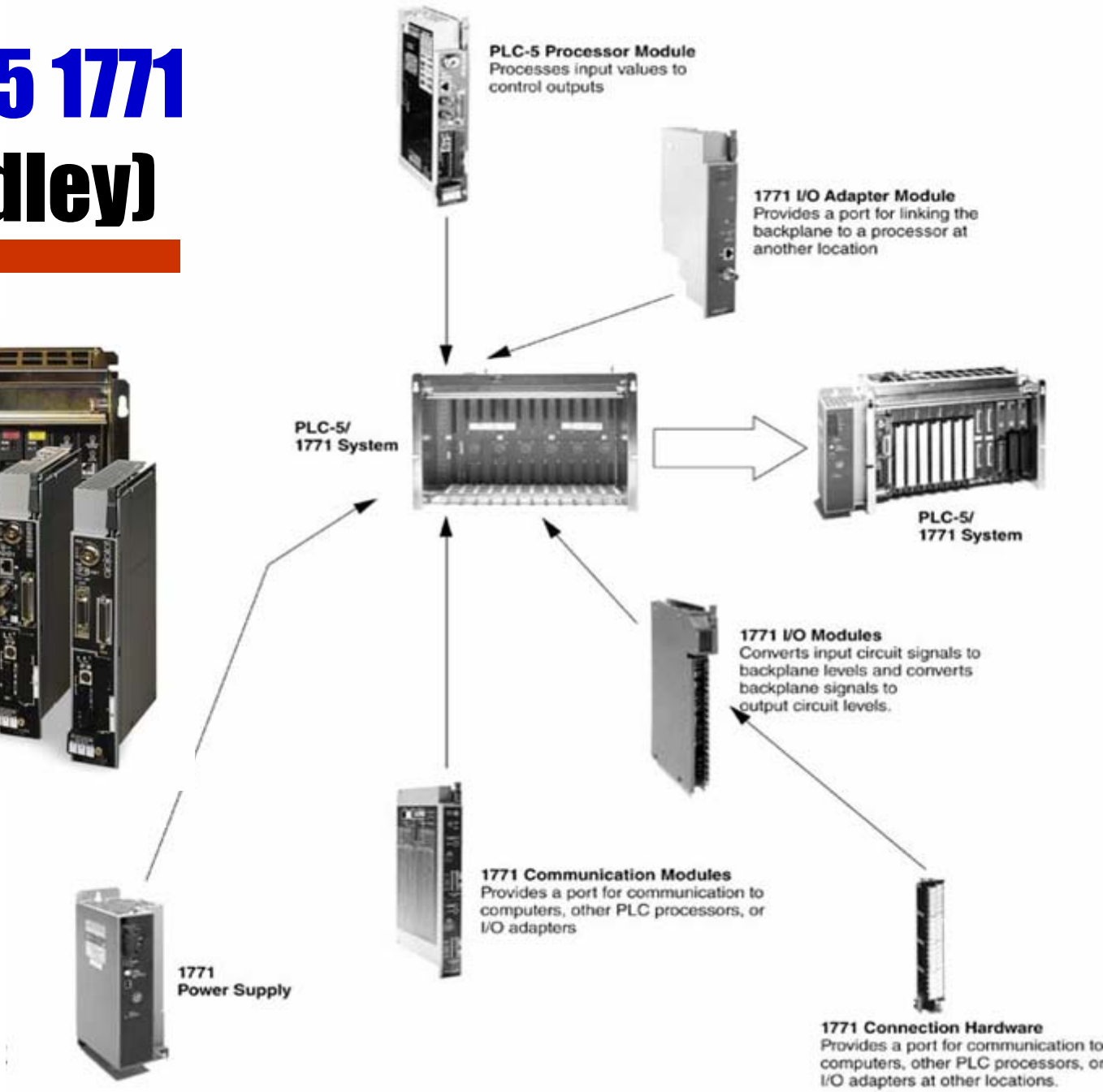


1: Load power supply (optional)
2: Backup battery (CPU 313 and above)
3: 24 V DC connection
4: Mode switch (Key-operated)
5: Status and fault LEDs
6: Memory card (CPU 313 and above)
7: MPI (multi-point interface)
8: Front connector
9: Front door



1: Power supply module
2: Buffer batteries
3: Mode switch (key-operated)
4: Status and fault LEDs
5: Memory card
6: Front connector with labelling field
7: CPU 1
8: CPU 2
10: I/O modules
11: IM interface modules

Ví dụ: PLC-5 1771 (Allen Bradley)



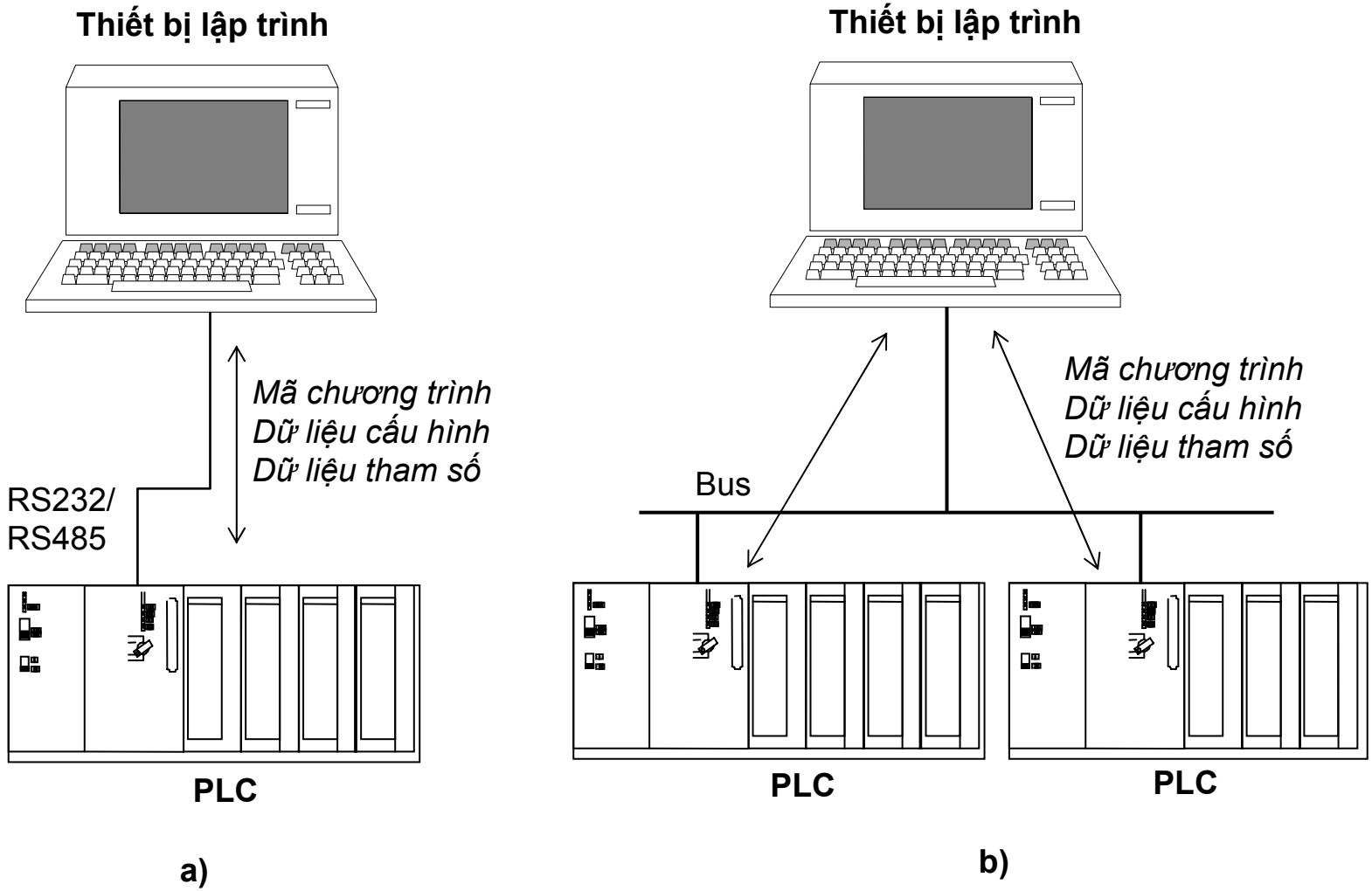
Các chức năng thông dụng của PLC

- Xử lý các giá trị vào/ra:
  - Chuyển đổi D/A, A/D, lọc nhiễu
  - Hạn chế tín hiệu ra
- Điều khiển:
  - Điều khiển logic, khóa liên động, điều khiển trình tự
  - *Điều chỉnh tự động: Điều khiển PID, điều khiển mờ*
- Tự chẩn đoán
- Xử lý truyền thông

Các tính năng quan trọng của PLC

- Tính năng thời gian thực
  - Hiệu năng CPU, dung lượng bộ nhớ
  - Xử lý đa nhiệm, theo sự kiện và theo thời gian
  - Chu kỳ vòng quét, chu kỳ task
- Khả năng ghép nối vào/ra
  - Các loại tín hiệu vào/ra
  - Vào/ra phân tán
  - Ghép nối bus trường, bus thiết bị
- Lập trình thuận tiện
  - Ngôn ngữ theo chuẩn quốc tế IEC 61131-3
  - Khả năng điều khiển lai (liên tục, trình tự và logic)
  - Thư viện khối chức năng mạnh
- Khả năng ghép nối truyền thông
  - Khả năng hỗ trợ các chuẩn giao thức
  - Khả năng lập trình phân tán

3.3 Phương pháp lập trình



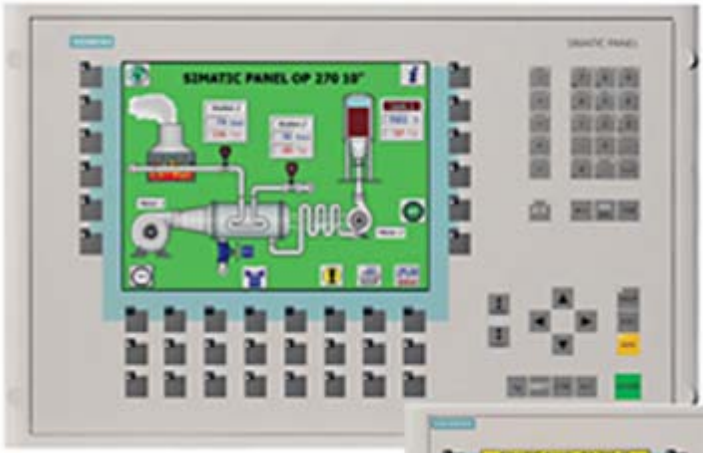
Chuẩn IEC 61131: Programmable controllers

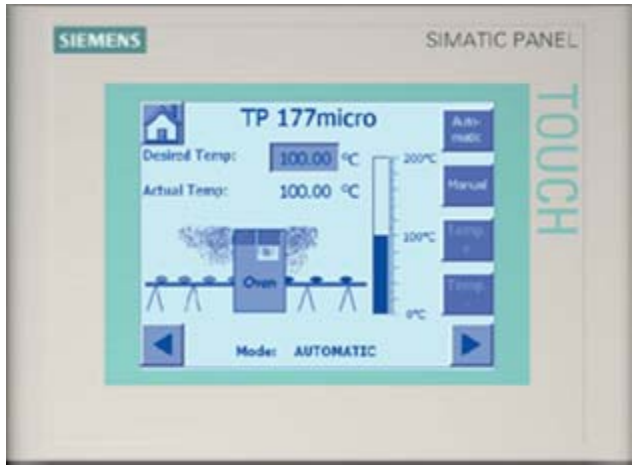
- Part 1 (*General Information*): Các định nghĩa chung và các đặc tính chức năng tiêu biểu (cơ chế thực hiện tuần hoàn, ảnh quá trình, thiết bị lập trình và giao diện người-máy)
- Phần 2 (*Equipment requirements*): Các yêu cầu điện học, cơ học và chức năng cho các thiết bị (nhiệt độ, độ ẩm, cung cấp nguồn, độ kháng nhiễu, phạm vi tín hiệu logic và sức bền cơ học), phương pháp kiểm tra và thử nghiệm các kiểu thiết bị.
- Phần 3 (*Programming languages*): Mô hình phần mềm và các ngôn ngữ lập trình.
- Phần 4 (*Guidelines for users*): Các nguyên tắc chỉ đạo cho nhà tích hợp hệ thống (phân tích hệ thống, lựa chọn thiết bị, vận hành, bảo trì hệ thống).
- Phần 5 (*Communication*): Các khối chức năng truyền thông giữa các PLC cũng như giữa PLC và một thiết bị khác trên cơ sở các khối hàm chuẩn.

3.4 Máy tính giám sát SCADA/HMI

- Thiết bị HMI chuyên dụng:
  - Operator Pannel (OP), Touch Pannel (TP)
  - Thiết kế gọn, độ bền công nghiệp cao, giá thành cao
  - Riêng cho một họ PLC (giao thức độc quyền) hoặc cho nhiều họ PLC (giao thức mở)
  - Phần mềm cơ sở cài đặt sẵn
  - Phát triển ứng dụng HMI với PC + phần mềm công cụ phát triển cung cấp kèm theo
- PC phổ thông, PC công nghiệp:
  - Sử dụng đa năng, linh hoạt
  - Phát triển ứng dụng với một công cụ SCADA/HMI chuyên dụng (độc lập) hoặc với một công cụ lập trình phổ thông
  - Có thể thực hiện thêm các chức năng quản lý dữ liệu, điều khiển cao cấp,...
  - Kết hợp với PLC để xây dựng một hệ lớn

Thiết bị HMI chuyên dụng





Phần mềm HMI/SCADA chuyên dụng

- Thông thường chạy trên PC + Windows
- Có khả năng ghép nối với nhiều loại PLC của nhiều hãng
- Phát triển ứng dụng bằng cách *soạn thảo* và *cấu hình* thay cho lập trình
- > 50 sản phẩm có mặt trên thị trường
- Các sản phẩm nổi tiếng:
  - iFIX (Intellution, GE Fanuc)
  - InTouch, Factory Suits (Wonderware)
  - WinCC (Siemens)
  - RSView (Rockwell Automation)

3.5 Các điểm mấu chốt của kiến trúc PLC/HMI

- Kiến trúc hệ thống: linh hoạt, lỏng lẻo
  - Theo chiều ngang: PLC không được thiết kế ngay từ đầu cho cấu trúc điều khiển phân tán
  - Theo chiều dọc: tách biệt rõ rệt giữa cấp điều khiển với cấp điều khiển giám sát, không có cơ sở dữ liệu chung
- Phát triển hệ thống: riêng biệt cho từng phần
- Giao diện quá trình: chủ yếu theo cách nối dây truyền thống (vào/ra tập trung hoặc vào/ra từ xa)
- Cơ chế làm việc: chủ yếu theo cơ chế vòng quét hoặc theo sự kiện
  - Thích hợp với điều khiển logic, điều khiển trình tự
  - Ít thích hợp với các bài toán điều khiển quá trình

Hệ thống điều khiển phân tán

Chương 4: Kiến trúc DCS

8/27/2005

Chương 4: Kiến trúc DCS

4.1 Giới thiệu chung

Phạm vi ứng dụng

4.2 Cấu hình cơ bản của các hệ DCS

Giới thiệu các thành phần chính

Các ví dụ sản phẩm minh họa

4.3 Các điểm mấu chốt trong kiến trúc DCS

4.4 So sánh kiến trúc PLC/HIM với DCS

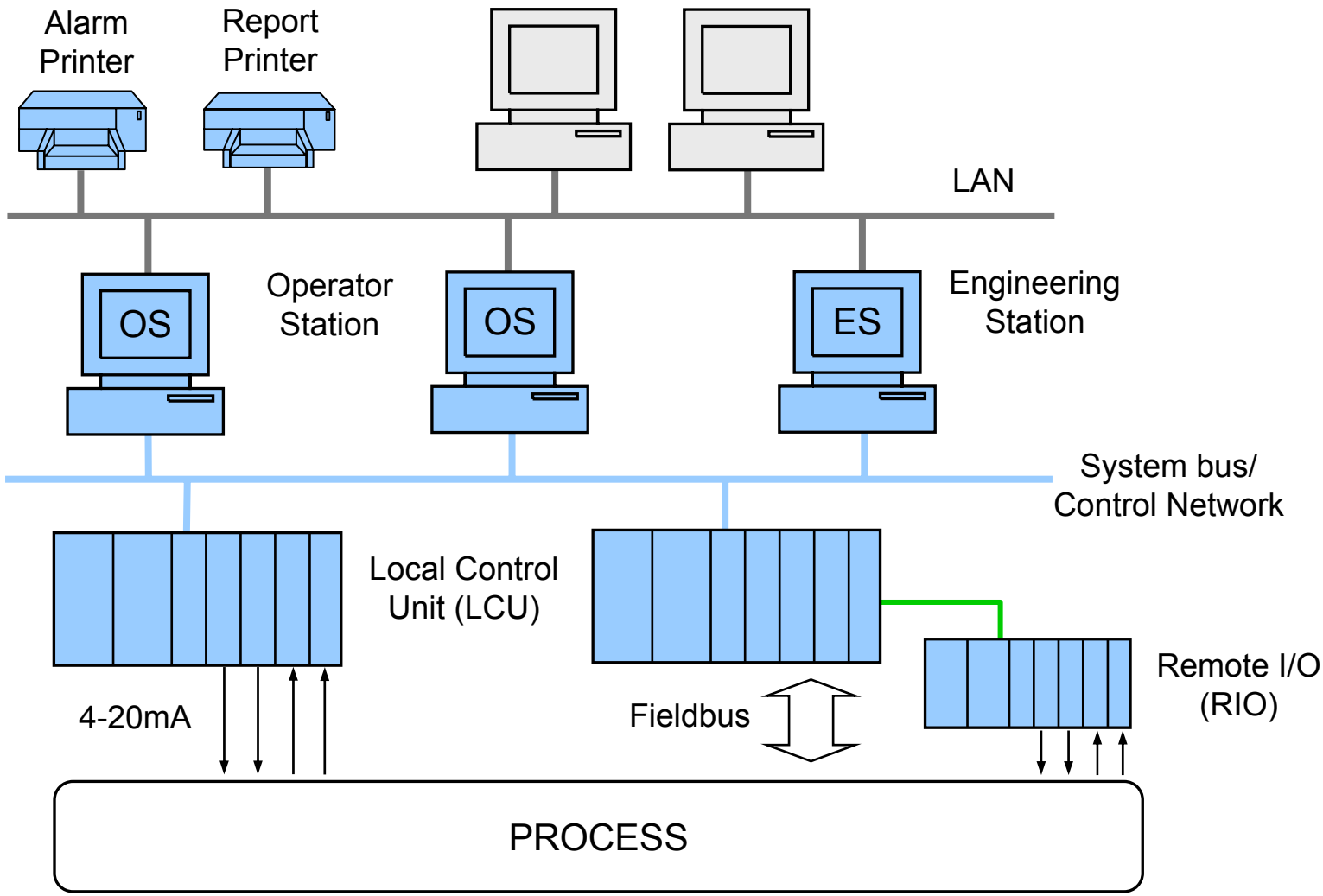
4.1 Giới thiệu chung về DCS

- DCS = Distributed Control System
  - Tiến hóa từ giải pháp DDC, phân tán chức năng điều khiển xuống nhiều bộ điều khiển nối mạng
  - Hệ điều khiển tích hợp trọn vẹn của một nhà sản xuất (tích hợp về phần cứng, phần mềm và chức năng)
- Hệ DCS đầu tiên: TDC2000 của Honeywell (1975)
- Các sản phẩm tiêu biểu:
  - ABB: Advant OCS, IndustrialIT
  - Emerson (Fisher-Rosemount): PROVOX, DeltaV
  - Honeywell: PlantScape
  - Invensys (Foxboro): I/A Series
  - Siemens: Teleperm, PCS7
  - Yokogawa: Centum CS1000/CS3000

Phạm vi ứng dụng của DCS

- Chủ yếu trong công nghiệp chế biến:
  - Hóa chất, hóa dầu, thực phẩm, mỹ phẩm, dược phẩm
  - Khai thác dầu khí, than
  - Điện năng, xi măng, giấy
  - Luyện kim, cán thép
  - ...
- Ưu điểm:
  - Tính tích hợp cao (phần mềm, phần cứng, giao tiếp, chức năng điều khiển và giám sát)
  - Phát triển ứng dụng trực quan, linh hoạt, đơn giản, gần gũi với công nghệ
  - Độ tin cậy cao nhờ khả năng độc lập cảnh giới, chẩn đoán lỗi của các trạm, cấu trúc phân tán và cấu hình dự phòng
- Nhược điểm:
  - Giá thành hệ thống tương đối cao
  - Nhiều hệ thống thể hiện tính đóng kín

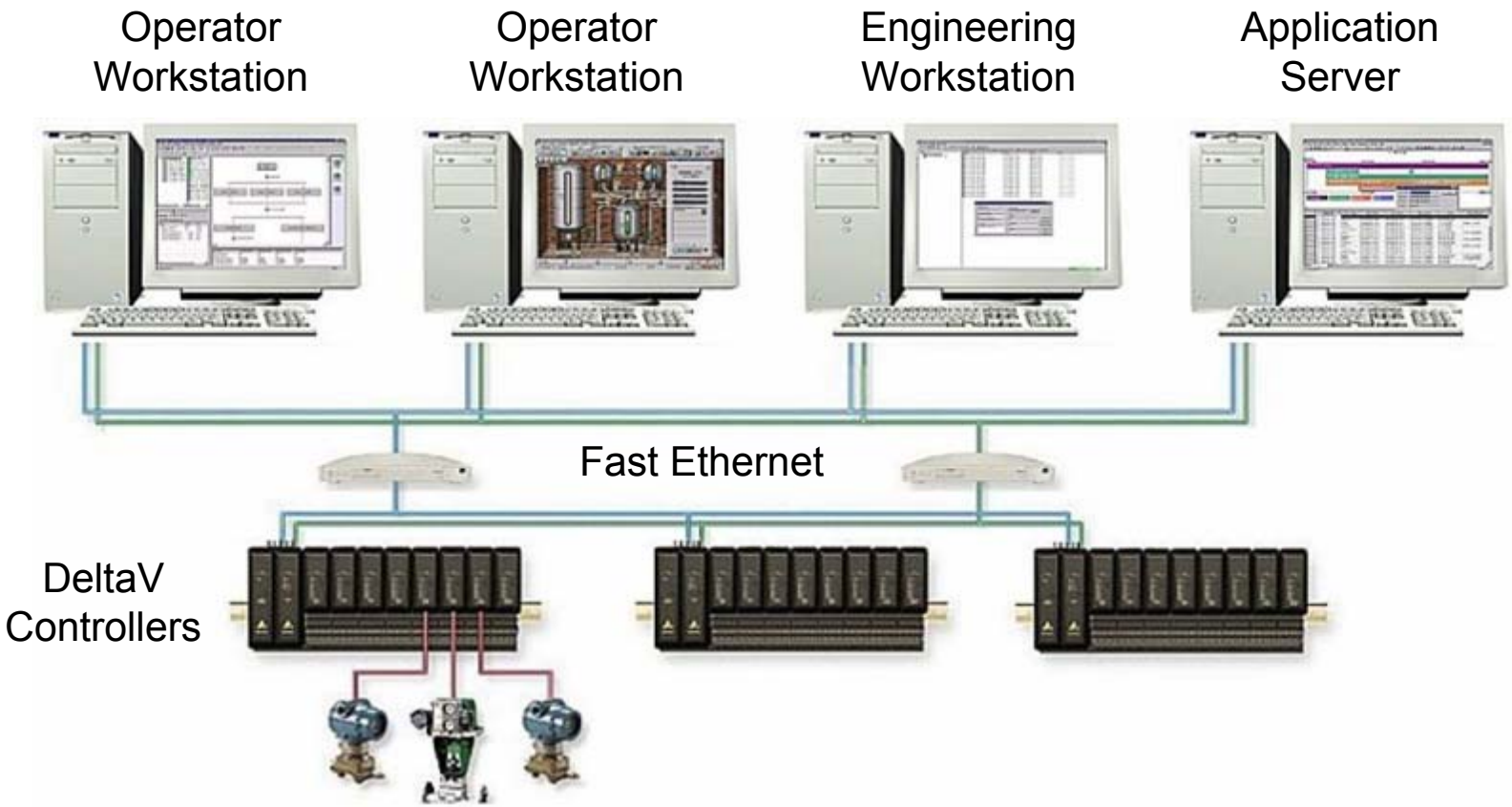
4.2 Cấu hình cơ bản một hệ DCS



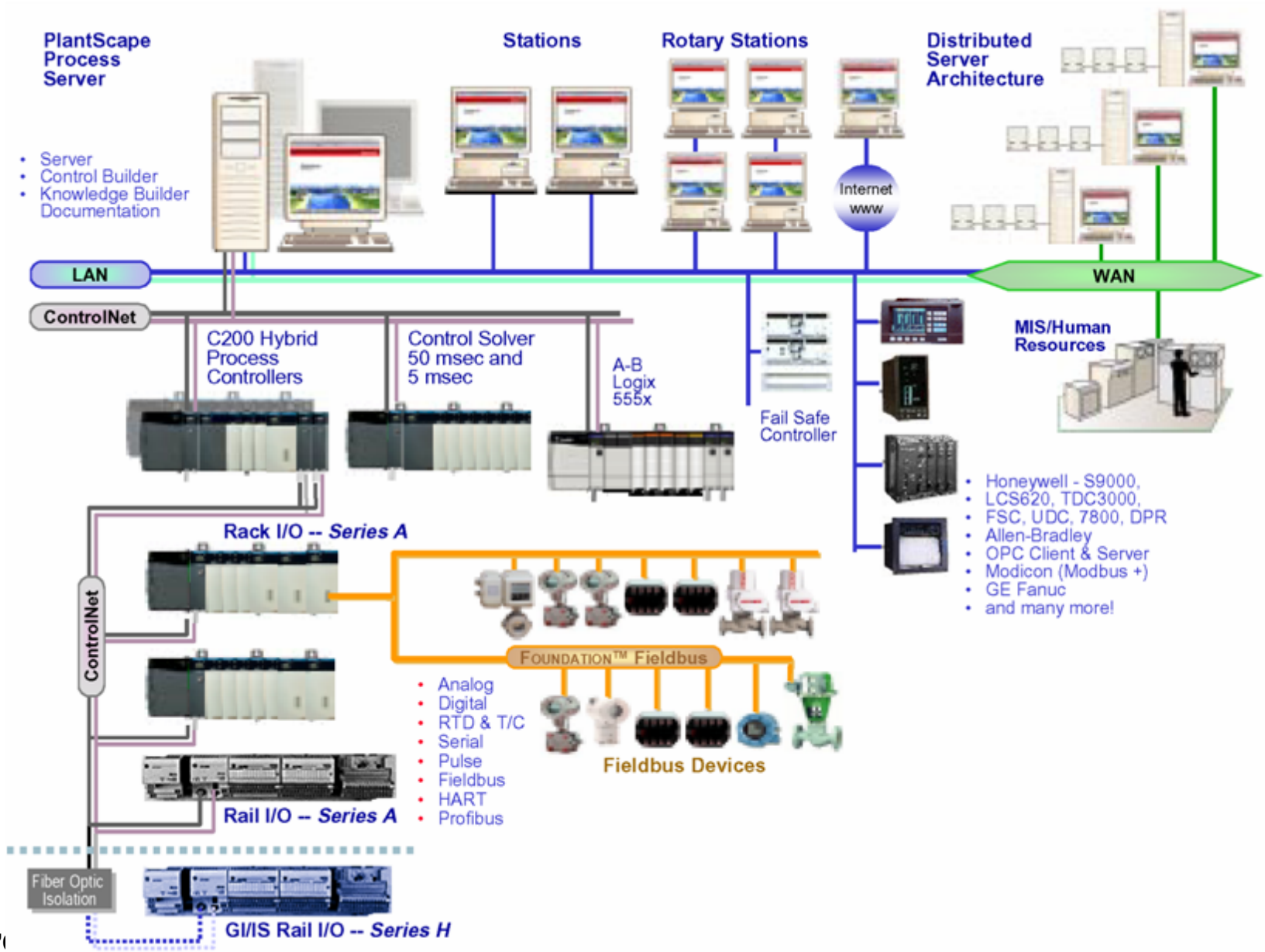
Các thành phần chính

- Khối điều khiển cục bộ (LCU), bộ điều khiển: Chức năng điều khiển cơ sở và giám sát (chủ yếu cho các biến tương tự), có thể cả điều khiển cao cấp
- Vào/ra từ xa (RIO), vào/ra phân tán
- Trạm vận hành: Chức năng giao diện vận hành & giám sát, quản lý dữ liệu
- Trạm kỹ thuật: Phát triển phần mềm, cấu hình và chẩn đoán hệ thống
- Bus hệ thống (system bus, data highway), bus điều khiển
- Tùy chọn: Các loại trạm chủ, các máy tính điều khiển cao cấp, các loại bus trường, bus thiết bị (Foundation Fieldbus, DP, DeviceNet...)

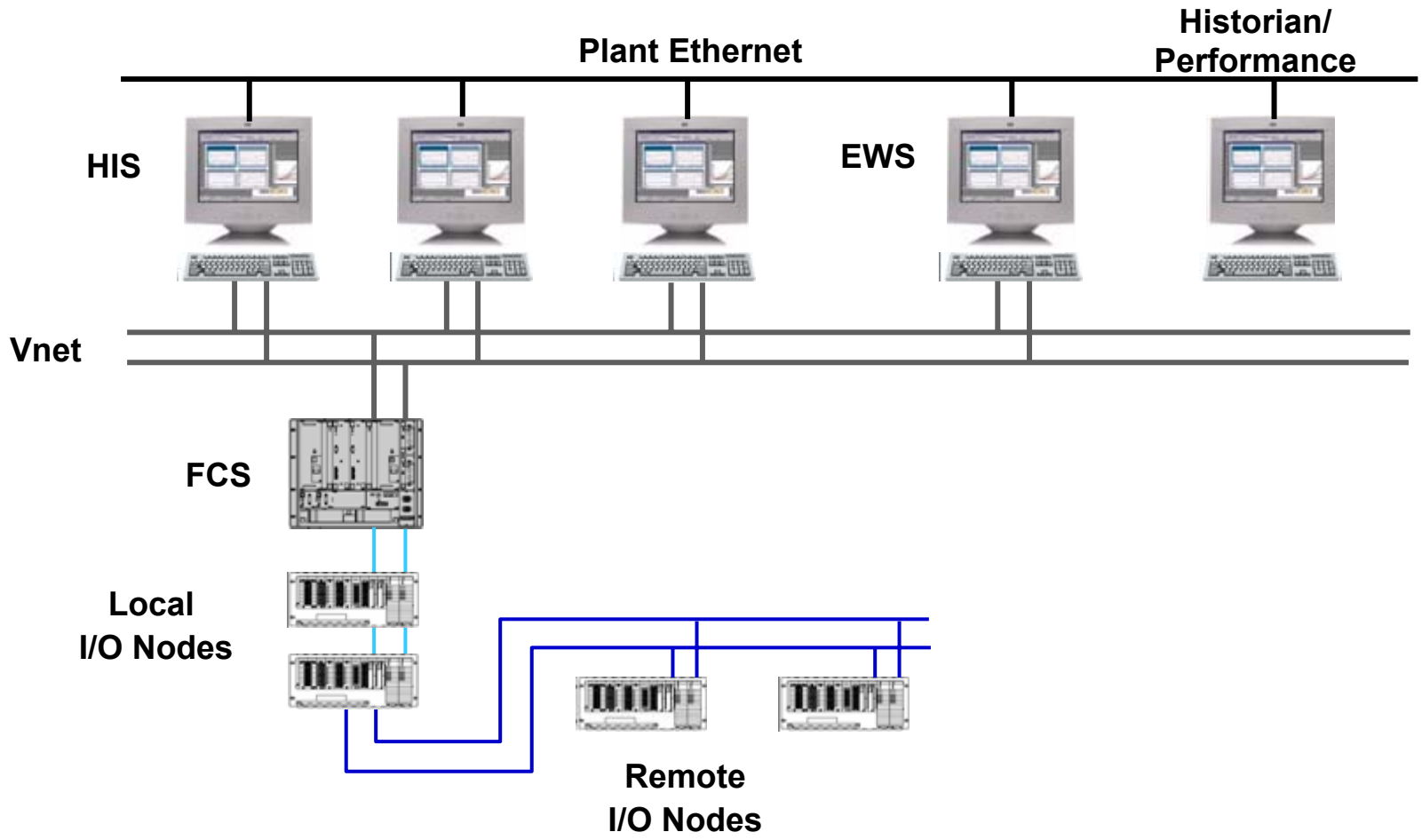
Ví dụ: DeltaV (Fisher-Rosermount)



Ví dụ: PlantScape (Honeywell)



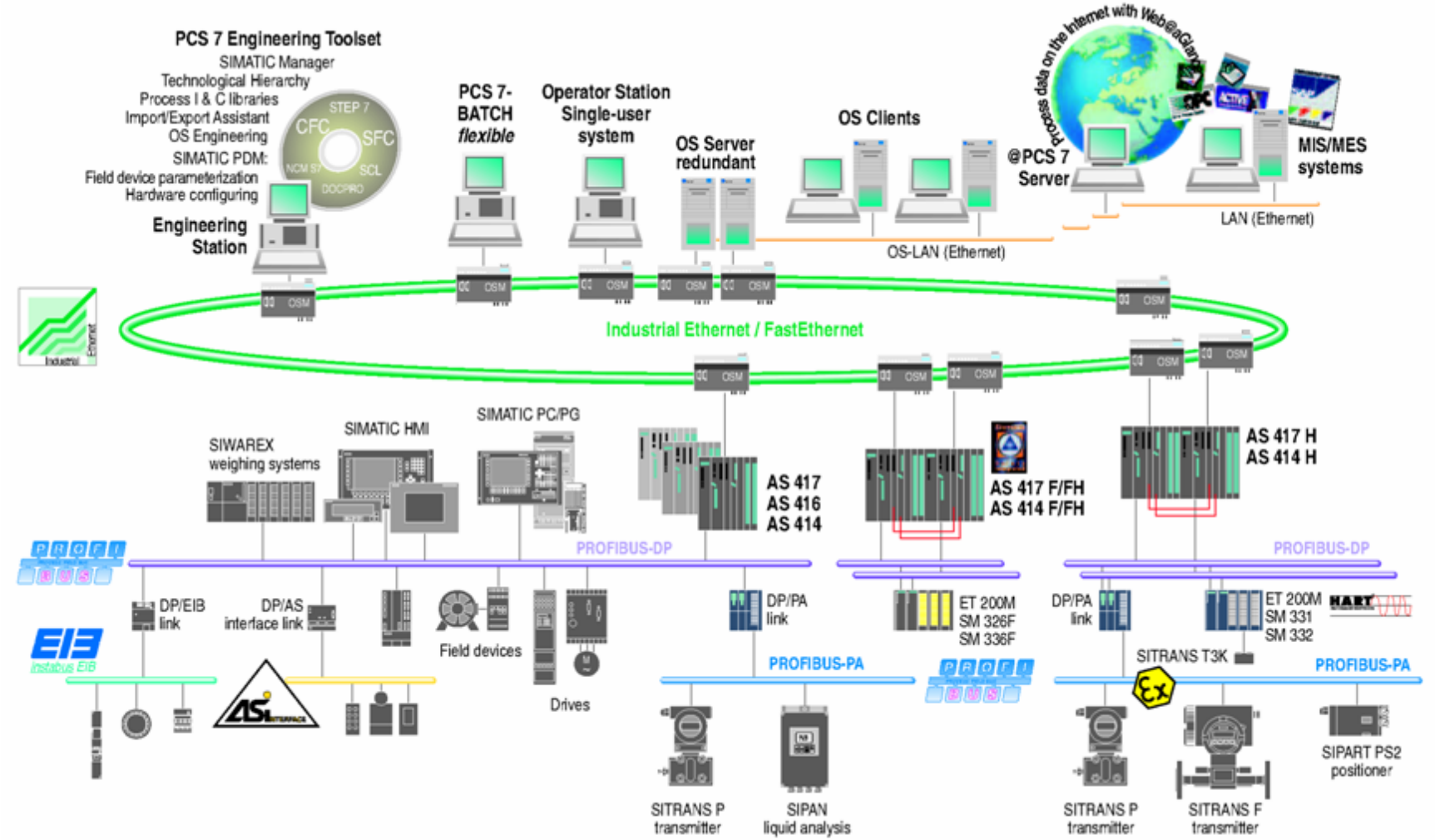
Ví dụ: Centum CS3000 (Yokogawa)



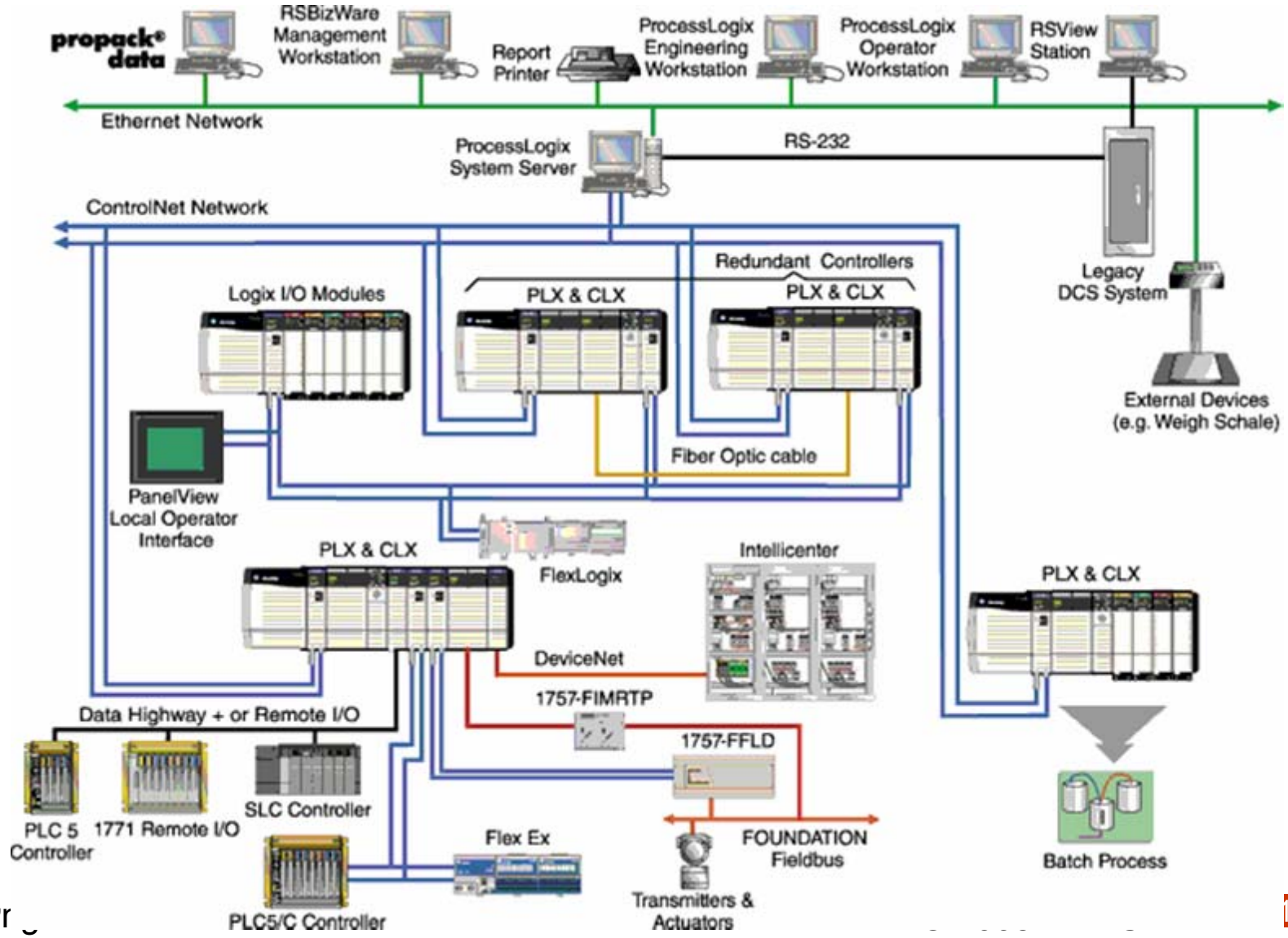
HIS: Human Interface Station

EWS: Engineering Workstation

Ví dụ: PCS7 (Siemens)

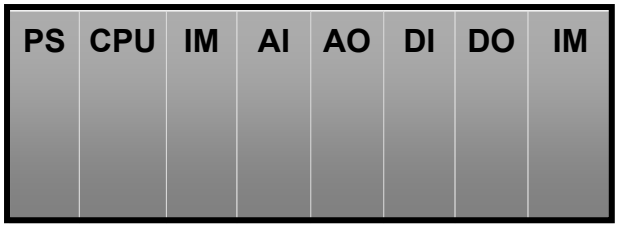


Ví dụ: ProcessLogix (Allen-Bradley)

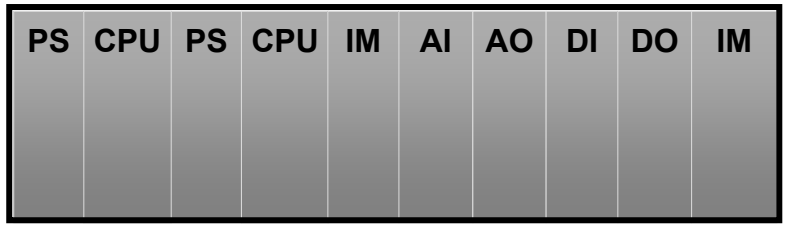


Bộ điều khiển (Controller, LCU)

Không dự phòng



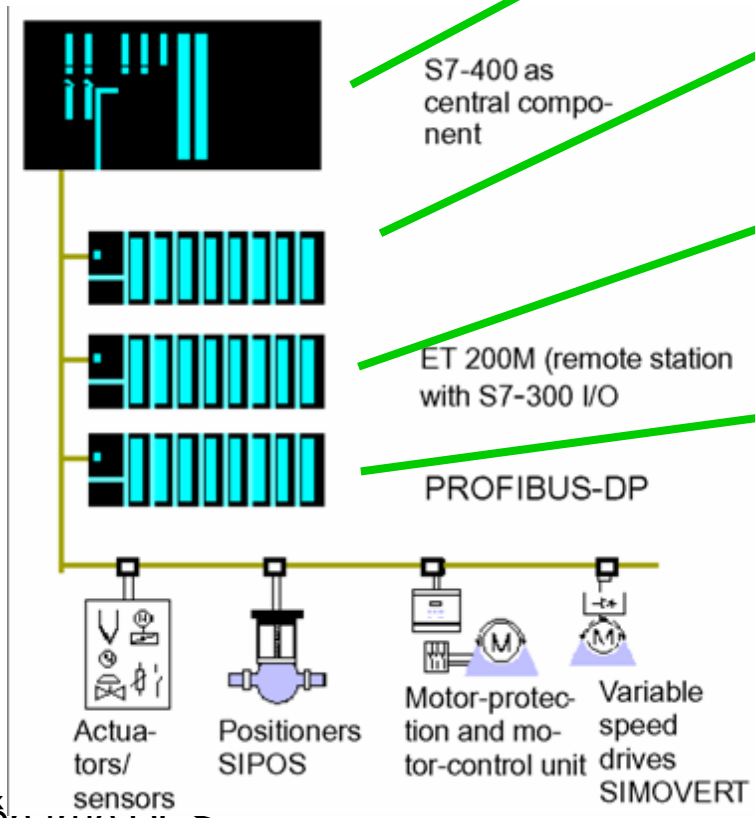
Có dự phòng

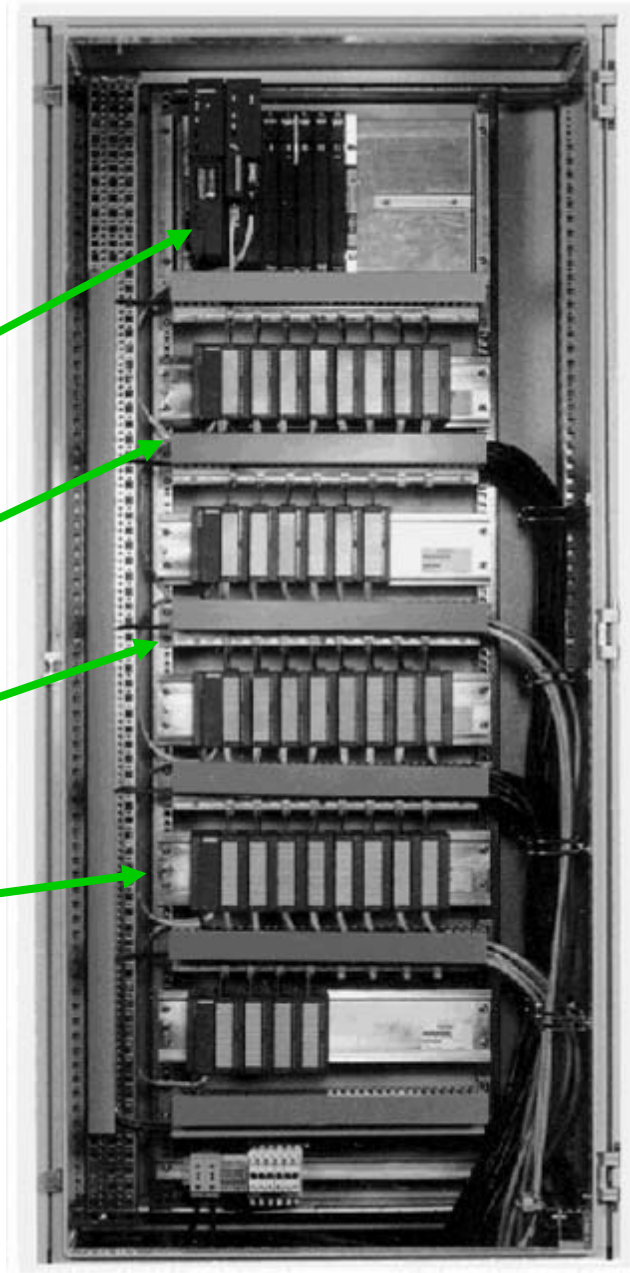


- Cấu trúc module:
  - Giá đỡ, thanh ray
  - Khối cung cấp nguồn (PS)
  - Khối xử lý trung tâm (CPU)
  - Các module vào/ra (thông thường, HART, an toàn cháy nổ)
  - Giao diện với bus hệ thống, bus điều khiển
  - Giao diện với bus trường (tùy chọn)
- Kiến trúc máy tính:
  - Máy tính đặc chủng, chỉ điều khiển liên tục-> DCS truyền thống
  - PLC -> PLC-based DCS
  - PC/IPC -> PC-based DCS
  - Các bộ điều khiển lai

DeltaV Controller

PCS 7 Controller





Các chức năng của bộ điều khiển

- Xử lý và giám sát các giá trị vào/ra:
  - Chuyển đổi D/A, A/D, lọc nhiễu, chuyển thang
  - Giám sát ngưỡng giá trị, tạo các thông báo báo động
  - Lưu trữ tạm thời các giá trị vào/ra
  - Hạn chế tín hiệu ra, đặt các tín hiệu ra về trạng thái an toàn trong trường hợp có sự cố
- Điều khiển:
  - Điều chỉnh tự động: Điều khiển PID, điều khiển tỉ lệ, điều khiển tầng, điều khiển mờ
  - Điều khiển trình tự, điều khiển khóa liên động
  - Điều khiển cao cấp: Điều khiển MPC, điều khiển theo công thức, điều khiển thích nghi
- Tự chẩn đoán
- Xử lý truyền thông

Các tính năng quan trọng của bộ điều khiển

- Tính năng thời gian thực
  - Hiệu năng CPU, dung lượng bộ nhớ
  - Xử lý đa nhiệm, hỗ trợ của hệ điều hành TGT
  - Thời gian chuyển đổi tương tự-số
  - Chu kỳ task (tối thiểu 100ms)
- Khả năng ghép nối vào/ra
  - Các loại tín hiệu vào/ra
  - Vào/ra phân tán
  - Ghép nối thiết bị HART, ghép nối bus trường
- Lập trình thuận tiện, cho phép sử dụng các thuật toán cao cấp
  - Ngôn ngữ theo chuẩn quốc tế IEC 61131-3
  - Khả năng điều khiển lai (liên tục, trình tự và logic)
  - Thư viện khối chức năng mạnh
- Độ tin cậy và tính sẵn sàng
  - Khả năng tự chẩn đoán, tự kiểm tra và sửa lỗi
  - Cơ chế dự phòng nóng
  - Thời gian chuyển mạch khi có sự cố

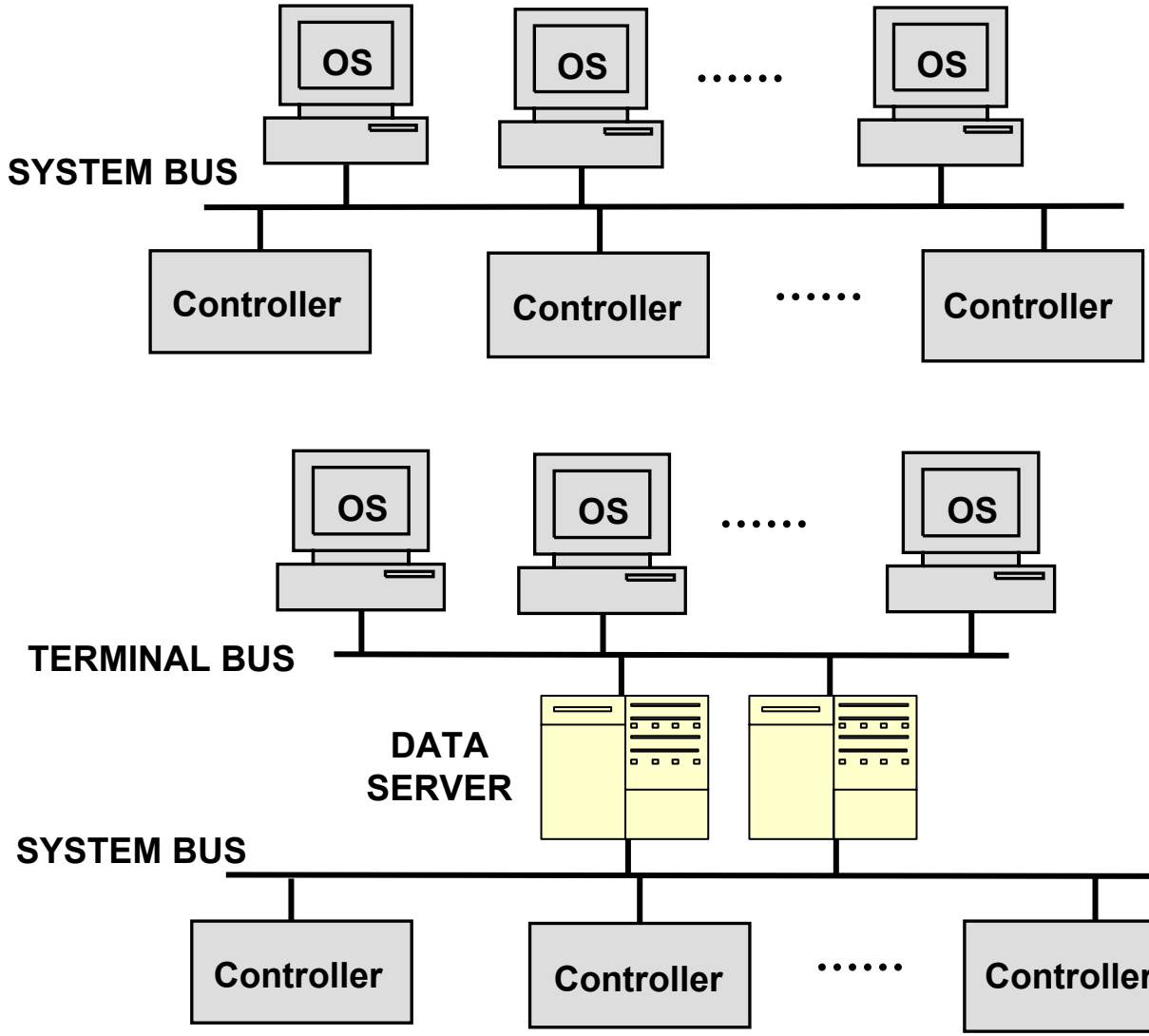
Trạm vận hành (Operator Station)

- Cấu hình tiêu biểu:
  - Máy tính trạm (Workstation) hoặc máy tính cá nhân (PC)
  - Hệ điều hành Windows NT/2000/XP hoặc UNIX, LINUX
  - Màn hình CRT 21", có thể màn hình kép

Các chức năng của trạm vận hành

- 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 hình ảnh đồ họa tự do: lưu đồ công nghệ, các phím điều khiển
  - Các đồ thị thời gian thực và đồ thị quá khứ
- Hỗ trợ vận hành:
  - Các công cụ thao tác tiêu biểu, xử lý mệnh lệnh
  - Hệ thống hướng dẫn chỉ đạo
  - Hướng dẫn trợ giúp
  - Bảo trì hệ thống
- Quản lý dữ liệu quá trình
  - Xử lý, lưu trữ
  - Phục vụ truy vấn
- Xử lý các sự kiện, sự cố
  - Gửi thông báo, yêu cầu xác nhận
  - Bảng hiển thị tóm tắt, bảng hiển thị chi tiết
- Hỗ trợ lập báo cáo tự động

Cấu trúc bố trí các trạm vận hành



Cấu trúc phẳng

Cấu trúc phân cấp

Trạm kỹ thuật (Engineering Station)

- Cấu hình:
  - Máy tính cá nhân (WinNT/2000/XP) hoặc máy tính trạm (UNIX,LINUX), màn hình CRT 19"
  - Thường có thể sử dụng là trạm vận hành
  - Có cơ chế khóa kiểm soát quyền sử dụng
- Các chức năng tiêu biểu:
  - Đặt cấu hình hệ thống, lưu trữ và quản lý dữ liệu cấu hình hệ thống
  - Tham số hóa, đưa các thiết bị trường vào vận hành
  - Lập trình điều khiển (LD, FBD/CFC, SFC, C/C++/BASIC,...)
  - Thử nghiệm và gỡ rối chương trình
  - Tạo giao diện người-máy và các chức năng điều khiển giám sát khác
  - Quan sát và chẩn đoán lỗi hệ thống

Ví dụ bố trí phòng điều khiển trung tâm

Bus hệ thống

- Chức năng:
  - Nối mạng các bộ điều khiển với các trạm vận hành/trạm kỹ thuật
  - Nối mạng các bộ điều khiển với nhau (bus điều khiển)
- Đặc điểm kỹ thuật:
  - Mạng tốc độ cao (10-100MBit/s)
  - Yêu cầu tính năng thời gian thực, đặc biệt với bus điều khiển
  - Độ tin cậy cao, thường có dự phòng 1-1
- Các loại mạng công nghiệp thường được sử dụng:
  - Ethernet và Industrial Ethernet (sử dụng Switches/Hub)
  - Profibus-FMS
  - ControlNet
  - Data Highway
  - Mạng đặc chủng của riêng hãng sản xuất

Bus trường

- Chức năng:
  - Ghép nối trạm điều khiển với các trạm vào/ra phân tán
  - Ghép nối trạm điều khiển trực tiếp với các thiết bị trường thông minh
- Đặc điểm kỹ thuật:
  - Mạng tốc độ thấp hoặc vừa phải (< 10Mbit/s)
  - Yêu cầu tính năng thời gian thực cao
  - Độ tin cậy cao, đặc biệt trong môi trường dễ cháy nổ
- Các loại bus thường được sử dụng:
  - Profibus-DP, Profibus-PA
  - Foundation Fieldbus H1
  - Controlnet
  - DeviceNet
  - Mạng đặc chủng của riêng hãng sản xuất, sử dụng RS-485

4.3 Các điểm mấu chốt của kiến trúc DCS

- Tích hợp toàn diện về phần cứng: Bộ điều khiển, I/O, ES, OS
- Tích hợp toàn diện về phần mềm
  - Các phần mềm chạy
  - Môi trường phát triển tích hợp (IDE), xuyên suốt
  - Cơ sở dữ liệu cấu hình chung
  - Cơ sở dữ liệu quá trình chung
- Tích hợp toàn diện về truyền thông
  - Cơ sở hạ tầng truyền thông
  - Giao tiếp ngầm (không cần cấu hình, lập trình riêng)
- Tích hợp toàn diện về chức năng
  - Điều khiển cơ sở, điều khiển cao cấp
  - Vận hành & giám sát,...

Kiến trúc DCS "truyền thống"

- Kiến trúc "đóng kín"
  - Hỗ trợ ít các chuẩn công nghiệp
  - Phương pháp lập trình riêng
  - Khó tích hợp sản phẩm hãng thứ 3 (phần cứng, phần mềm)
- Bộ điều khiển:
  - Chuyên dụng, đặc chủng
  - Chỉ điều khiển các quá trình liên tục hoặc theo mẻ
- Giao diện quá trình:
  - 4-20mA, HART
  - Vào/ra tập trung hoặc từ xa, không dùng bus chuẩn hóa
- Phân tán chưa triệt để (chức năng điều khiển vẫn chỉ nằm ở các bộ điều khiển)

Kiến trúc DCS hiện đại

- Kiến trúc mở:
  - Hỗ trợ nhiều các chuẩn giao tiếp công nghiệp (COM, OPC, Ethernet, ODBC, ActiveX, MMS, XML)
  - Phương pháp lập trình theo chuẩn IEC 61131-3
  - Có thể tích hợp sản phẩm hãng thứ 3 (PLC, I/O, biến tần, MES, ERP,..)
- Bộ điều khiển:
  - Đa dạng
  - Khả năng điều khiển lai
- Giao diện quá trình:
  - 4-20mA, HART
  - Vào/ra tập trung hoặc từ xa qua bus chuẩn hóa
  - Ghép nối trực tiếp với các thiết bị bus trường
- Chức năng điều khiển: có thể đưa xuống cấp trường

4.4 So sánh DCS và PLC/HMI

- Ngày càng giống nhau về:
  - Cấu trúc phần cứng
  - Phạm vi chức năng
  - Ngôn ngữ lập trình
- Khác nhau cơ bản: mức độ tích hợp
  - DCS: Phần cứng trọn gói, môi trường phát triển tích hợp (điều khiển cơ sở, điều khiển cao cấp, HMI, ...), cơ sở dữ liệu toàn cục
  - PLC/HMI: Cấu hình phần cứng tương đối tự do, công cụ lập trình PLC và công cụ SCADA/HMI độc lập với nhau, cơ sở dữ liệu độc lập => phức tạp hơn

⇨ **DCS ≠ PLC+HMI**

DCS hay PLC/HMI?

Đầu tư ban đầu:	DCS	**PLC/HMI**
Mức độ tích hợp:	**DCS**	PLC/HMI
Công cụ phát triển:	**DCS**	PLC/HMI
Độ tin cậy, sẵn sàng:	**DCS**	PLC/HMI
Qui mô ứng dụng lớn:	**DCS**	PLC/HMI
Quá trình liên tục:	**DCS**	PLC/HMI
Quá trình rời rạc:	DCS	**PLC/HMI**
Quá trình hỗn hợp:	**DCS + PLC** hoặc **hệ lai**	

Hệ thống điều khiển phân tán

Chương 5: Kiến trúc PC-based Control

8/27/2005

Chương 5: Kiến trúc PC-based Control

5.1 Tại sao sử dụng giải pháp PC

5.2 Các vấn đề cơ bản của giải pháp PC

5.3 Cấu hình cơ bản một hệ PC-based Control

Các loại giải pháp khác nhau

5.4 Hệ điều khiển phân tán trên nền PC

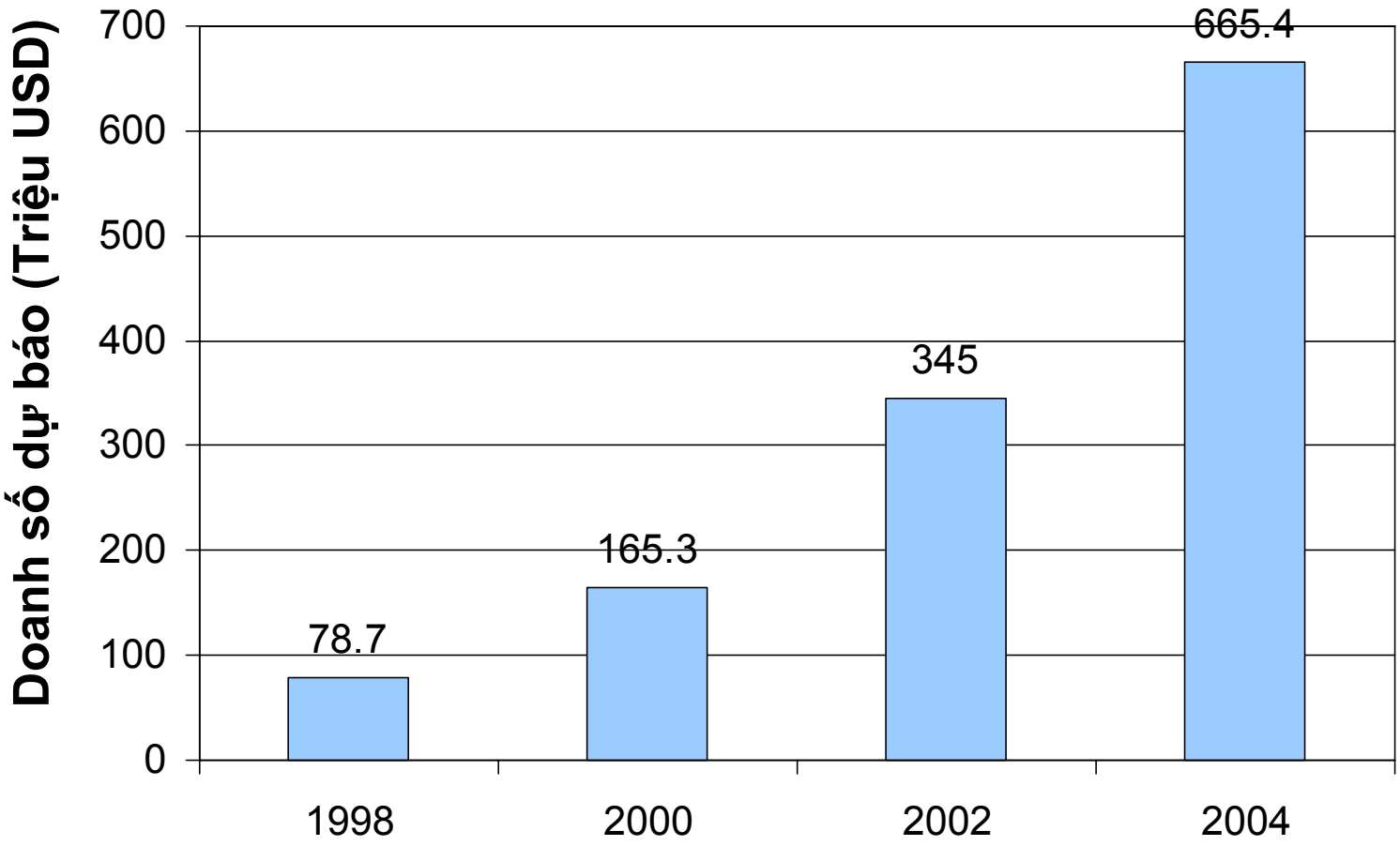
5.5 Các điểm mấu chốt trong kiến trúc PC-based Control

5.1 Tại sao sử dụng giải pháp IPC?

- Hiệu năng tính toán cao với giá thành thấp, chu kỳ điều khiển có thể xuống tới 1ms
- Kiến trúc máy tính phổ thông, quen thuộc
- Tính năng mở
  - Hệ điều hành thông dụng
  - Nối mạng đơn giản
  - Lập trình tự do, công cụ lập trình mạnh
  - Sử dụng các thành phần chuẩn (*off-the-shelf components*) => *component-based system* (khác với *integrated system*)
- Có thể kết hợp các chức năng điều khiển cơ sở, điều khiển cao cấp và vận hành-giám sát (*all-in-one system*),
- Dễ dàng ghép nối với các ứng dụng cấp trên
- Độ tin cậy ngày càng được cải thiện
- Ghép nối vào/ra đơn giản qua bus trường

Thị trường PC-based Control ở châu Âu

(Theo *Computerzeitung 5/1998*)



Các phạm vi ứng dụng tiêu biểu

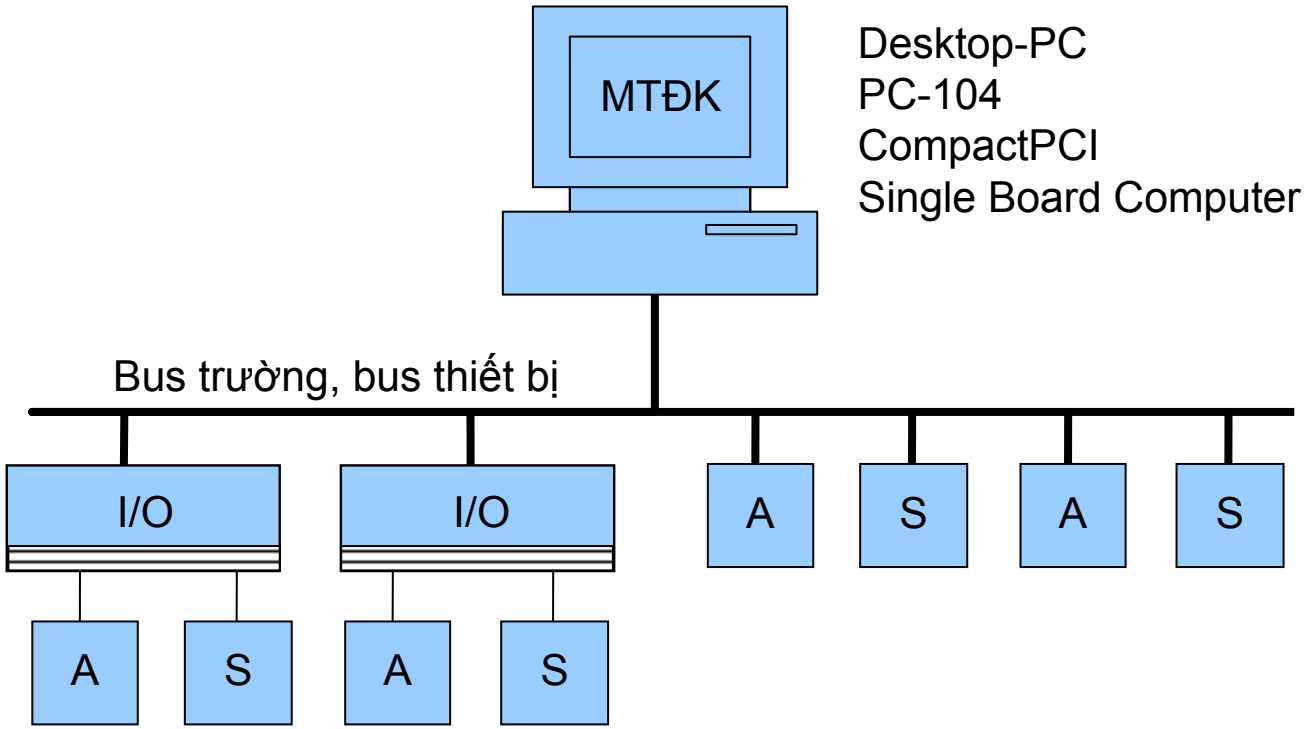
- Các dây chuyền chế tạo, lắp ráp, đóng bao:
  - Thay thế giải pháp PLC truyền thống => Slot-PLC, Soft-PLC
  - Kết hợp chức năng điều khiển và vận hành-giám sát tại chỗ
  - Độ tin cậy không phải là vấn đề đáng lo ngại
- Điều khiển chuyển động:
  - Thay thế giải pháp CNC truyền thống => Soft-CNC
  - Điều khiển tay máy
- Điều khiển một nhóm thiết bị, máy móc đơn lẻ
- Điều khiển quá trình: công nghệ thực phẩm, dược phẩm, xử lý nước sạch, nước thải, CN bán dẫn,...
- SCADA

⇨ **Khả năng xử lý nhanh, hỗn hợp, linh hoạt, dễ tích hợp HMI và các chức năng cao cấp**

5.2 Các vấn đề cơ bản của giải pháp PC

- Nâng cao độ tin cậy:
  - Sử dụng các chủng loại PC công nghiệp hoặc ít ra phải là PC có thương hiệu tin cậy
  - Nếu có thể, nên sử dụng FlashROM thay cho đĩa cứng
  - Cần hệ điều hành tốt, hoạt động ổn định
  - Loại trừ hoàn toàn các chương trình ứng dụng khác
  - Cần giải pháp dự phòng nóng trong trường hợp cần thiết
- Đảm bảo tính năng thời gian thực:
  - Hệ điều hành thời gian thực hoặc ít ra là HĐH đa nhiệm có đáp ứng phần cơ bản về tính năng thời gian thực (quan trọng nhất: chu kỳ điều khiển và độ rung, jitter)
- Lập trình thuận tiện
  - Nếu dùng ngôn ngữ bậc cao: cần thư viện mạnh, dễ sử dụng
  - Tốt hơn hết: công cụ lập trình trực quan + phần mềm khung

5.3 Cấu hình cơ bản

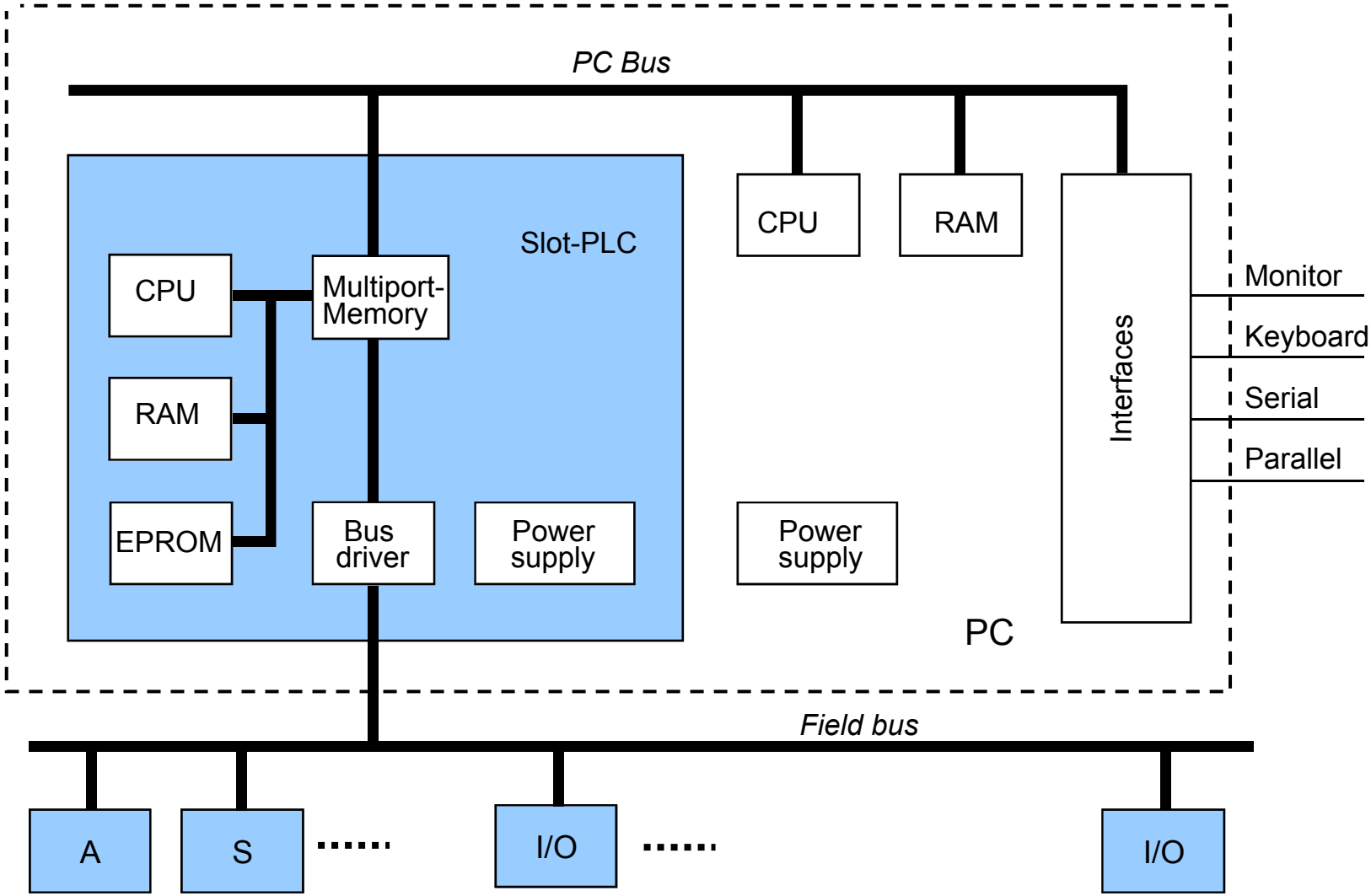


➲ *Bắt buộc sử dụng vào/ra từ xa hoặc thiết bị bus trường*

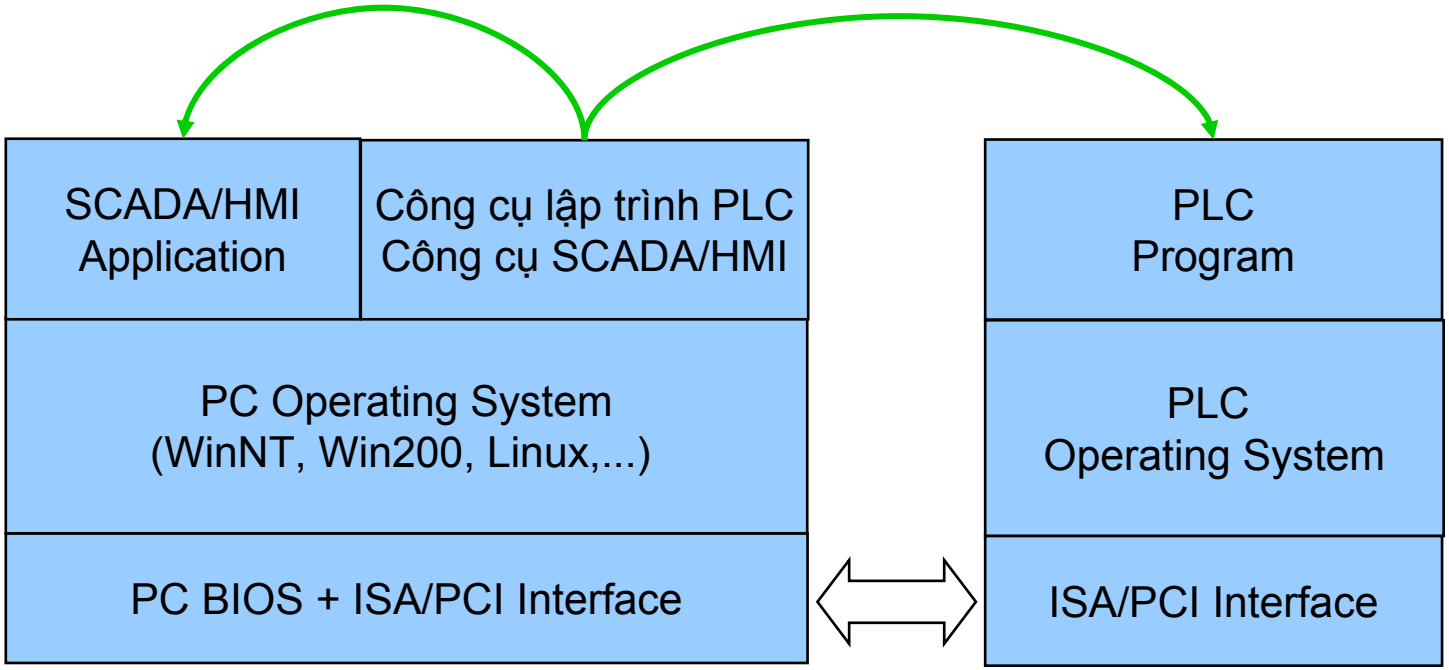
Slot-PLC, Embedded PLC

- PC + PLC dưới dạng một card ISA/PCI
- PLC cho điều khiển, PC cho lập trình & vận hành-giám sát
- PLC hoạt động độc lập, chỉ sử dụng nguồn cấp từ PC
- PLC được cài đặt hệ điều hành TGT
- Lập trình hoàn toàn tương tự như cho PLC thông thường
- Giao tiếp PC <=> PLC đơn giản qua bus PCI/ISA
- Ưu điểm: gọn nhẹ, tương đối tin cậy
- Nhược điểm:
  - Chưa lợi dụng được thế mạnh thực sự của PC
  - Ít có sự lựa chọn các khối vào/ra

Cấu trúc phần cứng Slot-PLC



Mô hình phần mềm Slot-PLC



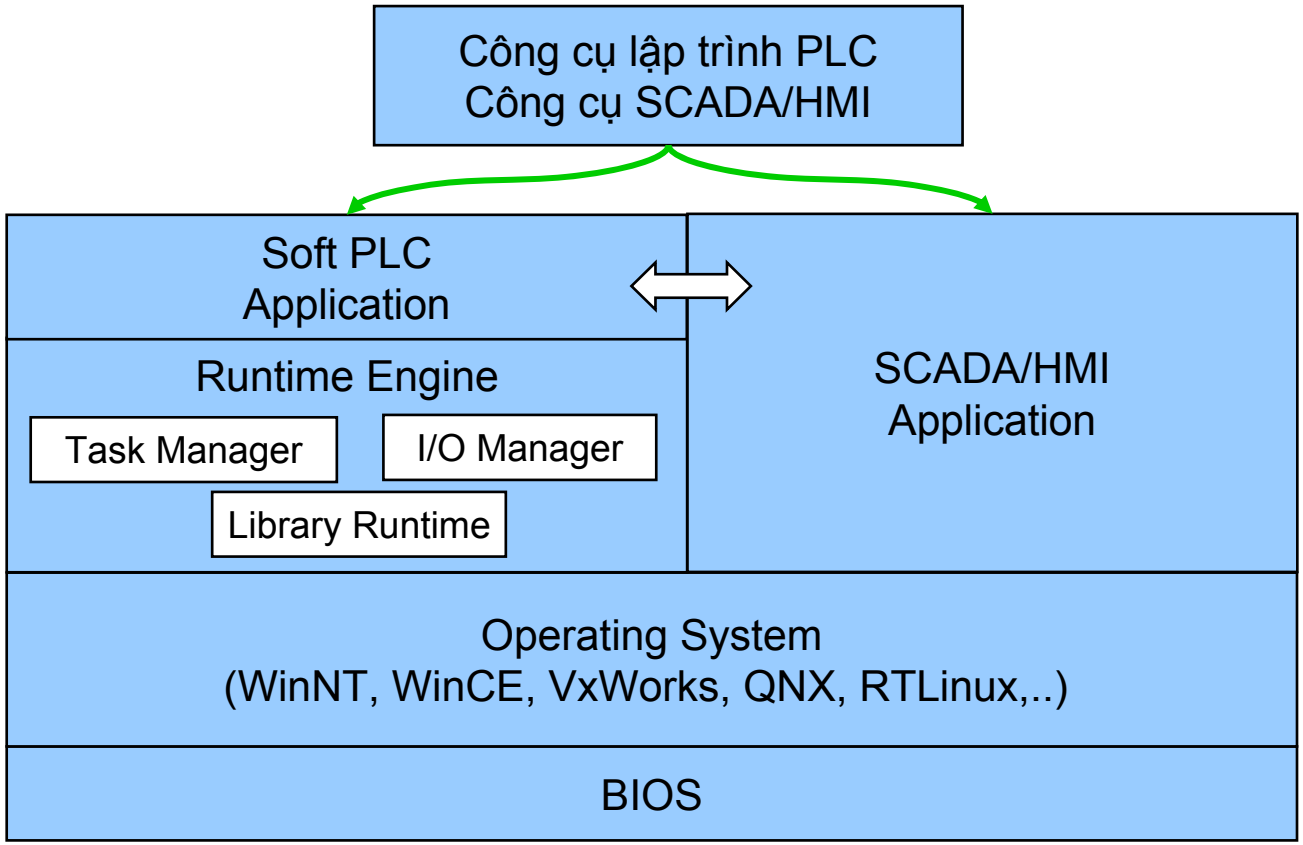
Ví dụ sản phẩm:

- Phoenix Contact: PC WORX (sử dụng Interbus)
- Siemens: WinAC, Component-based Automation

Soft-PLC, SoftLogic

- PC thực hiện với vai trò như một PLC
- Yêu cầu phần mềm chạy (*PLC runtime engine*)
- Mô hình lập trình hoàn toàn tương tự như cho PLC thông thường
- Có thể tích hợp chức năng ĐK cao cấp, vận hành-giám sát
- Ưu điểm: gọn nhẹ, rẻ
- Nhược điểm: độ tin cậy phụ thuộc vào PC

Mô hình phần mềm Soft-PLC

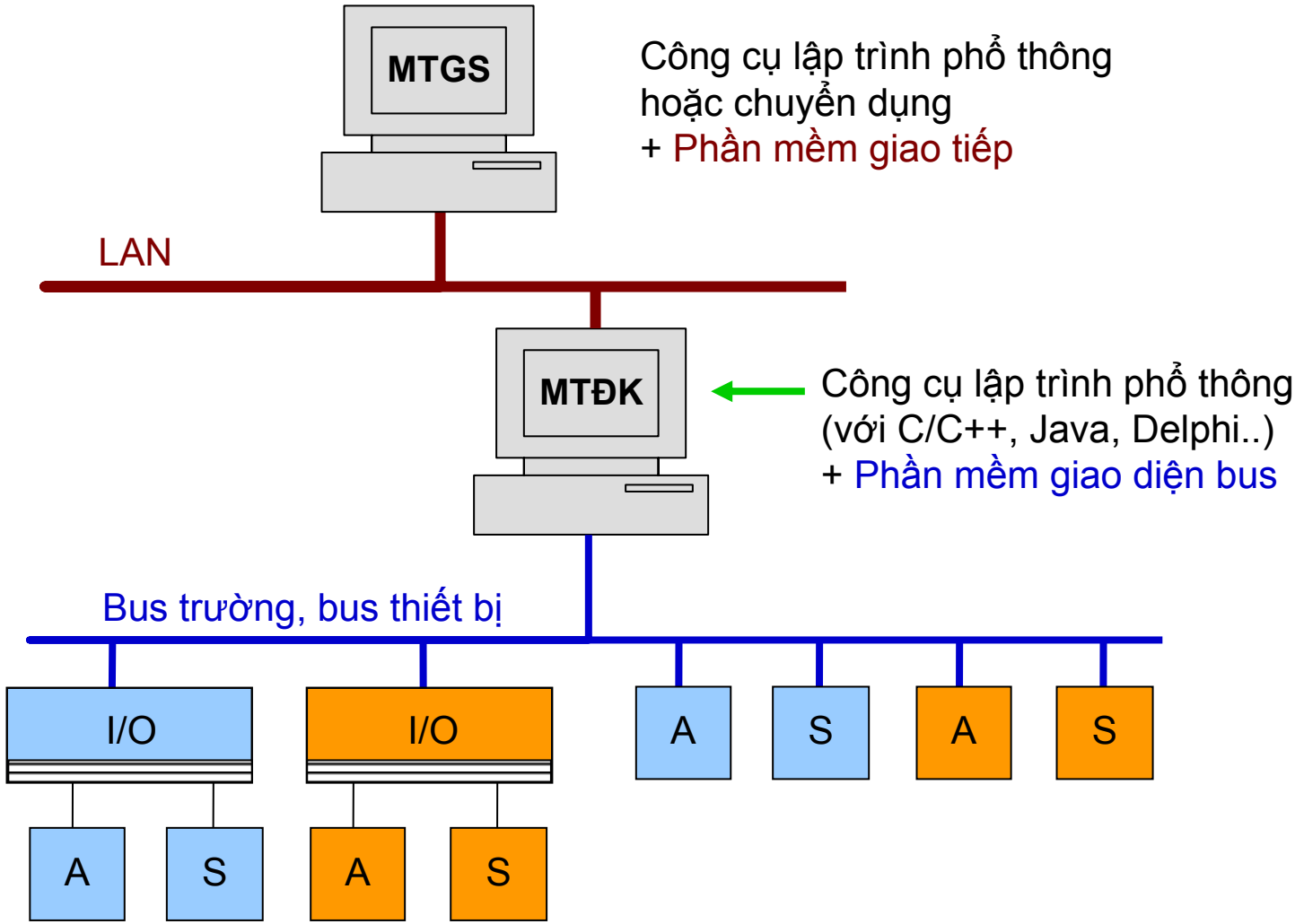


Ví dụ sản phẩm:

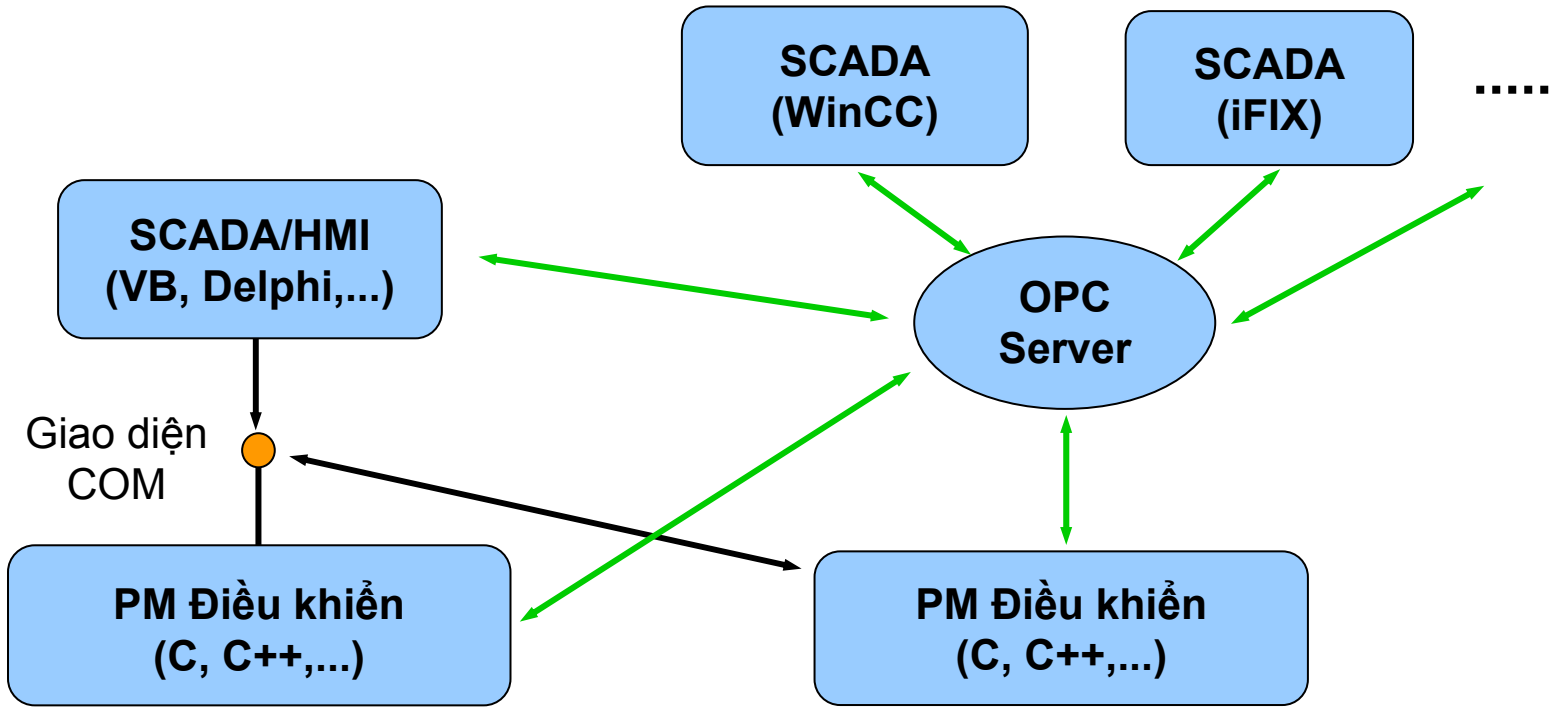
– Softing: 4Control (nhiều loại bus trường)

– Siemens: WinLC

Mô hình giải pháp tự do



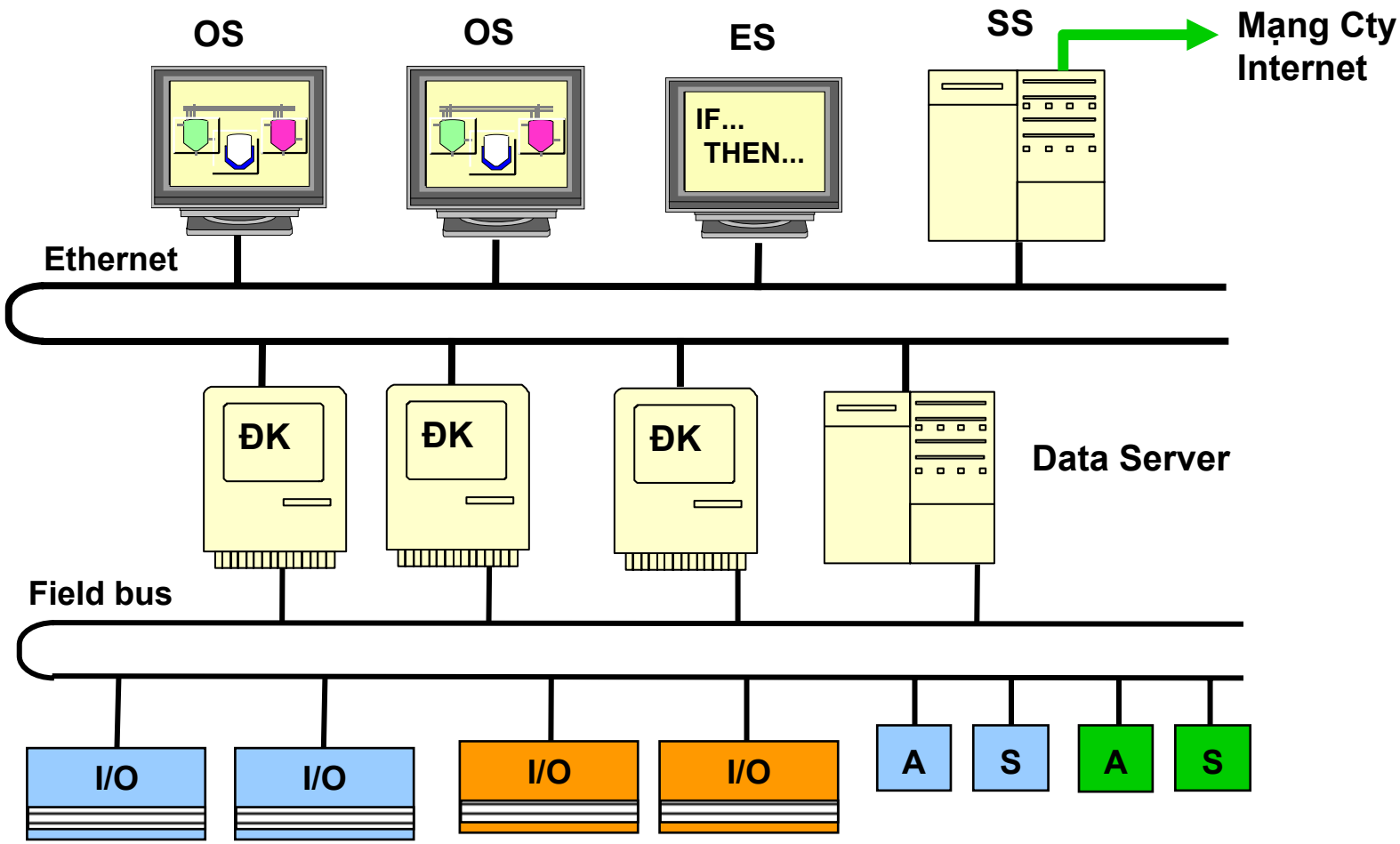
Mô hình giao tiếp qua COM và OPC



Giao diện COM thông thường:
Hiệu suất cao
Khó tích hợp các công cụ chuyên dụng

Giao diện OPC:
Hiệu suất khá cao
Đa năng

5.4 Hệ điều khiển phân tán trên nền PC



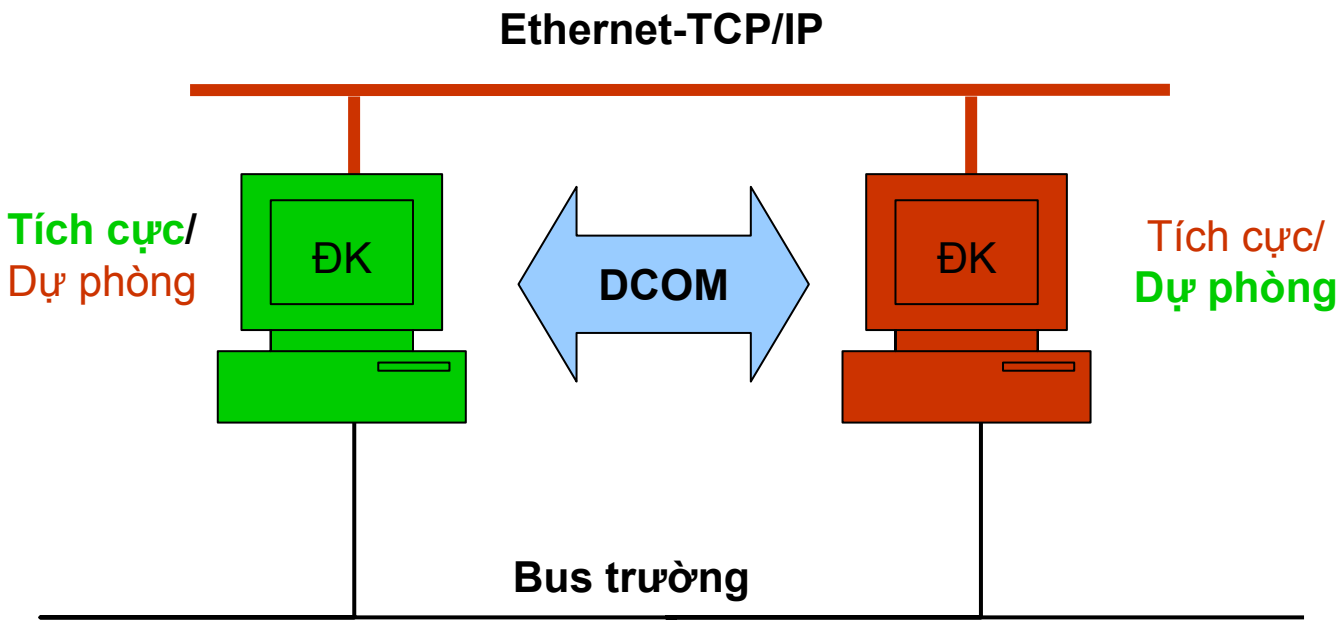
Máy tính điều khiển

- Cấu hình phần cứng tiêu biểu:
  - CPU: Pentium XX, RAM: > 64 MB
  - Không cần màn hình
  - Đĩa cứng hoặc FlashROM
  - Giao diện bus trường (DP, FF, DeviceNet,...)
  - Giao diện LAN
- Cấu hình phần mềm tiêu biểu
  - Hệ điều hành: WinCE/NT/2000, VxWorks, QNX, RTLinux
  - Control Runtime: Quản lý tác vụ, vào/ra, chẩn đoán, thư viện chức năng,...
  - Phần mềm giao tiếp: COM/OPC Server
- Phương pháp lập trình
  - Công cụ chuyên dụng theo IEC 61131-3
  - Có thể sử dụng bổ sung: C/C++, Java,..

Trạm vận hành/Trạm kỹ thuật

- Cấu hình phần cứng tiêu biểu:
  - CPU: Pentium IV, RAM: > 256 MB
  - Màn hình 21" (x 2) cho OS và 19" cho ES
  - Dung lượng ổ cứng: > 40GB
  - Giao diện Fast Ethernet
- Cấu hình phần mềm tiêu biểu
  - Hệ điều hành: NT/2000/XP
  - SCADA Runtime
  - COM/OPC Client
  - Đối với ES: Công cụ lập trình, công cụ SCADA,...
- Phương pháp tạo ứng dụng
  - Công cụ SCADA/HMI chuyên dụng, độc lập
  - Có thể sử dụng bổ sung: C/C++, Java,..

Dự phòng máy tính điều khiển



5.5 Các điểm mấu chốt của kiến trúc PC-based Control

- Kiến trúc hệ thống:
  - Mở, xây dựng trên cơ sở các thành phần chuẩn hóa, *off-the-shelf-components*
  - Điều khiển phân tán hoặc tập trung đều phù hợp
  - Chức năng điều khiển chủ yếu trên PC
  - Giao tiếp qua các chuẩn công nghiệp
- Phát triển hệ thống: Thông thường riêng biệt cho từng phần (trừ giải pháp PC-based DCS)
- Giao diện quá trình:
  - Chủ yếu dựa trên công nghệ bus trường (vào/ra từ xa hoặc sử dụng trực tiếp thiết bị bus trường
  - Với PC có cấu trúc module (ví dụ PC-104) có thể sử dụng vào/ra tập trung cho ứng dụng qui mô nhỏ

DCS, PLC hay PC?

	DCS	PLC	PC
Qui mô ứng dụng	vừa/lớn	vừa/nhỏ	vừa/nhỏ
Thời gian	>100ms	> 20ms	> 1ms
Điều khiển	liên tục	rời rạc	lai
Tính sẵn sàng	++	+	+/o
Giá thành	cao	vừa phải	vừa phải
Phát triển	++	+	+/++
Tính năng mở	o	o	++
Chủ động	o	+	++

Hệ thống điều khiển phân tán

Chương 6: Kiến trúc FCS

8/24/2006

Chương 6: Kiến trúc FCS

6.1 Các vấn đề của kiến trúc DCS/PLC

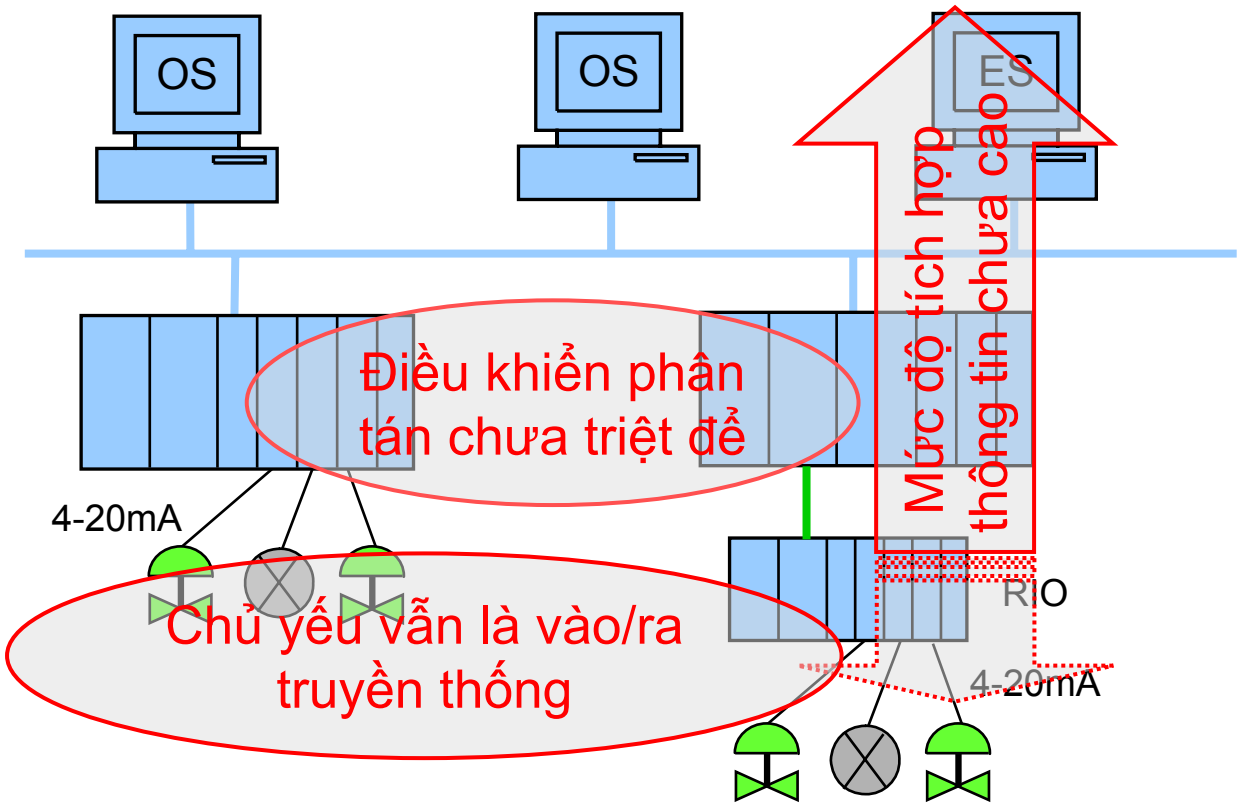
6.2 Cấu trúc hệ thống

6.3 Phân bố chức năng điều khiển

6.4 Phát triển hệ thống

6.5 Tóm tắt các ưu điểm chính

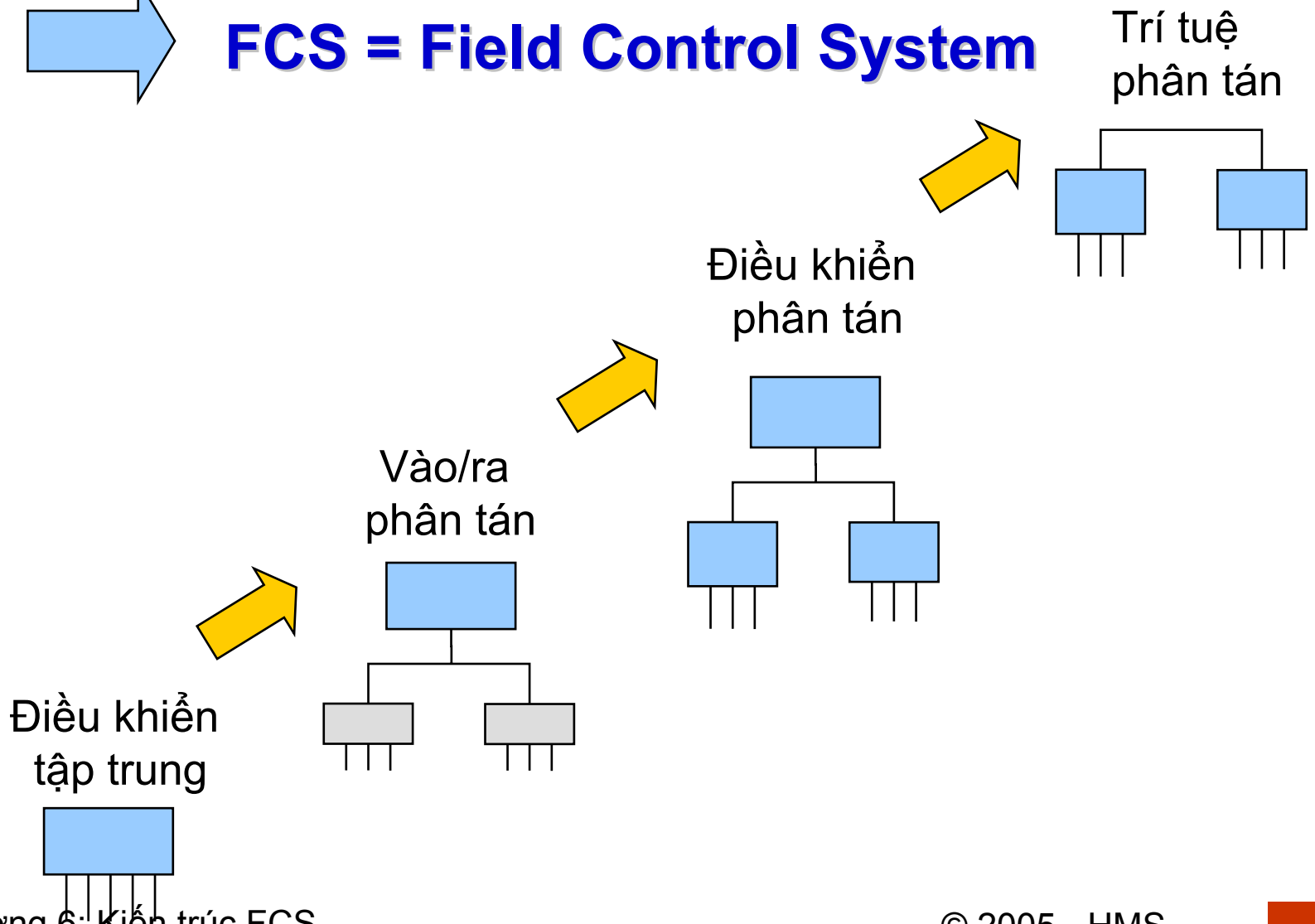
6.1 Các vấn đề của kiến trúc DCS/PLC



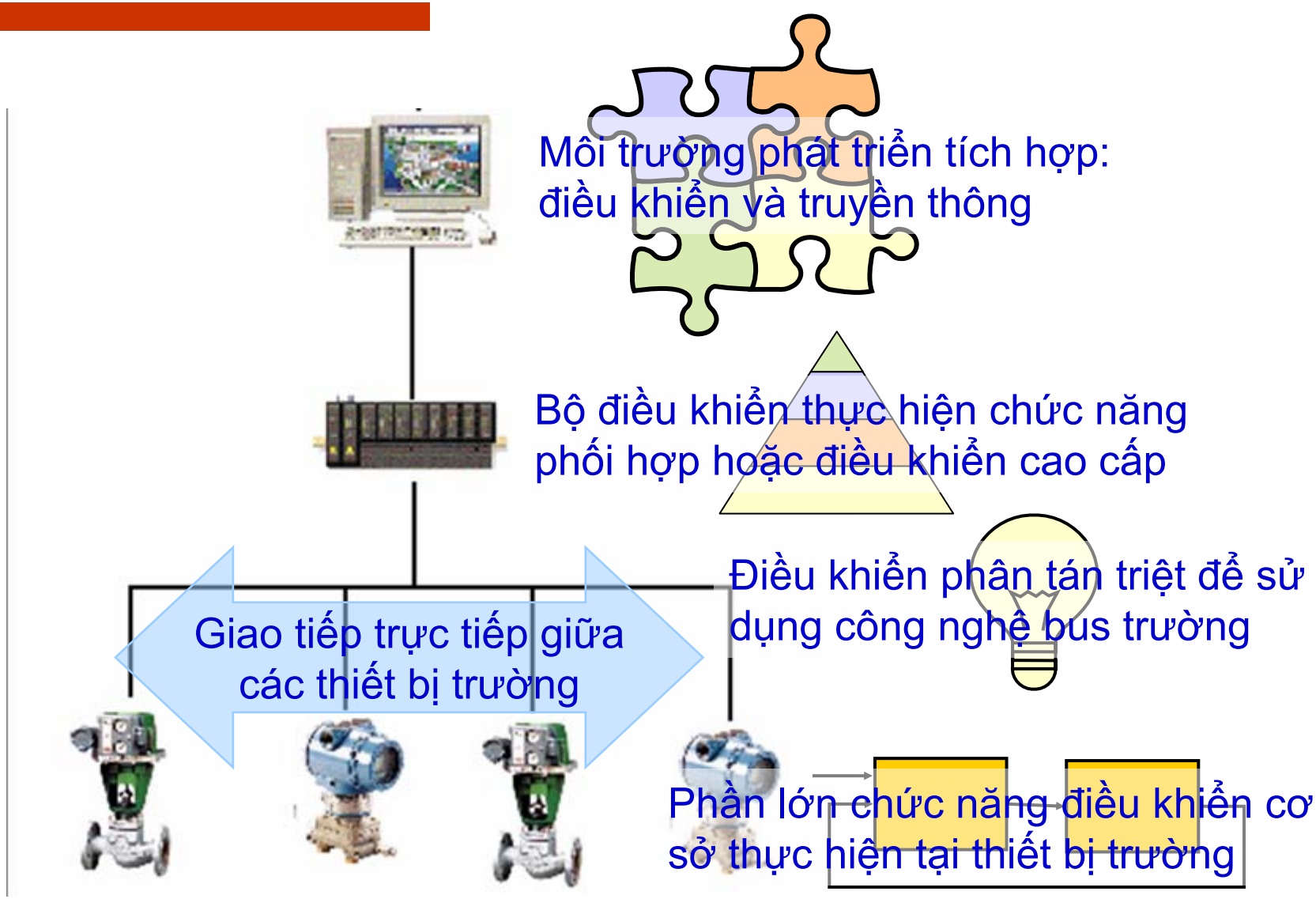
- Điều khiển phân tán chưa triệt để
  - Chức năng điều khiển vẫn tập trung ở bộ điều khiển
  - Hiệu năng của hệ thống chưa cao
  - Kém linh hoạt trong thay đổi chương trình
- Giao diện quá trình chủ yếu vẫn là nối dây truyền thống
  - Sử dụng I/O, I/O termination
  - Tốn cáp truyền, công lắp đặt
  - Tốn thời gian đưa vào vận hành
- Mức độ tích hợp thông tin chưa cao
  - Thông tin về giá trị đo còn nghèo nàn
  - Khả năng tham số hóa và chẩn đoán thiết bị trường hạn chế
  - Khó khăn trong tích hợp khả năng bảo trì phòng ngừa

Các mức của sự phân tán

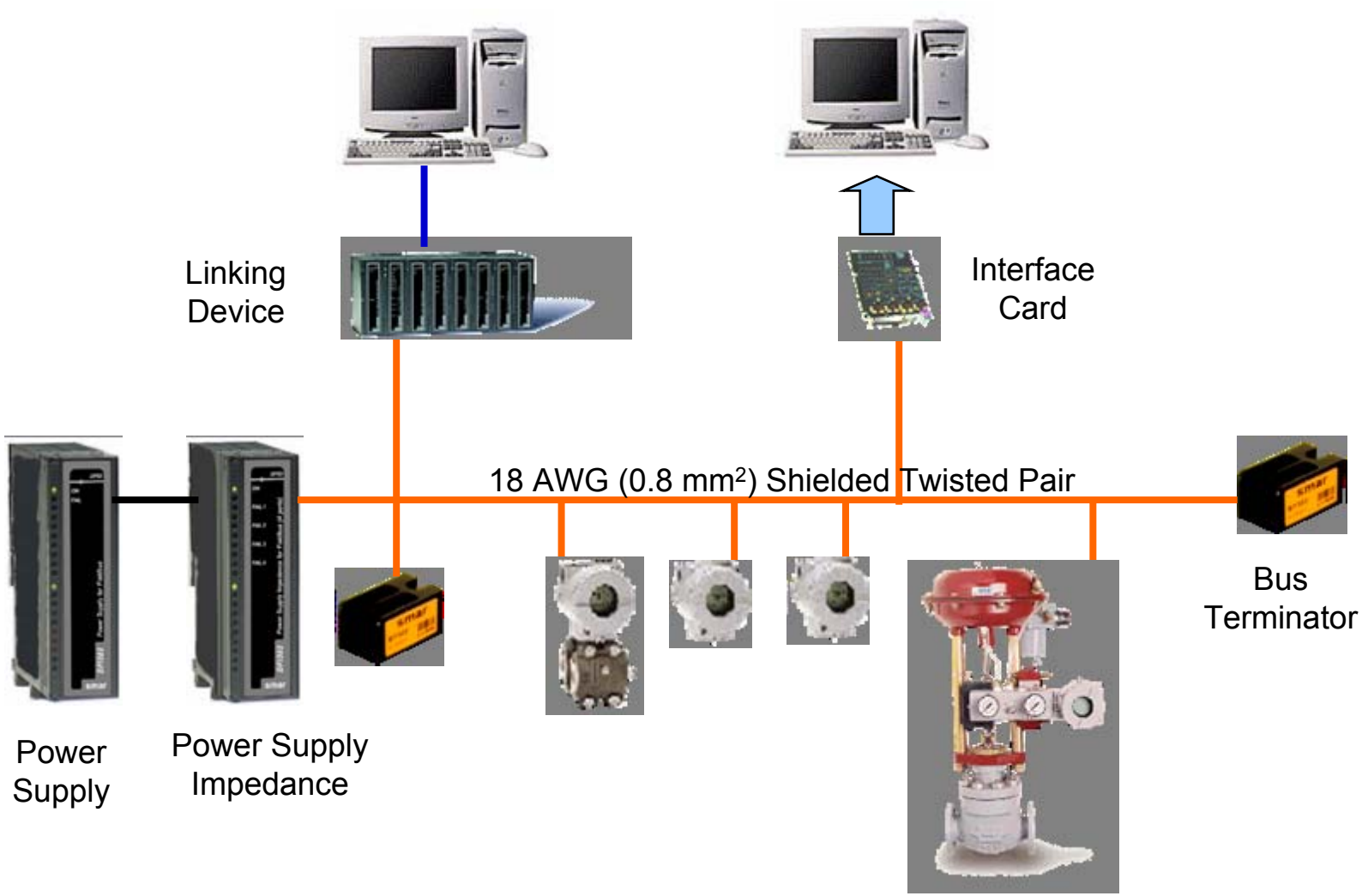
FCS = Field Control System



6.2 Cấu trúc hệ thống

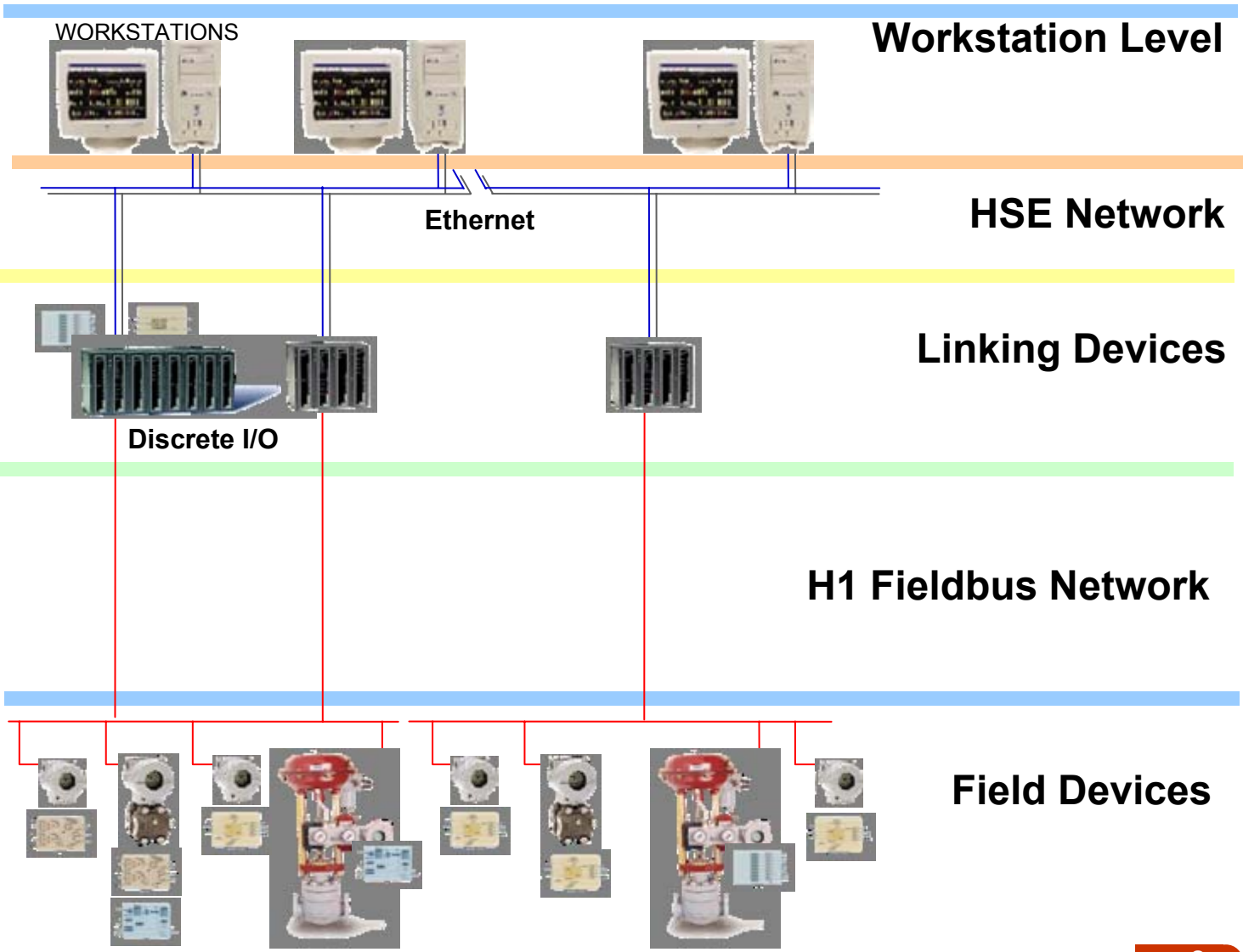


Cấu hình tiêu biểu với Foundation Fieldbus

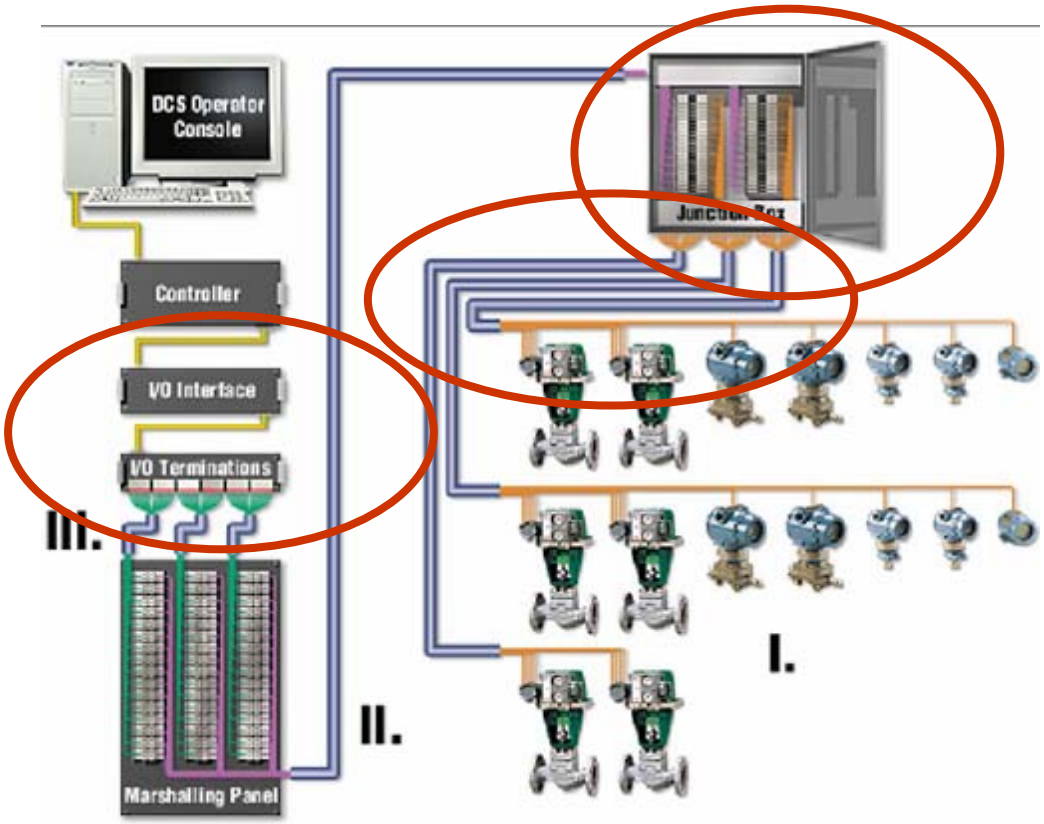


Cấu trúc phân cấp thiết bị

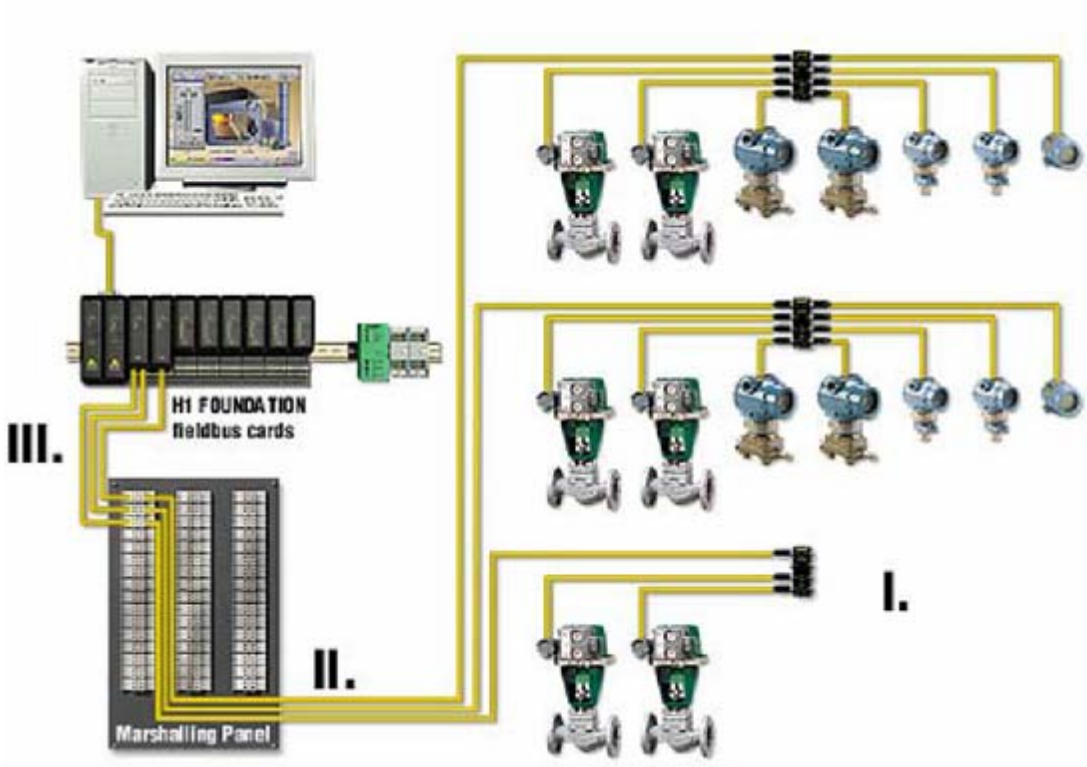
TRUE DISTRIBUTED CONTROL



Nối dây trong kiến trúc cổ điển (PLC,DCS)



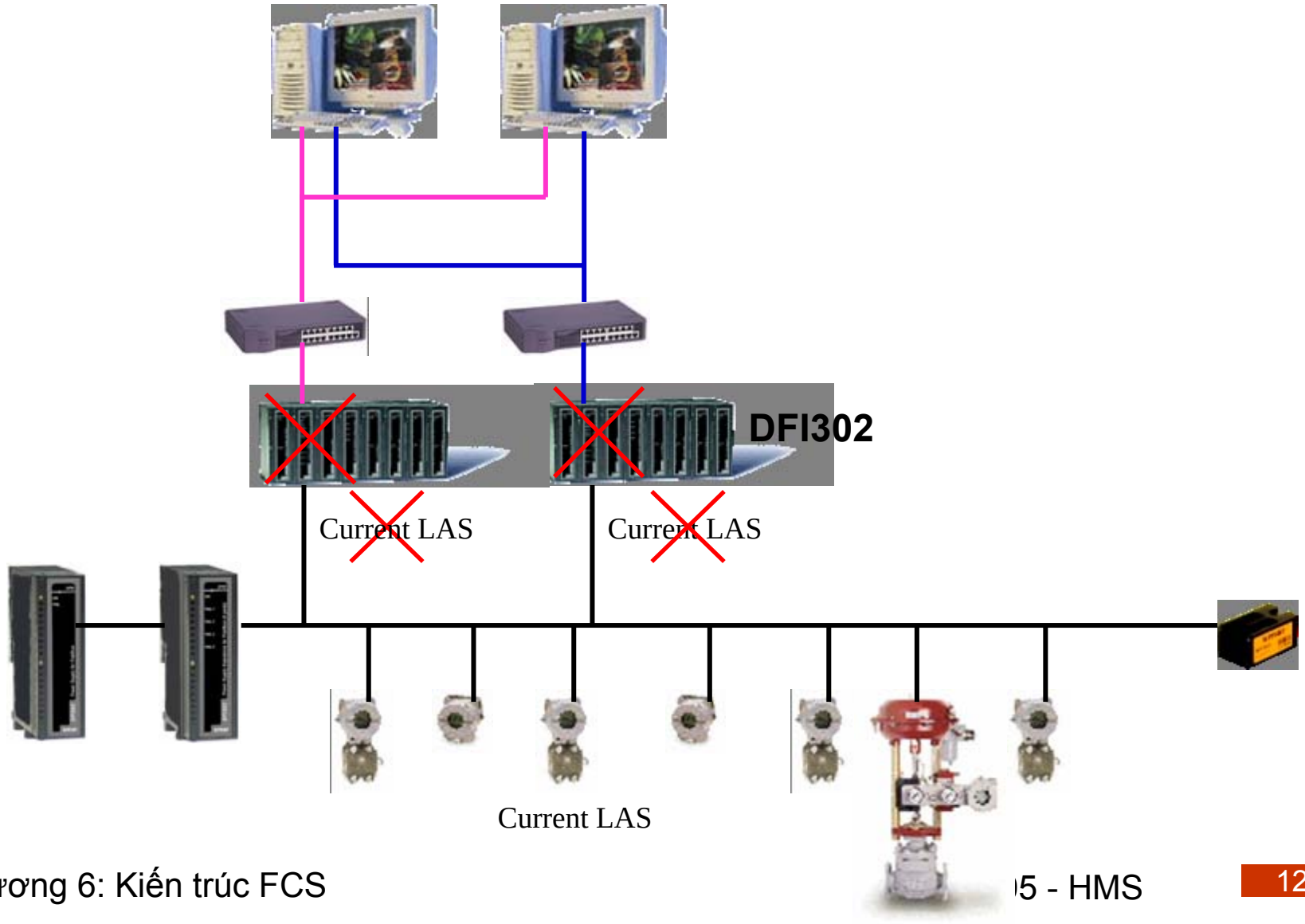
Nối dây trong kiến trúc FCS



Tiết kiệm vật liệu với FCS

- I/O terminations 75%
- I/O cards 93%
- Dây nối 98%
- Transmitters 28%
- Kích thước tủ điều khiển 67%

Cấu hình dự phòng trạm chủ

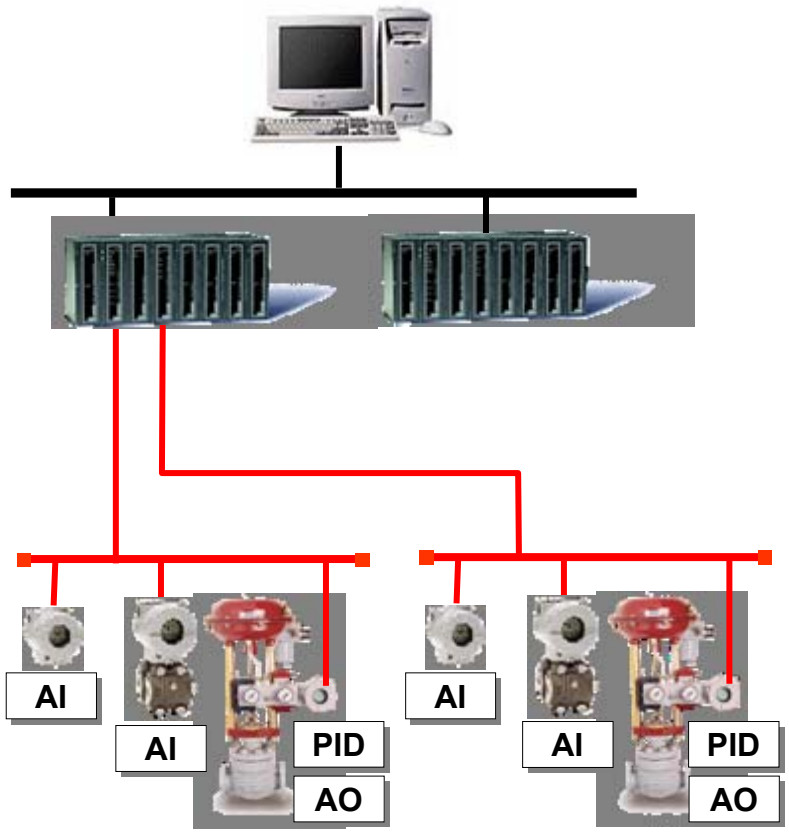


6.3 Phân tán chức năng điều khiển

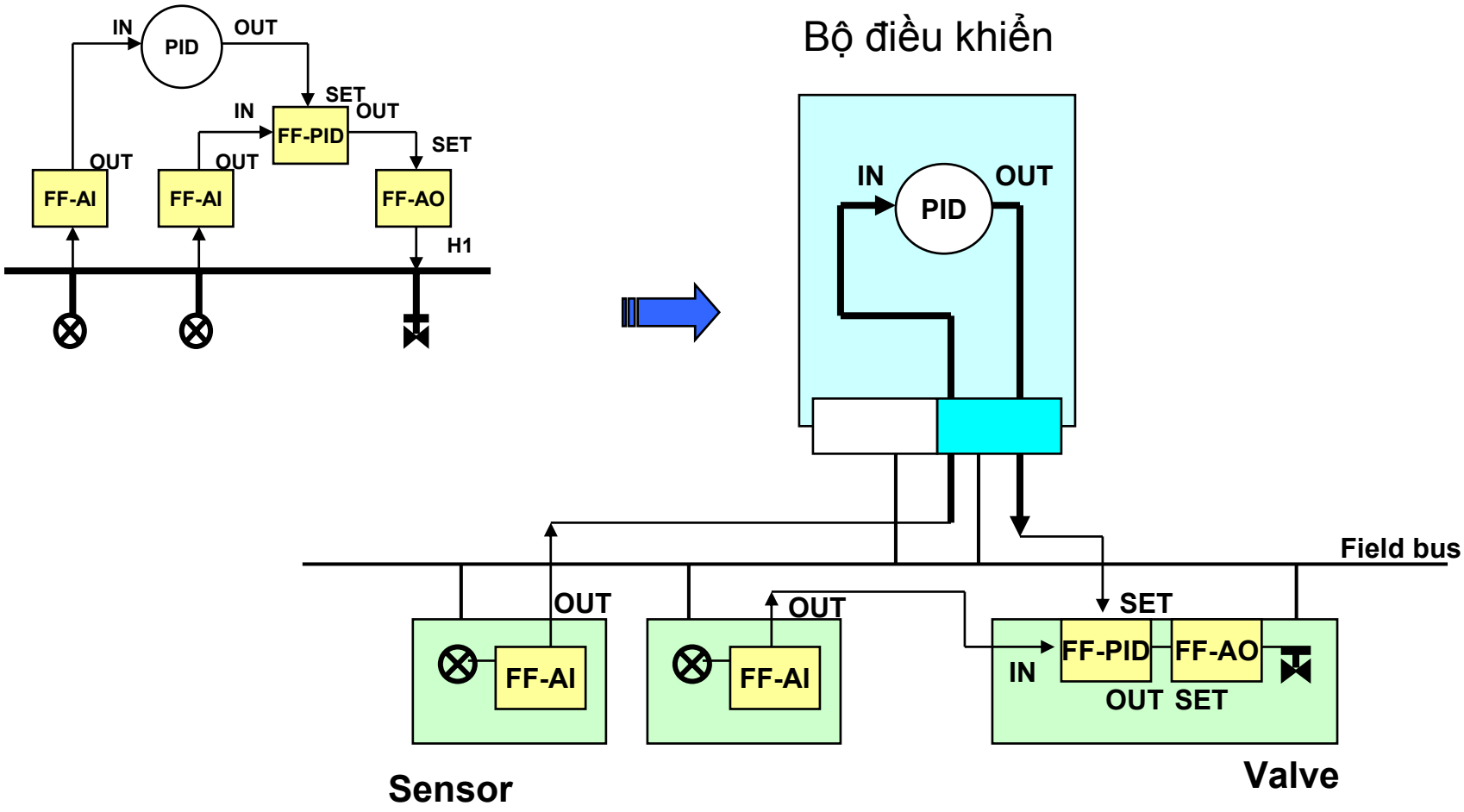
KIẾN TRÚC PLC/DCS

- Không cần I/O
- Không cần bộ điều khiển dự phòng
- Nâng cao hiệu năng và độ tin cậy
- Cấu hình thay vì lập trình

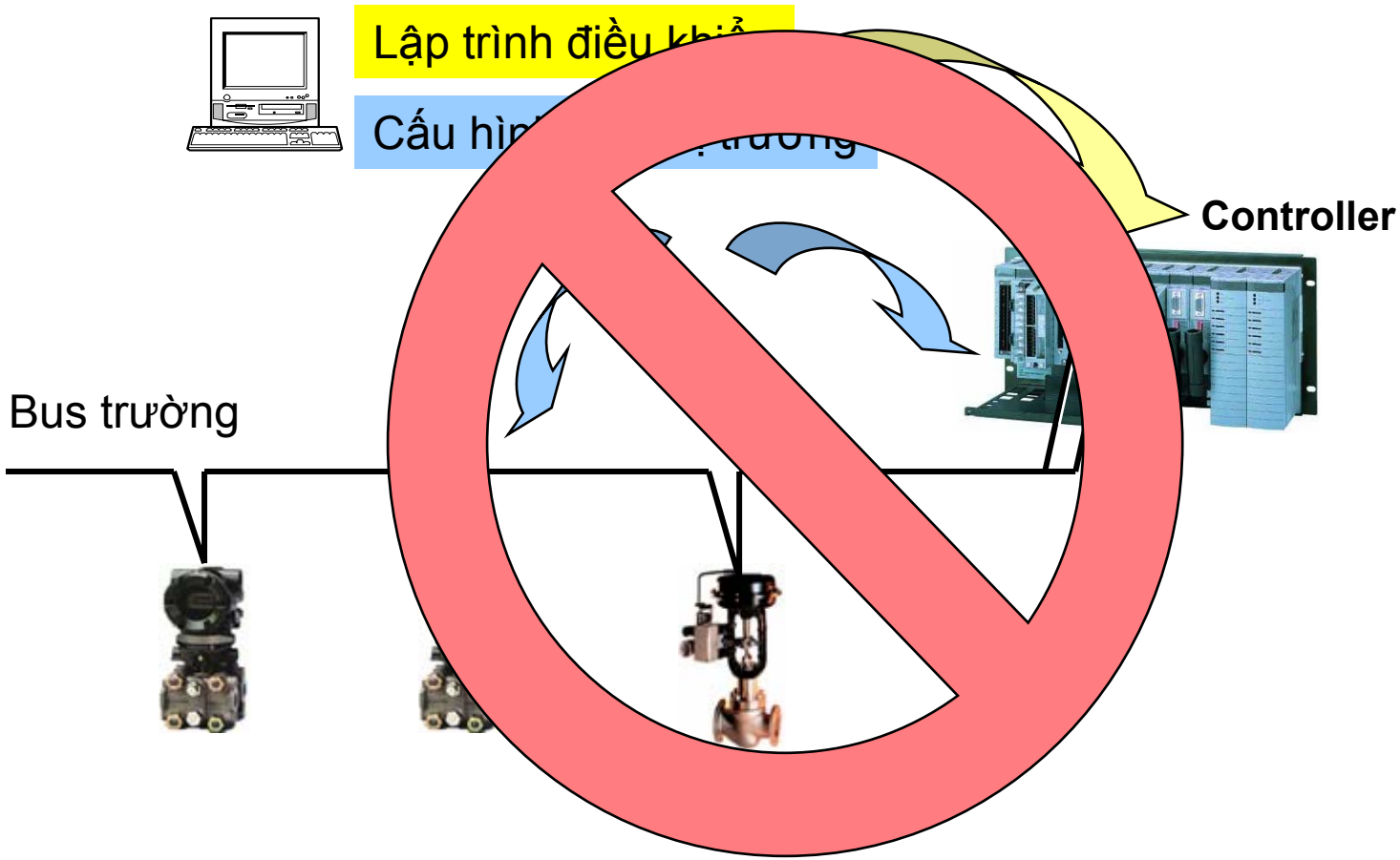
KIẾN TRÚC FCS



Phân cấp điều khiển - Ví dụ ĐK cascade

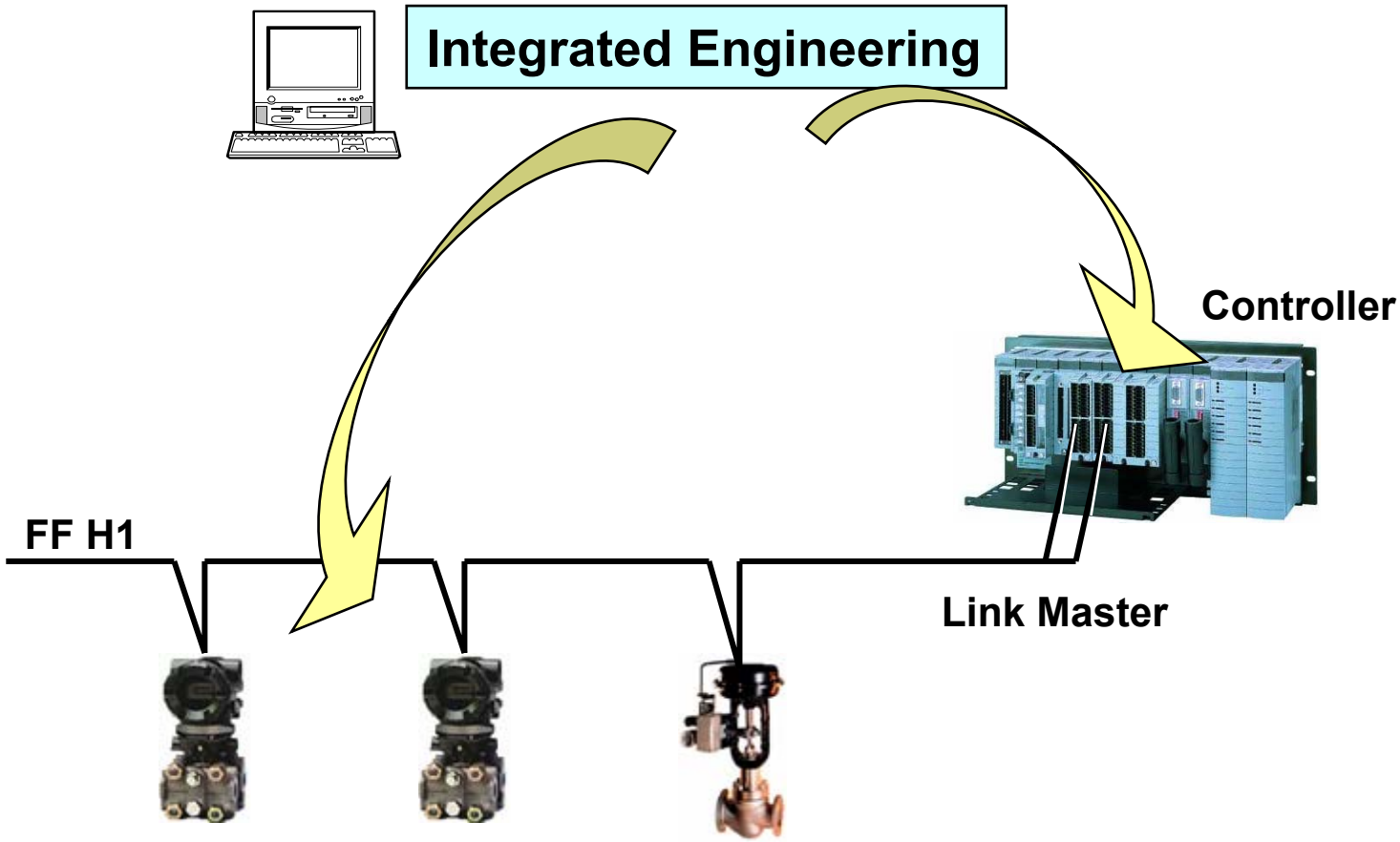


6.4 Phát triển hệ thống

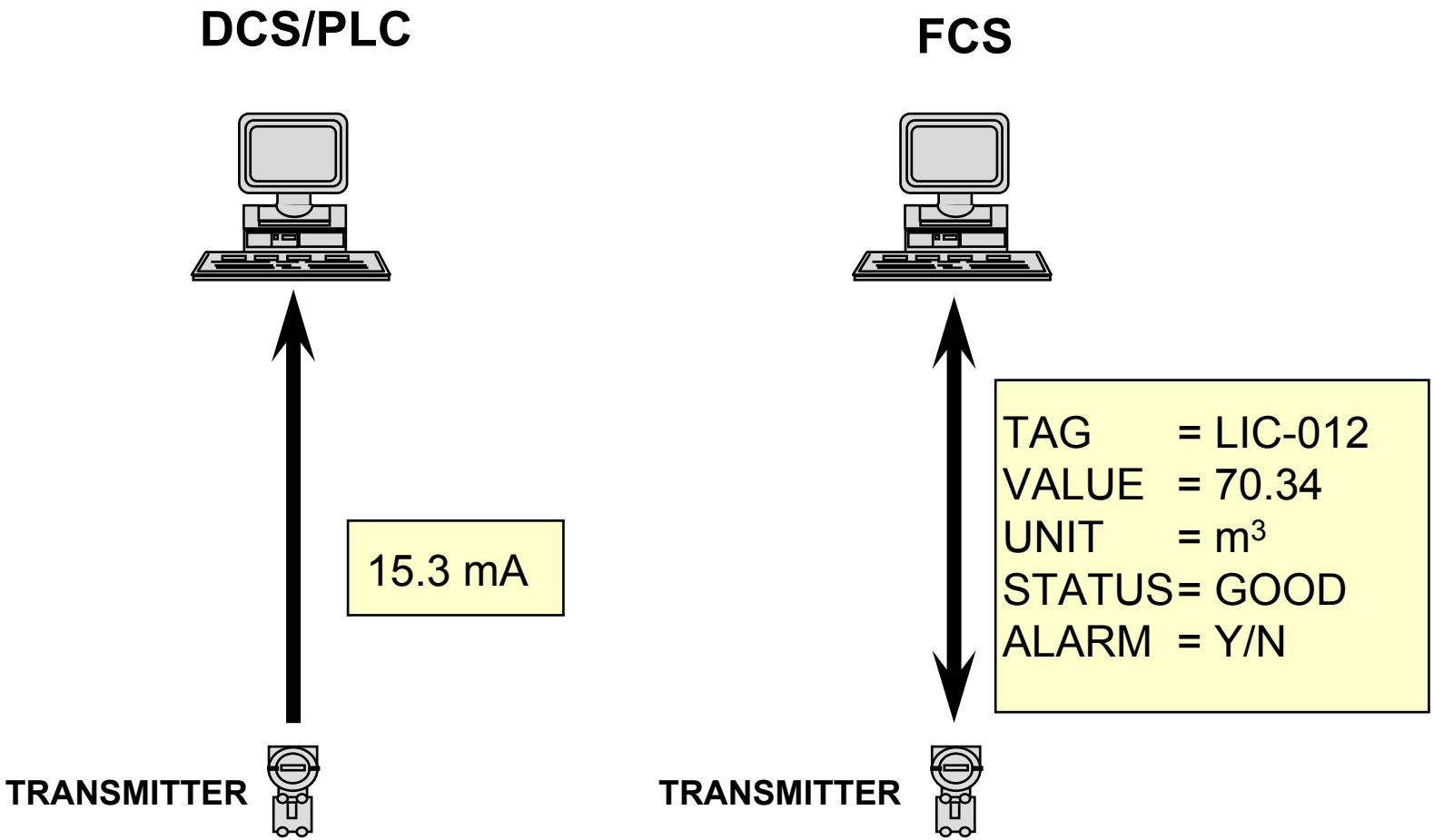


Giải pháp cổ điển: Tách biệt giữa nhiệm vụ điều khiển và cấu hình thiết bị trường

FCS: Phát triển tích hợp



Tích hợp thông tin



6.5 Tóm tắt các ưu điểm chính

- Tiết kiệm vật liệu (I/O, I/O termination, cáp truyền, tủ điều khiển) và công lắp đặt
- Nâng cao hiệu năng và độ tin cậy của hệ thống nhờ điều khiển tại chỗ, giảm tải bus
- Đưa thiết bị trường vào vận hành đơn giản nhờ môi trường phát triển tích hợp
- Tạo dựng ứng dụng điều khiển đơn giản => cấu hình thay vì lập trình
- Nâng cao độ tin cậy nhờ khả năng chẩn đoán => bảo trì phòng ngừa

Hệ thống điều khiển phân tán

Chương 7: Xử lý thời gian thực và xử lý phân tán

13.09.06

Chương 7: Nội dung

7.1 Khái niệm “thời gian thực”

7.2 Hệ điều hành thời gian thực

7.3 Khái niệm “xử lý phân tán”

7.4 Các kiến trúc xử lý phân tán

7.5 Các cơ chế giao tiếp trong hệ phân tán

7.1 Khái niệm thời gian thực

Tại sao cần nghiên cứu về *xử lý thời gian thực*

- *Xử lý thời gian thực* là nguyên lý làm việc cơ bản của mỗi bộ điều khiển, nhìn từ quan điểm *tin học*
- Chất lượng điều khiển và độ tin cậy của hệ thống điều khiển không chỉ phụ thuộc vào *thuật toán điều khiển*, *công nghệ phần cứng*, mà còn phụ thuộc một cách tất yếu vào phương pháp *xử lý thời gian thực*
- Chúng ta còn biết quá ít về cơ chế thực hiện các chức năng bên trong một bộ điều khiển (số)
- Chúng ta cũng còn biết tương đối ít về cơ chế giao tiếp giữa các thành phần mềm trong một hệ thống điều khiển phân tán

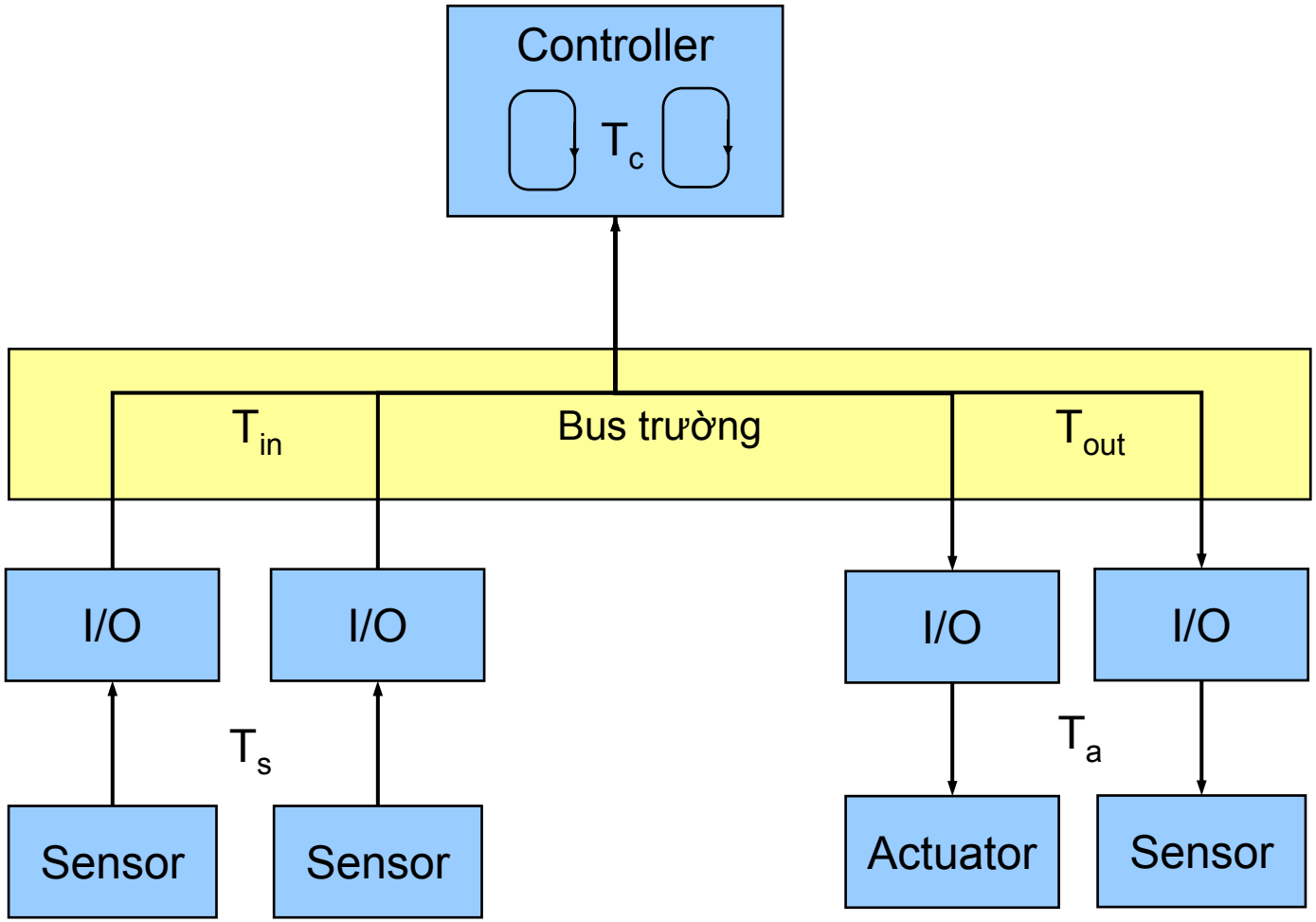
Hệ thời gian thực là gì?

A real-time system is one in which the correctness of the system depends not only on the logical results, but also on the time at which the results are produced.

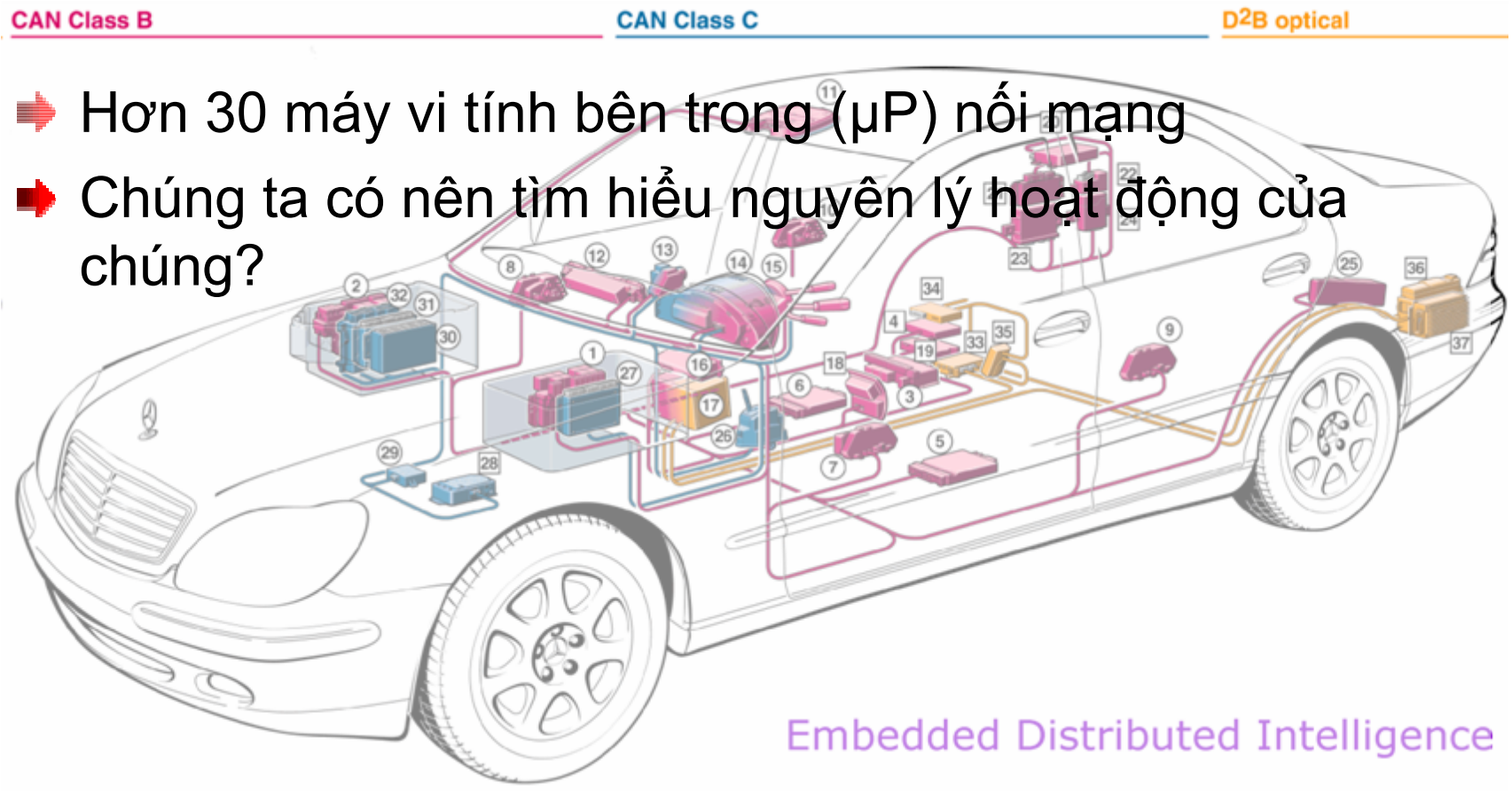
JOHN A. STANKOVIC ET AL.: Strategic Directions in Real-Time and Embedded Systems. ACM Computing Surveys, Vol. 28, No. 4, December 1996

- Mỗi hệ thống điều khiển là một hệ thời gian thực
- Phần lớn các hệ thời gian thực là các hệ thống điều khiển

Vấn đề thời gian trong hệ ĐK qua mạng

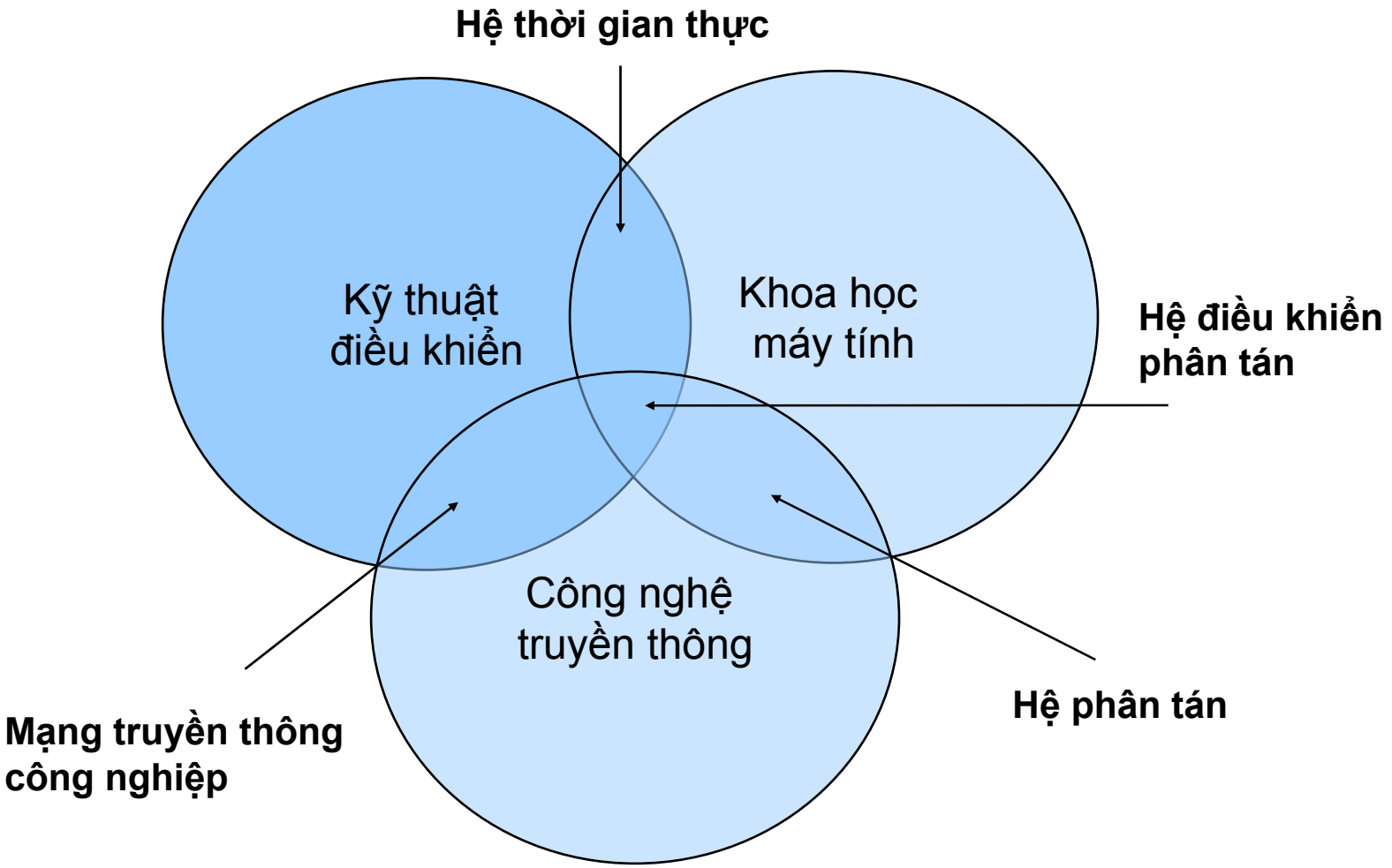


Chiếc xe hơi có là một hệ thời gian thực?



- Hơn 30 máy vi tính bên trong (μP) nối mạng
- Chúng ta có nên tìm hiểu nguyên lý hoạt động của chúng?

Nội dung liên ngành



Một hệ thời gian thực có các đặc điểm:

- *Tính phản ứng*: 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 đồng thời*: Hệ thống phải có khả năng phản ứng và xử lý đồng thời nhiều sự kiện diễn ra.
- *Tính tiền 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.

Xử lý thời gian thực là gì?

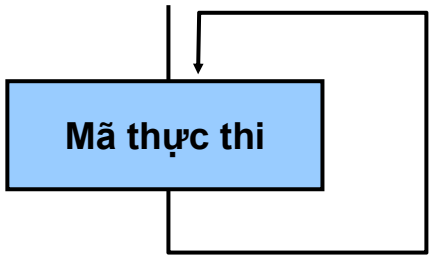
Xử lý thời gian thực là hình thức xử lý thông tin trong một hệ thống để đảm bảo tính năng thời gian thực của nó.

- Luôn liên quan với các *sự kiện bên ngoài* (*tính phản ứng*)
- Yêu cầu cao về *hiệu suất phần mềm* (*tính nhanh nhạy*)
- Đòi hỏi *xử lý đồng thời* nhiều tác vụ (*tính đồng thời*)
- Đòi hỏi cơ sở lý thuyết chặt chẽ phục vụ phân tích và đánh giá (*tính tiền định*)

Khái niệm "tác vụ" (task)

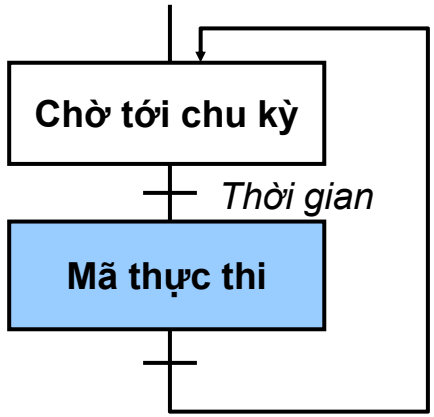
- Một quá trình tính toán cho một nhiệm vụ cụ thể, có thể được thực hiện đồng thời, ví dụ:
  - Các tác vụ xử lý giá trị vào/ra
  - Các tác vụ điều chỉnh
  - Các tác vụ điều khiển logic
  - Các tác vụ xử lý biến cố
  - ...
- Một tác vụ là sự thi hành một chương trình hoặc một phần chương trình
  - Một chương trình chạy nhiều lần => nhiều tác vụ
  - Một đoạn mã chương trình (ví dụ một hàm) được gọi tuần hoàn với các chu kỳ khác nhau => nhiều tác vụ khác nhau
- Multitasking (đa nhiệm): khả năng thi hành đồng thời nhiều tác vụ

Phân loại tác vụ (IEC 61131-3)



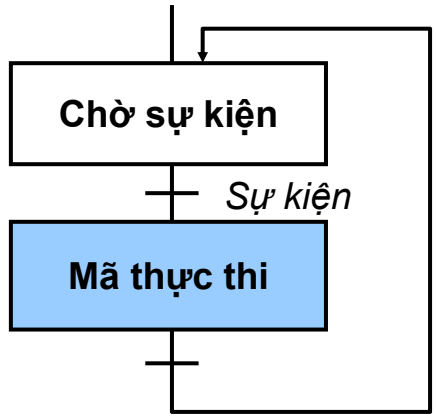
Tác vụ mặc định

Ví dụ:
- Điều khiển logic
- Kiểm tra lỗi



Tác vụ tuần hoàn

Ví dụ:
- Điều chỉnh vòng kín
- Xử lý truyền thông



Tác vụ sự kiện

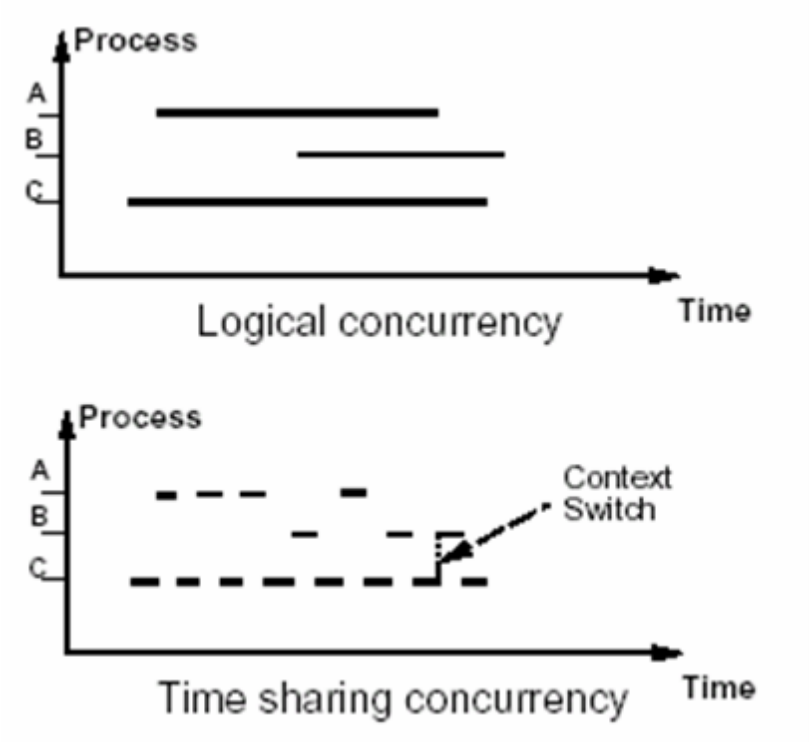
Ví dụ:
- Điều khiển trình tự
- Xử lý sự cố

Các hình thức xử lý đồng thời

- *Xử lý song song*: Các tác vụ (*task*) được phân chia thực hiện song song trên nhiều bộ xử lý
- *Xử lý cạnh tranh*: Nhiều tác vụ chia sẻ thời gian của một bộ xử lý.
- *Xử lý phân tán*: Mỗi (nhóm) tác vụ được thực hiện riêng trên một máy tính (trường hợp đặc biệt của xử lý song song).

- *Xử lý cạnh tranh là hình thức quan trọng nhất trong các hệ thống điều khiển (có thể kết hợp với xử lý phân tán)*

Xử lý cạnh tranh



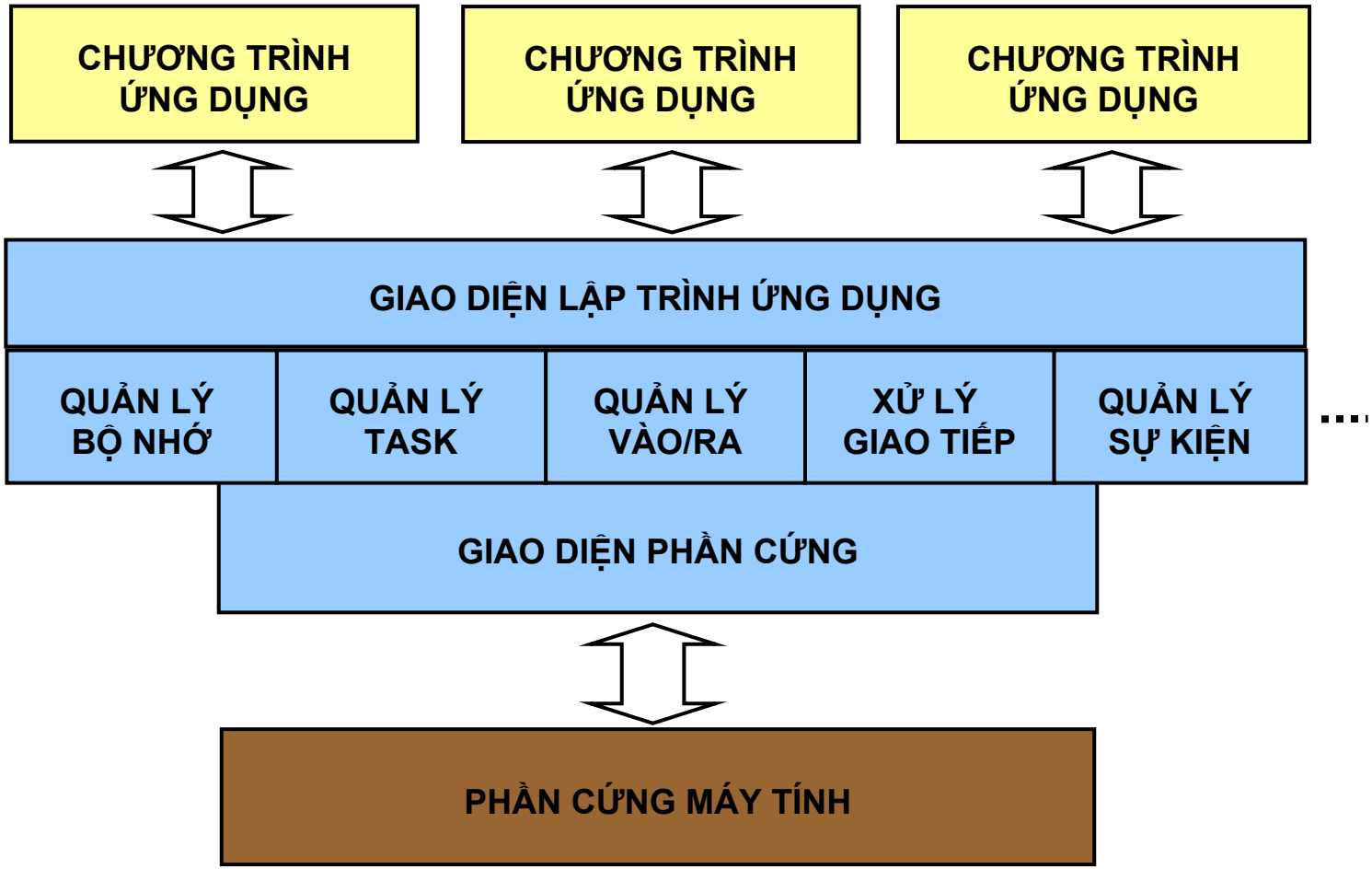
- Các vấn đề:
  - Tổ chức, lập lịch phân chia tài nguyên cho các tác vụ
  - Giao tiếp giữa các tác vụ
  - Đồng bộ hóa giữa các tác vụ

7.2 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
- Hầu hết các bộ điều khiển công nghiệp (PLC, DCS,...) đều hoạt động trên nền một hệ điều hành thời gian thực (RTOS, *Real-time Operating System*)
- Bản thân hệ điều hành thời gian thực cũng là một hệ thời gian thực
- Một hệ điều hành thời gian thực bao giờ cũng là một hệ đa nhiệm (*multitasking*), hỗ trợ xử lý cạnh tranh hoặc/và xử lý song song.

Các nhiệm vụ chính của hệ điều hành thời gian thực trong một bộ điều khiển

- Nạp chương trình, hỗ trợ thử nghiệm, gỡ rối chương trình
- Quản lý dữ liệu vào/ra và quản lý truyền thông
  - Giúp các chương trình ứng dụng dễ dàng truy cập dữ liệu mà không cần quan tâm tới cơ chế phần cứng cụ thể
- Quản lý tác vụ:
  - *Lập lịch*: Phân chia thời gian CPU cho thi hành các tác vụ khác nhau (trong xử lý cạnh tranh)
  - *Hỗ trợ đồng bộ hóa quá trình*: Giúp các tác vụ chia sẻ tài nguyên sử dụng chung (bộ nhớ, cổng vào/ra,..)
  - *Hỗ trợ giao tiếp liên quá trình*: Giúp các tác vụ thực hiện giao tiếp, trao đổi dữ liệu hoặc phối hợp hoạt động
- Các chức năng kiểm tra, chẩn đoán lỗi


CHƯƠNG TRÌNH ỨNG DỤNG
CHƯƠNG TRÌNH ỨNG DỤNG
CHƯƠNG TRÌNH ỨNG DỤNG
GIAO DIỆN LẬP TRÌNH ỨNG DỤNG
QUẢN LÝ BỘ NHỚ
QUẢN LÝ TASK
QUẢN LÝ VÀO/RA
XỬ LÝ GIAO TIẾP
QUẢN LÝ SỰ KIỆN
....
GIAO DIỆN PHẦN CỨNG
PHẦN CỨNG MÁY TÍNH

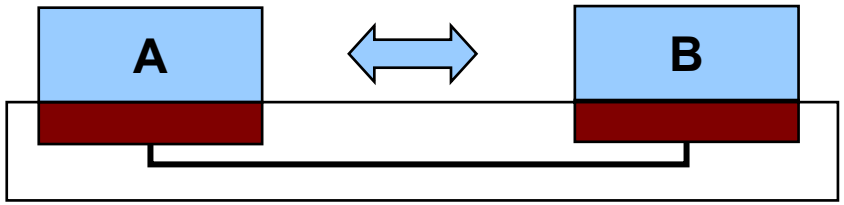
Phương pháp lập lịch

- Cơ chế lập lịch
  - Lập lệnh tĩnh: thứ tự thực hiện các tác vụ được xác định trước khi hệ thống đi vào hoạt động.
  - Lập lệnh động: thứ tự thực hiện các tác vụ được xác định trong khi hệ thống đang hoạt động.
- Sách lược lập lịch
  - FIFO: đến trước sẽ được thực hiện trước.
  - Mức ưu tiên cố định/động: các tác vụ được đặt các mức ưu tiên cố định hoặc có thể thay đổi nếu cần.
  - Preemptive: chen hàng, chọn một tác vụ để thực hiện trước các tác vụ khác.
  - Non-preemptive: không chen hàng, các tác vụ được thực hiện bình thường dựa trên mức ưu tiên của chúng.
- Thuật toán lập lịch
  - 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.

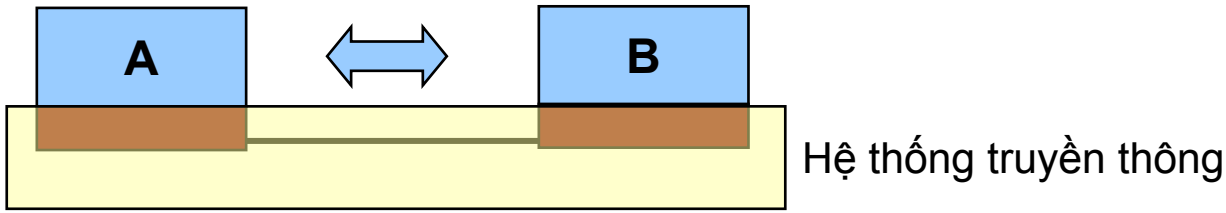
7.3 Khái niệm xử lý phân tán

- *Xử lý phân tán* là hình thức xử lý thông tin tất yếu của các hệ thống phân tán nói chung và các hệ thống điều khiển phân tán nói riêng
- 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 (giao tiếp hiện)
  - Xử lý phân tán => ứng dụng phân tán (giao tiếp ngầm)

Giao tiếp ngầm ⇔ Giao tiếp hiện



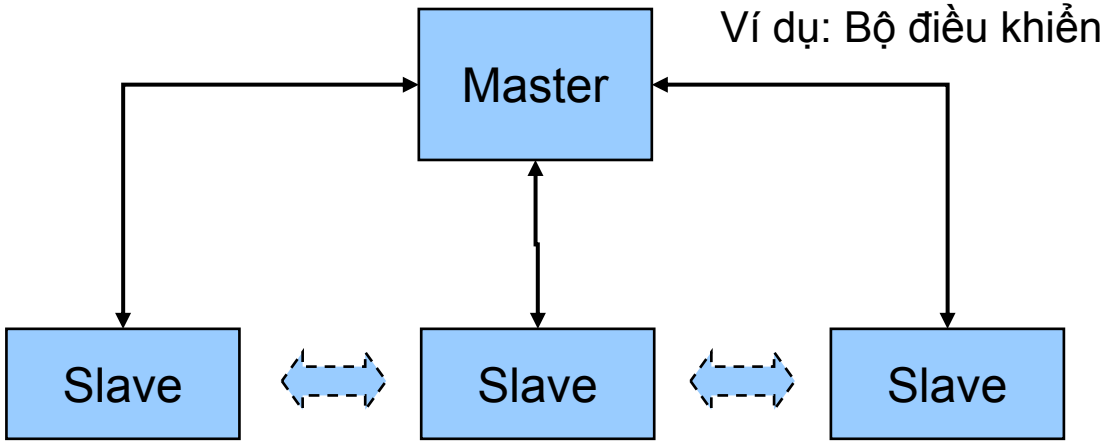
- Giao tiếp hiện (*explicit communication*):
  - Hoạt động giao tiếp được coi là chức năng riêng
  - Sử dụng dịch vụ giao tiếp (ví dụ lập trình) cần biết rõ về hệ thống truyền thông (kiến trúc giao thức)



- Giao tiếp ngầm (im*plicit communication*):
  - Hoạt động giao tiếp được thực hiện ngầm khi cần thiết
  - Sử dụng dịch vụ giao tiếp (ví dụ lập trình) cần biết rõ về hệ thống truyền thông (kiến trúc giao thức)

7.4 Các kiến trúc xử lý phân tán

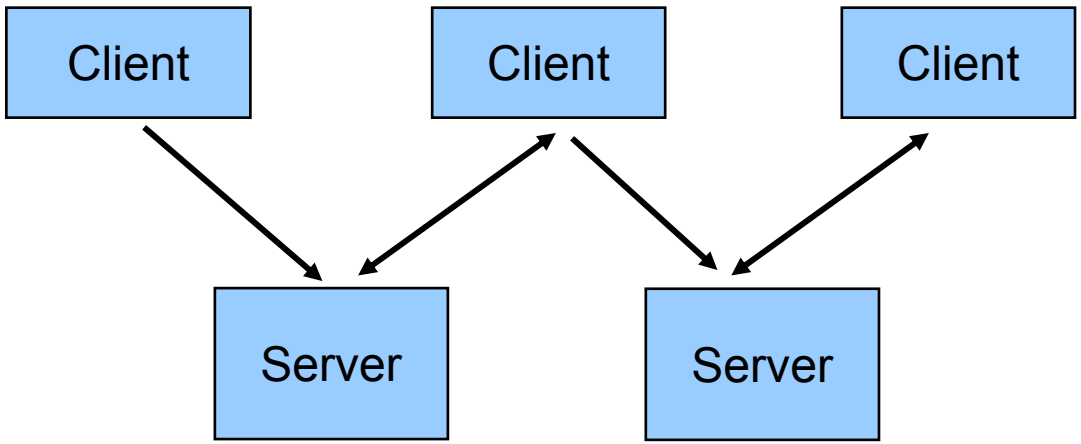
- Kiến trúc Master/Slave
  - Chức năng xử lý thông tin được phân chia trên nhiều trạm tớ
  - Một trạm chủ phối hợp hoạt động của nhiều trạm tớ
  - Các trạm tớ có vai trò, nhiệm vụ tương tự như nhau
  - Các trạm tớ có thể giao tiếp trực tiếp, hoặc không



Ví dụ: Các vào/ra phân tán, các thiết bị trường

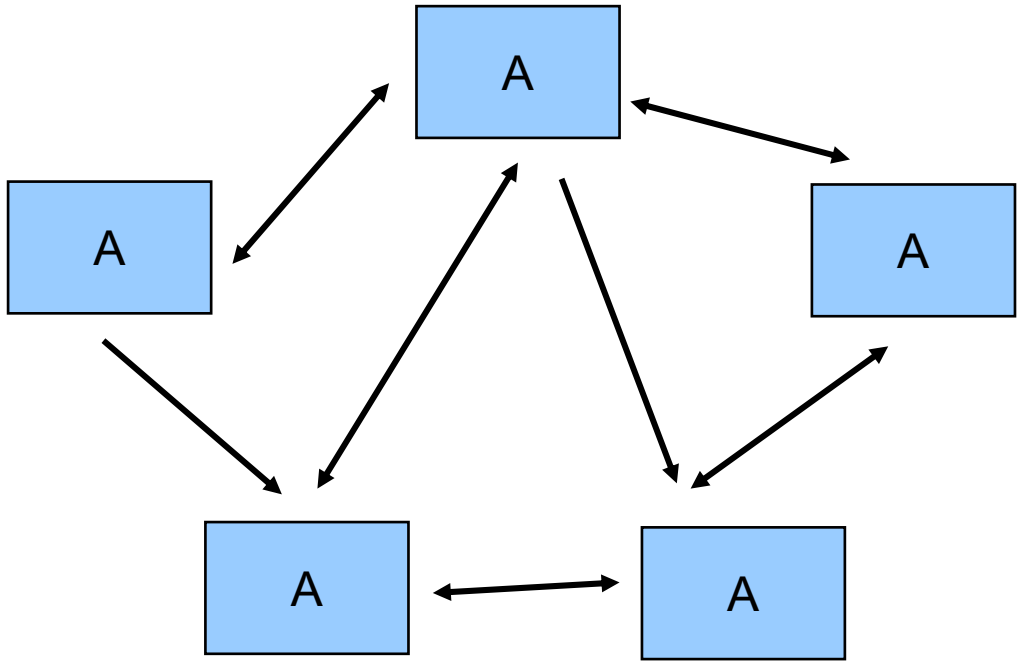
- Kiến trúc Client/Server
  - Chức năng xử lý thông tin được phân chia thành hai phần khác nhau, phần sử dụng chung cho nhiều bài toán được thực hiện trên các server, phần riêng thực hiện trên từng client.
  - Giữa các client không cần thiết có giao tiếp trực tiếp
  - Vai trò chủ động trong giao tiếp thuộc về client

Ví dụ: Các trạm vận hành



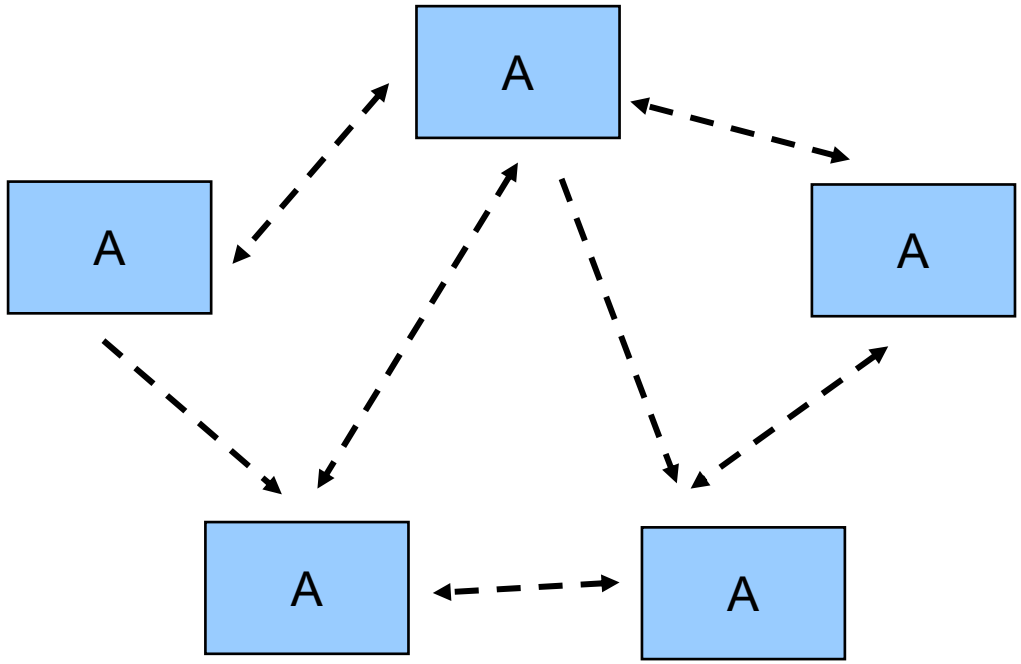
Ví dụ: Các bộ điều khiển hoặc các trạm quản lý dữ liệu

- Kiến trúc bình đẳng
  - Các trạm có vai trò bình đẳng, phải phối hợp hoạt động, hình thức giao tiếp trực tiếp với nhau không qua trung gian



Ví dụ: Các trạm điều khiển phân tán (kiến trúc PLC/DCS) hoặc các thiết bị trường thông minh (kiến trúc FCS)

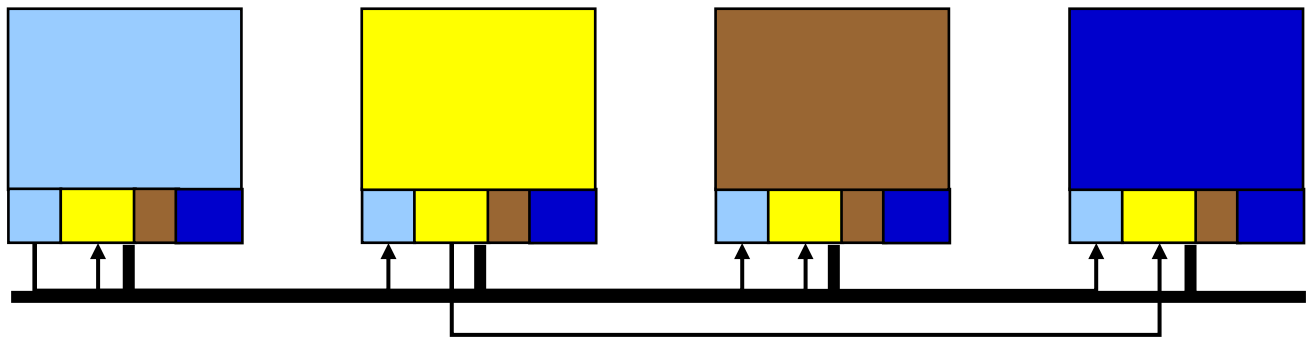
- Kiến trúc tự trị
  - Các trạm có vai trò bình đẳng, có thể hoạt động hoàn toàn độc lập nhưng sự phối hợp hoạt động tạo hiệu quả cao nhất



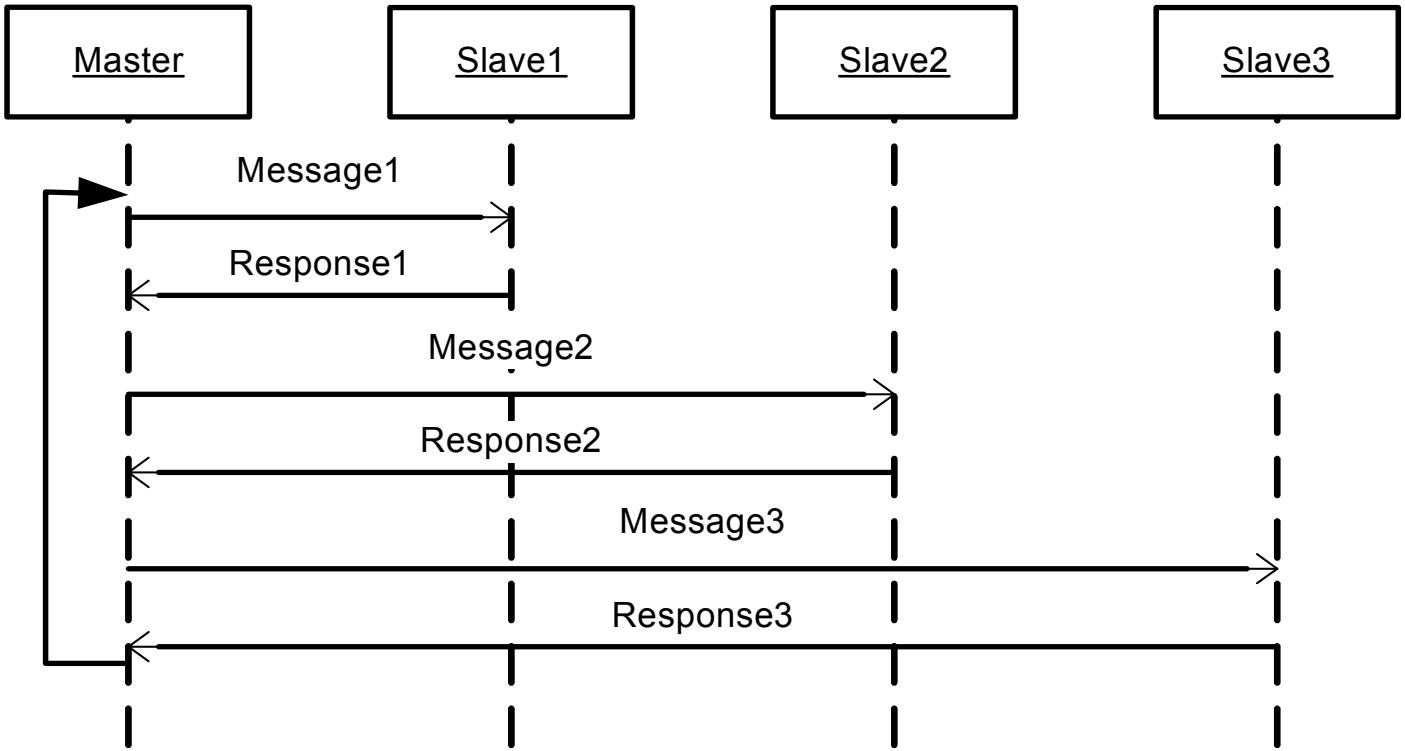
Ví dụ: Các hệ thống xây dựng theo công nghệ Agent, Multi-Agent

7.5 Các cơ chế giao tiếp trong hệ ĐKPT

- 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, tiền 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).

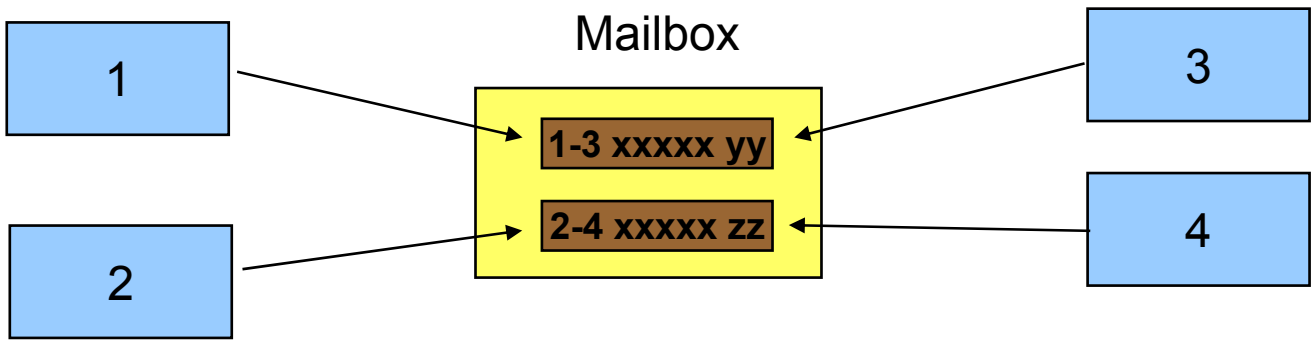


- Hỏi tuần tự (Polling, Scanning)
  - Một trạm đóng vai trò Master
  - Cơ chế hỏi/đáp tuần tự theo trình tự đặt trước
  - Đơn giản, tiền định, hiệu quả cao
  - Áp dụng cho trao đổi dữ liệu tuần hoàn, thích hợp trong kiến trúc Master/Slave



- 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, tiền 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, không 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.



Hệ thống điều khiển phân tán

Chương 8: Công nghệ hướng đối tượng trong điều khiển phân tán

13.09.06

Chương 8: Công nghệ đối tượng trong điều khiển phân tán

8.1 Qui trình công nghệ phần mềm

8.2 Công nghệ đối tượng là gì

8.3 Ngôn ngữ mô hình hóa thống nhất UML

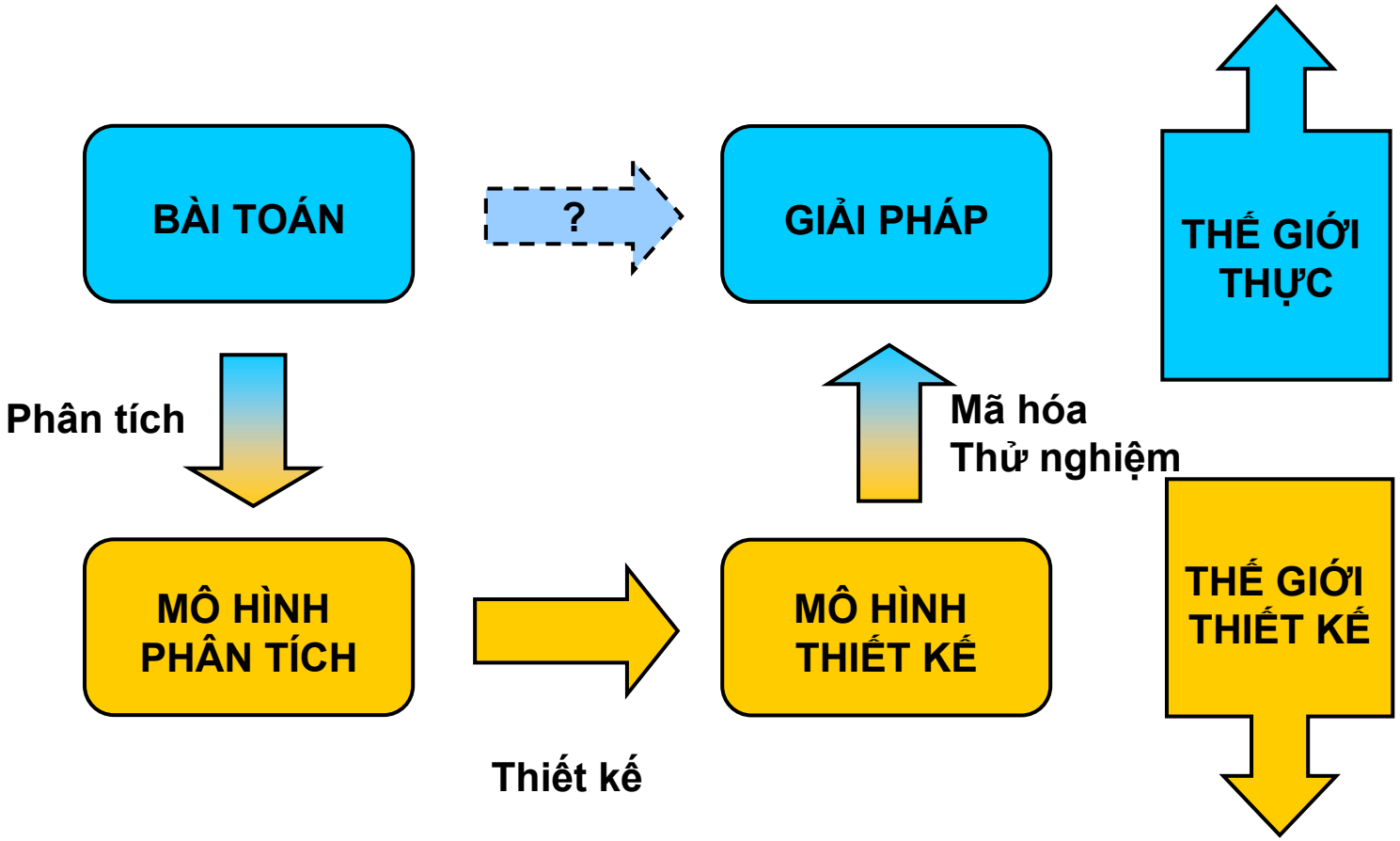
8.4 Khái niệm đối tượng phân tán

8.5 Mô hình COM/DCOM

8.6 Lập trình với COM/DCOM

Tài liệu: *Tự động hóa ngày nay* 5/04-8/04 (CD: \papers\ )
UML Reference Manual (CD:\UML\refman.pdf)
Dr. GUI on COM and ATL (CD: \com-dcom\)

8.1 Qui trình công nghệ phần mềm



Phân tích yêu cầu (Requirement analysis)

- Bởi vì: Khách hàng thường không biết là họ muốn gì, nhưng họ biết chắc chắn là họ không muốn gì
- Cho nên: Cần phải cùng với khách hàng làm rõ những yêu cầu về phạm chức năng, về giao diện sử dụng
- Kết quả: Mô hình đặc tả (*Specification Model*), một phần của hợp đồng
- Cần một ngôn ngữ mô hình hóa dễ hiểu để trao đổi giữa khách hàng và nhóm phân tích

⇨ Trả lời câu hỏi: **Khách hàng cần những gì**

Phân tích hệ thống (System analysis)

- Phân tích mối liên hệ của hệ thống với môi trường xung quanh
- Tìm ra cấu trúc hệ thống và các thành phần quan trọng
- Định nghĩa chức năng cụ thể của các thành phần
- Nhận biết các đặc điểm của từng thành phần
- Phân loại các thành phần, tổng quát hóa, đặc biệt hóa
- Nhận biết mối liên hệ giữa các thành phần
- Kết quả: Mô hình hệ thống (*System model*)
- Cần một ngôn ngữ mô hình hóa để trao đổi giữa các thành viên trong nhóm phân tích và với nhóm thiết kế

⇨ Trả lời câu hỏi: **Những gì sẽ phải làm**

Thiết kế hệ thống (System Design)

- Dựa trên mô hình hệ thống, xây dựng các mô hình chi tiết phục vụ sẵn sàng mã hóa/cài đặt
- Bao gồm:
  - Thiết kế cấu trúc (*structured design*): chương trình, kiểu dữ liệu, đối tượng, quan hệ cấu trúc giữa các đối tượng và kiểu)
  - Thiết kế tương tác (*interaction design*): quan hệ tương tác giữa các đối tượng
  - Thiết kế hành vi (*behaviour design*): sự kiện, trạng thái, phép toán, phản ứng
  - Thiết kế chức năng (*funtional design*): tiến trình hành động, hàm, thủ tục)
- Kết quả: Mô hình thiết kế (các bản vẽ và lời văn mô tả)

⇨ Trả lời câu hỏi: **Làm như thế nào**

Các bước khác

- Mã hóa/cài đặt (*Coding/Implementation*): Thể hiện mô hình thiết kế bằng một ngôn ngữ/công cụ lập trình cụ thể
- Thử nghiệm (*Testing, Verification*): Chạy thử, phân tích và kiểm chứng:
  - Thử đơn vị (*Unit Test*)
  - Thử tích hợp (*Integration Test*)
- Gỡ rối (*Debugging*): Tìm ra và sửa các lỗi chương trình chạy (các lỗi logic)
- Xây dựng tài liệu (*Documenting*): Xây dựng tài liệu phát triển, tài liệu hướng dẫn sử dụng
- Đào tạo, chuyển giao
- Bảo trì, bảo dưỡng

8.2 Công nghệ (hướng) đối tượng là gì?

Các nội dung của công nghệ phần mềm, được xây dựng trên cơ sở phương pháp luận hướng đối tượng

- Mô hình hóa hướng đối tượng
- Phân tích, thiết kế hướng đối tượng
- Lập trình hướng đối tượng
- Phần mềm thành phần
- Đối tượng phân tán
- ...

Công nghệ hướng đối tượng có vai trò then chốt trong công nghiệp phần mềm hiện nay và trong tương lai

Đối tượng là gì?

- Mô hình/đại diện của một đối tượng vật lý:
  - Tank, Heater, Furnace
  - Motor, Pump, Valve
  - Sensor, Thermometer, Flowmeter
  - Control Loop, Control System
- Hoặc một đối tượng logic ("conceptual object):
  - Trend, Report, Button, Window
  - Matrix, Vector, Polynomial
- Đóng gói dữ liệu + phép toán áp dụng

Một đối tượng có...

- Các thuộc tính (attributes)
- Trạng thái (state)
  - Dữ liệu
  - Quan hệ
- Hành vi (behavior)
  - Các phép toán
  - Đặc tính phản ứng
- Căn cước (identity)
- Ngữ nghĩa/trách nhiệm (semantic/responsibilities)

Nguyên lý cơ bản của phương pháp luận hướng đối tượng

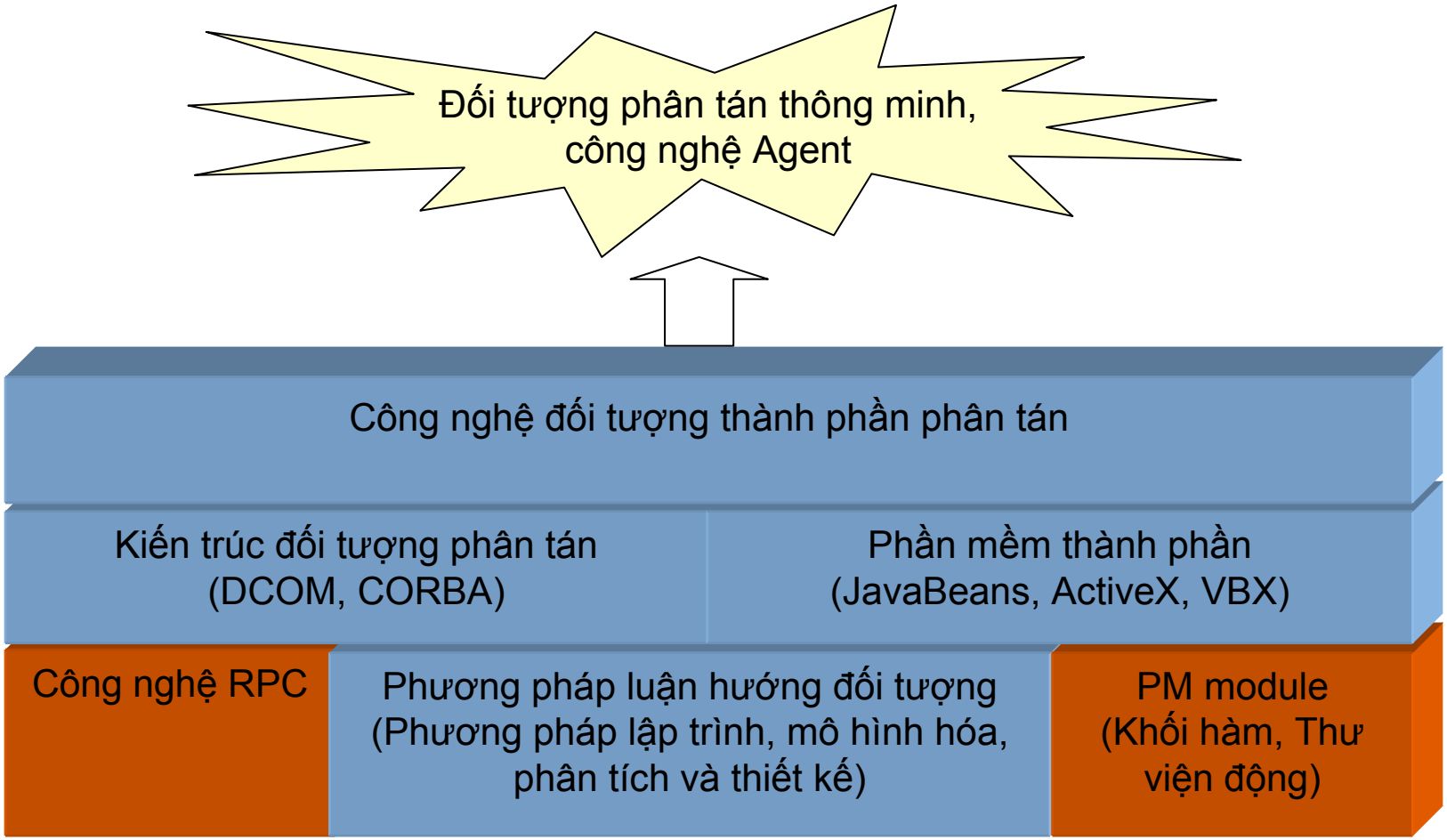
- Trừu tượng hóa (*abstraction*): giúp đơn giản hóa vấn đề, dễ sử dụng lại
- Đóng gói dữ liệu/che dấu thông tin (*data encapsulation/ information hiding*): nâng cao giá trị sử dụng lại và độ tin cậy của phần mềm
- Dẫn xuất/thừa kế (*subtyping/inheritance*): giúp dễ sử dụng lại mã phần mềm và thiết kế
- Đa hình/đa xạ (*polymorphism*): giúp phản ánh trung thực thế giới thực và nâng cao tính linh hoạt của phần mềm

Tại sao lại "hướng đối tượng"

*Phương pháp luận hướng đối tượng cho phép tư duy ở mức **trừu tượng cao** nhưng **gần với thế giới thực***

- Thế giới thực cấu thành bởi các đối tượng và mỗi liên hệ giữa chúng
- Mô hình nhất quán cho toàn bộ qui trình công nghệ phần mềm
- Trừu tượng hóa vấn đề tốt hơn
- Bền vững hơn với thay đổi
- Khả năng sử dụng lại cao
- Khả năng phù hợp với nhiều qui mô khác nhau
- Hỗ trợ tốt hơn cho phát triển các hệ tin cậy và an toàn
- Hỗ trợ tốt hơn cho xử lý cạnh tranh

Sự tiến hóa của công nghệ đối tượng



Vai trò của công nghệ đối tượng trong các hệ thống điều khiển?

- Vai trò của công nghệ phần mềm trong các hệ thống điều khiển?
- Có một công cụ phần mềm nào trong hệ thống điều khiển không được lập trình hướng đối tượng?
- Ví dụ về các đối tượng cụ thể:
  - Các khối chức năng: PID, AI, AO,...
  - Các khối đồ họa Windows Controls, ActiveX-Controls: Đồ thị, phím bấm, cửa sổ, bình chứa, van điều khiển, băng tải,...
  - OPC server, Web server,...

8.3 Ngôn ngữ mô hình hóa UML

Mô hình là gì?

- Một ánh xạ thế giới thực (đang tồn tại hoặc cần xây dựng)
- Mô tả thế giới thực từ một góc nhìn
- Các dạng mô hình:
  - Mô hình toán học
  - Mô hình đồ họa
  - Mô hình máy tính
- Một mô hình tốt cần đơn giản nhưng thể hiện được các đặc tính quan trọng cần quan tâm của thế giới thực
- *"Không có mô hình nào chính xác, nhưng có một số mô hình có ích!"*

Mô hình để làm gì?

- Trừu tượng hóa (đơn giản hóa) vấn đề
- Phương tiện giao tiếp trong nhóm phát triển
- Phương tiện giao tiếp giữa nhóm phát triển và khách hàng
- Phương tiện phân tích, thiết kế và kiểm chứng
- Tài liệu phần mềm

➲ Cần một ngôn ngữ mô hình hóa tốt và một phương pháp mô hình hóa thích hợp !

Thế nào là một ngôn ngữ mô hình hóa tốt

- Đơn giản, trực quan, dễ hiểu, dễ xây dựng (đồ họa)
- Khả năng biểu diễn mạnh (toán, văn bản, đồ họa)
- Khả năng thực thi (máy tính, văn bản, đồ họa máy tính)
- Linh hoạt, khả mở
- Nhất quán: cho suốt qui trình công nghệ phần mềm
- Chuẩn hóa quốc tế

UML: Unified Modeling Language

- Ngôn ngữ mô hình hóa rất mạnh, có đầy đủ các đặc tính tốt đã nêu
- Hỗ trợ mô hình hóa hướng đối tượng, hướng thành phần và các phương pháp luận khác
- Thống nhất Rumbaugh‘s *OMT*, *Booch‘94* và Ivar Jacobson‘s *Use Case*
- Chắt lọc, thừa kế nhiều phương pháp luận khác
- Ngôn ngữ mô hình hóa trung lập
- Kết hợp biểu tượng đồ họa + văn bản
- Chuẩn công nghiệp (OMG consortium: www.omg.org), đặc tả hiện tại V1.5

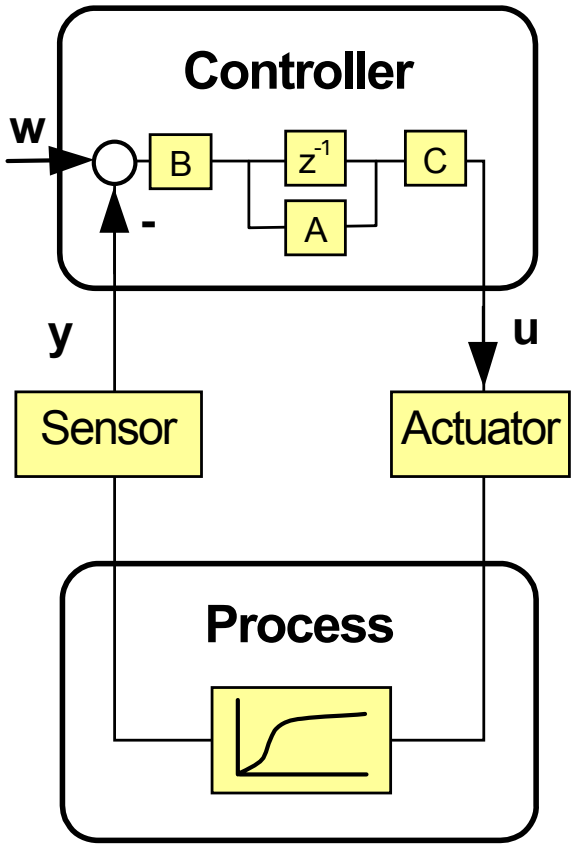
Mô hình hóa cấu trúc

- Static view
  - ⇨ Biểu đồ lớp: class, interface, inheritance, association, ...
- Use case view
  - ⇨ Biểu đồ use case: use case, scenario, ...
- Implementation view
  - ⇨ Biểu đồ thành phần: component, package, module, ...
- Deployment view
  - ⇨ Biểu đồ phân bố: node, processor, component,...

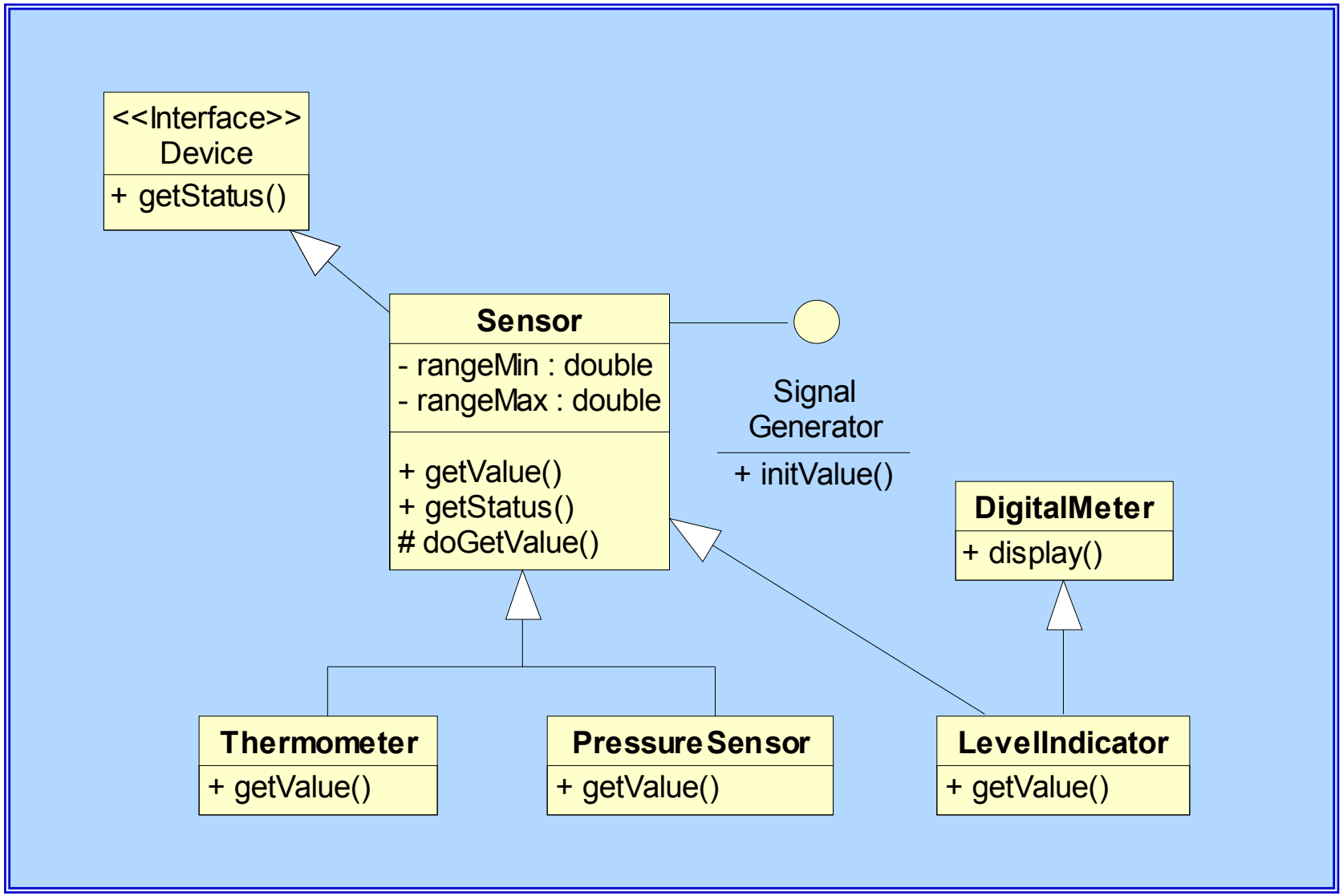
Lớp, đối tượng và giao diện

- Một lớp là thực thi của các đối tượng có chung:
  - Ngữ nghĩa
  - Thuộc tính
  - Quan hệ
  - Hành vi
- Một giao diện là một kiểu dịch vụ của đối tượng, ví dụ
  - Truy nhập thuộc tính
  - Thực hiện các phép toán

Ví dụ: Hệ thống điều khiển



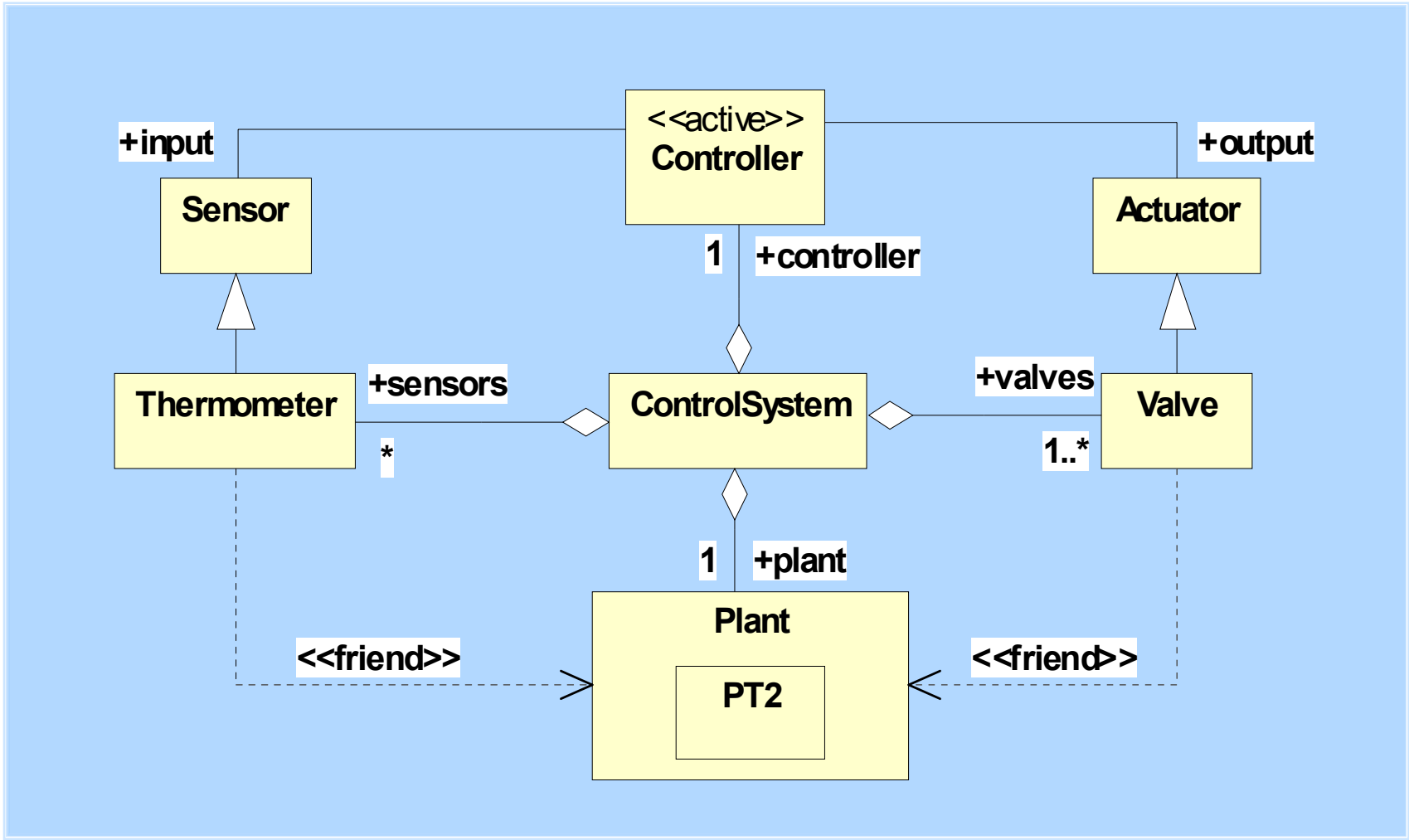
Lớp và giao diện trong UML



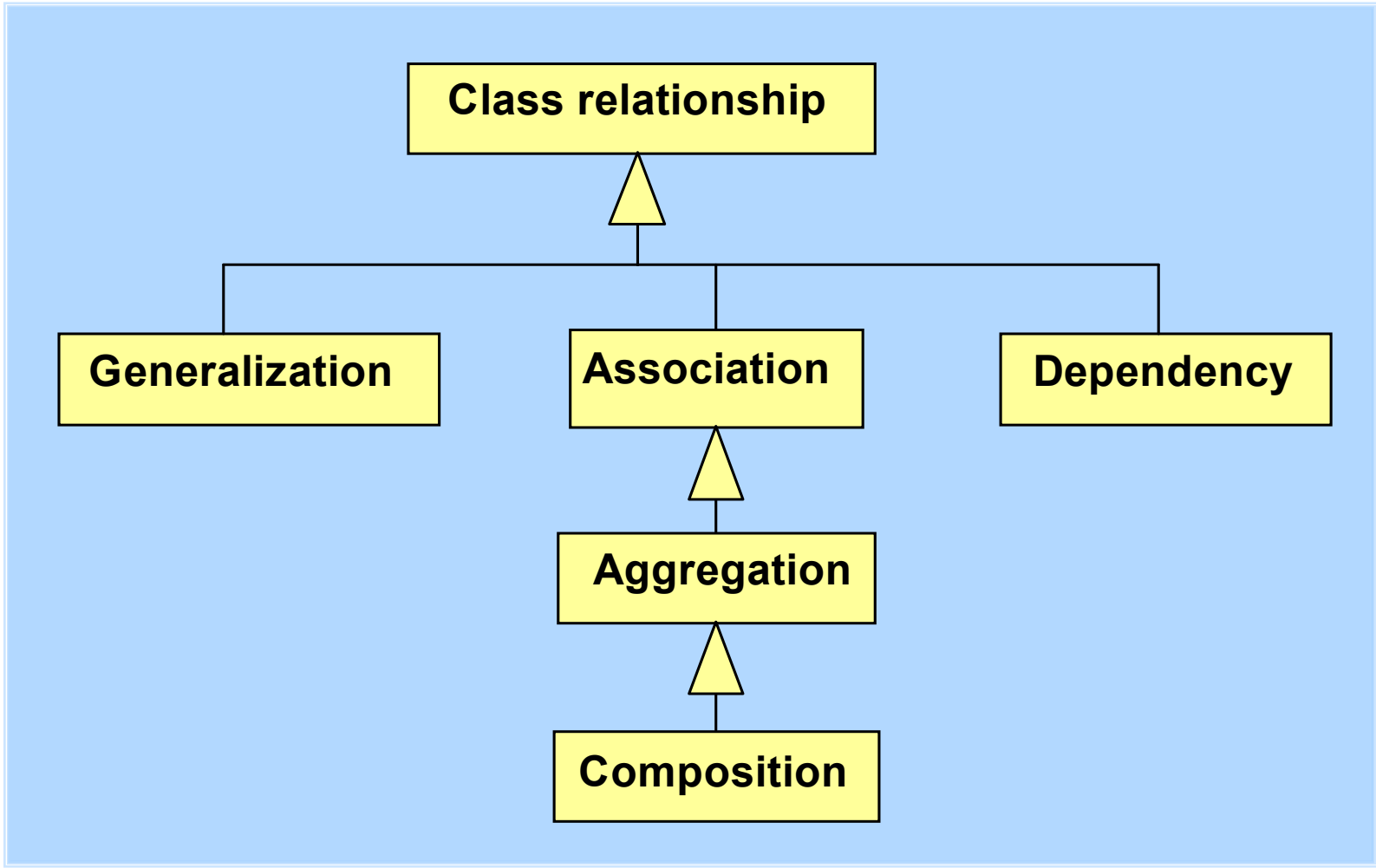
Quan hệ lớp/đối tượng

- Generalization/Specialization: Thừa kế (Inheritance), Dẫn xuất (Subtyping)
- Association: Quan hệ chung chung
- Aggregation: Quan hệ sở hữu
- Composition: Quan hệ cấu thành
- Dependency: Quan hệ sử dụng

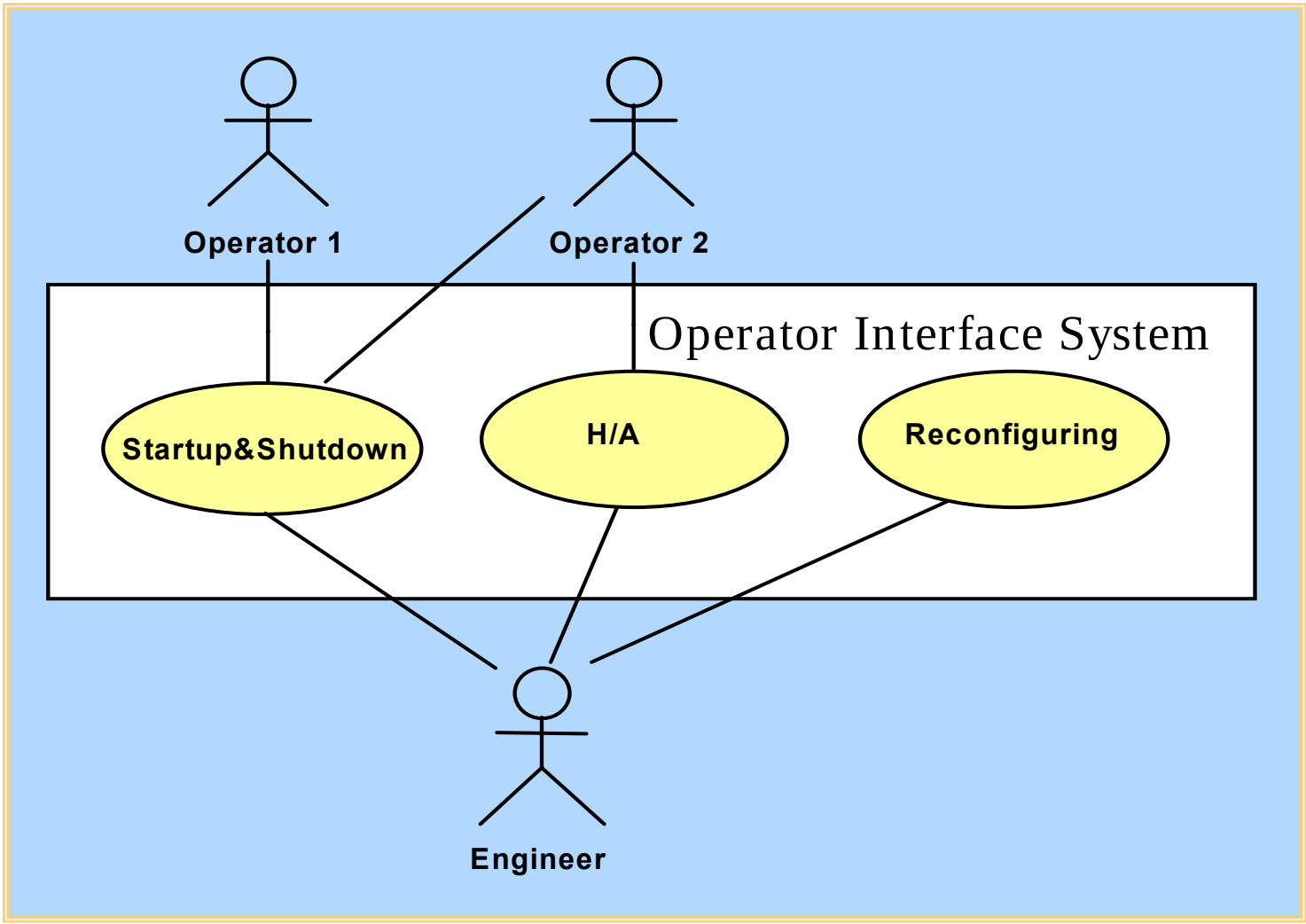
Quan hệ lớp trong UML



Quan hệ lớp - Meta model



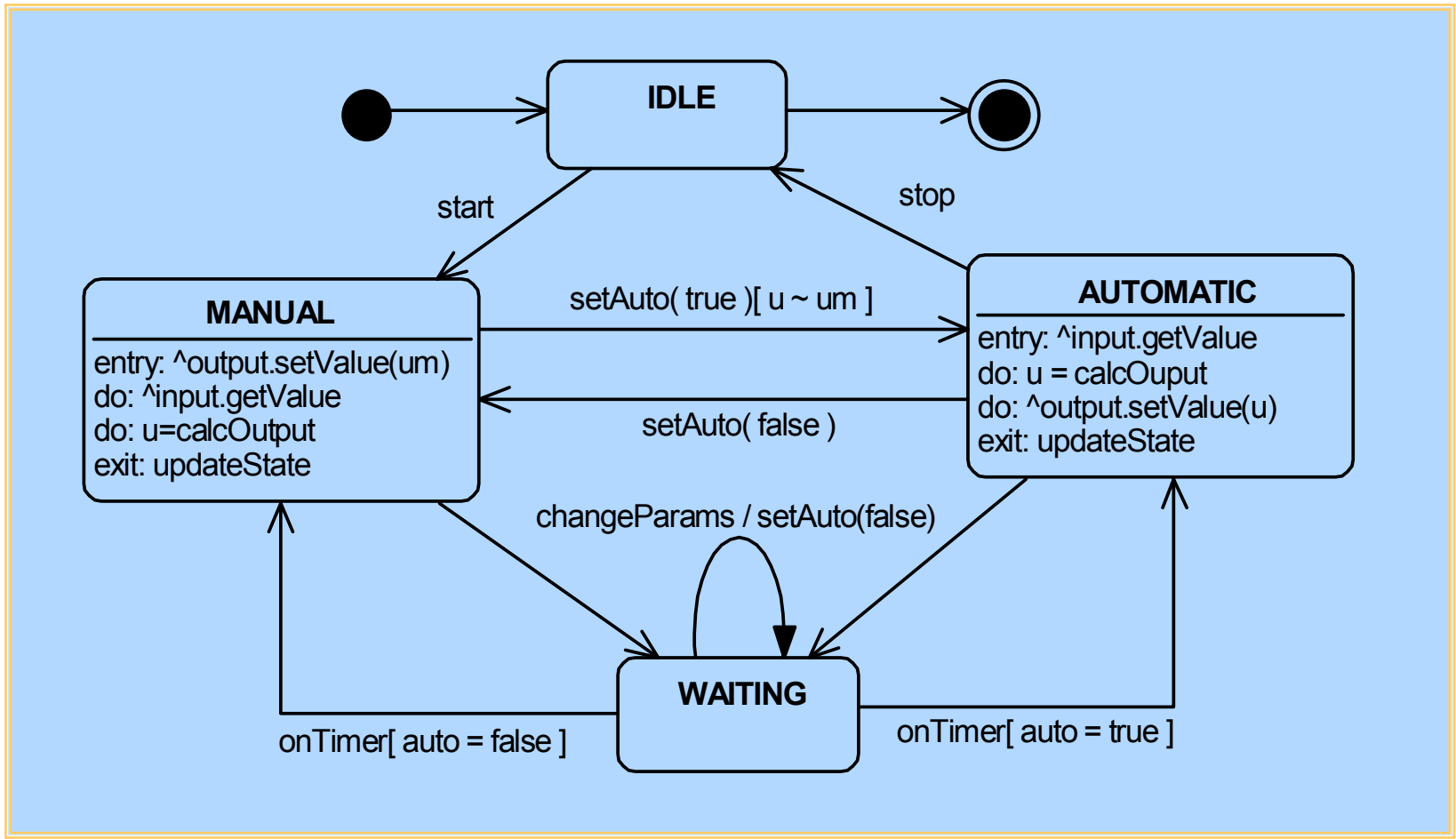
Use case



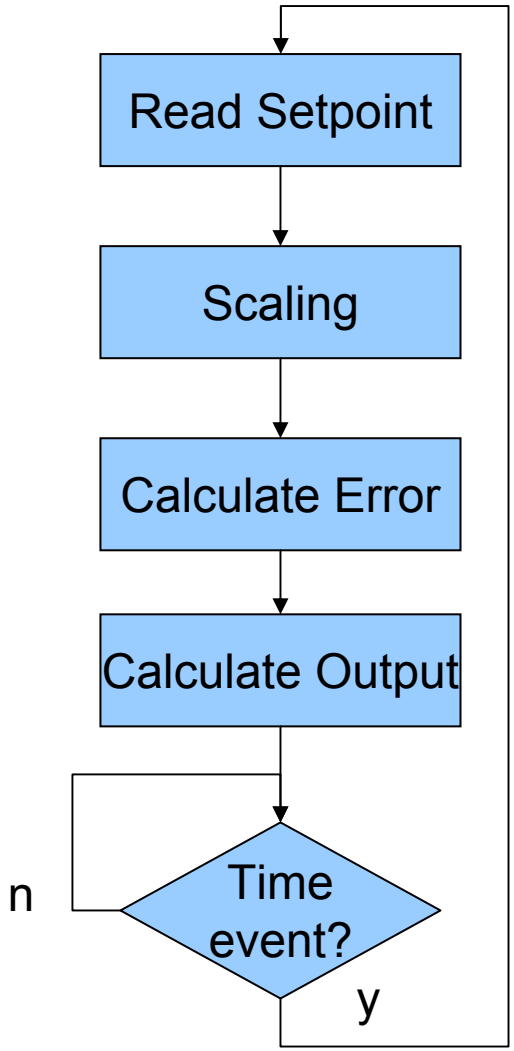
Mô hình hóa hành vi

- Hành vi đối tượng:
  - Biểu đồ trạng thái (*Statecharts*)
  - Biểu đồ hành động (*Activity diagramm*)
- Tương tác giữa các đối tượng
  - Biểu đồ trình tự (*Sequence diagram)*
  - Biểu đồ cộng tác (*Collaboration diagram*)

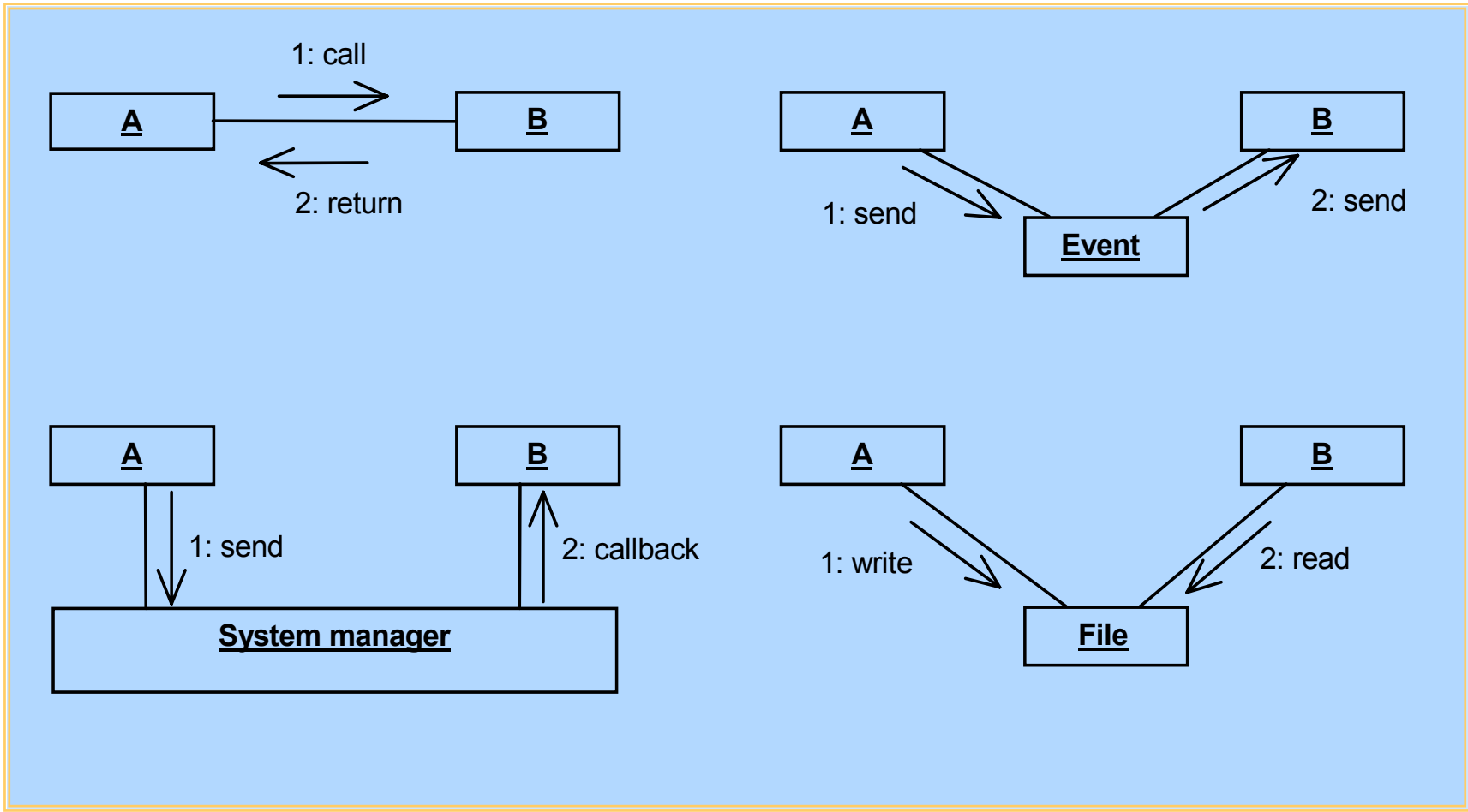
Biểu đồ trạng thái (bộ điều khiển)



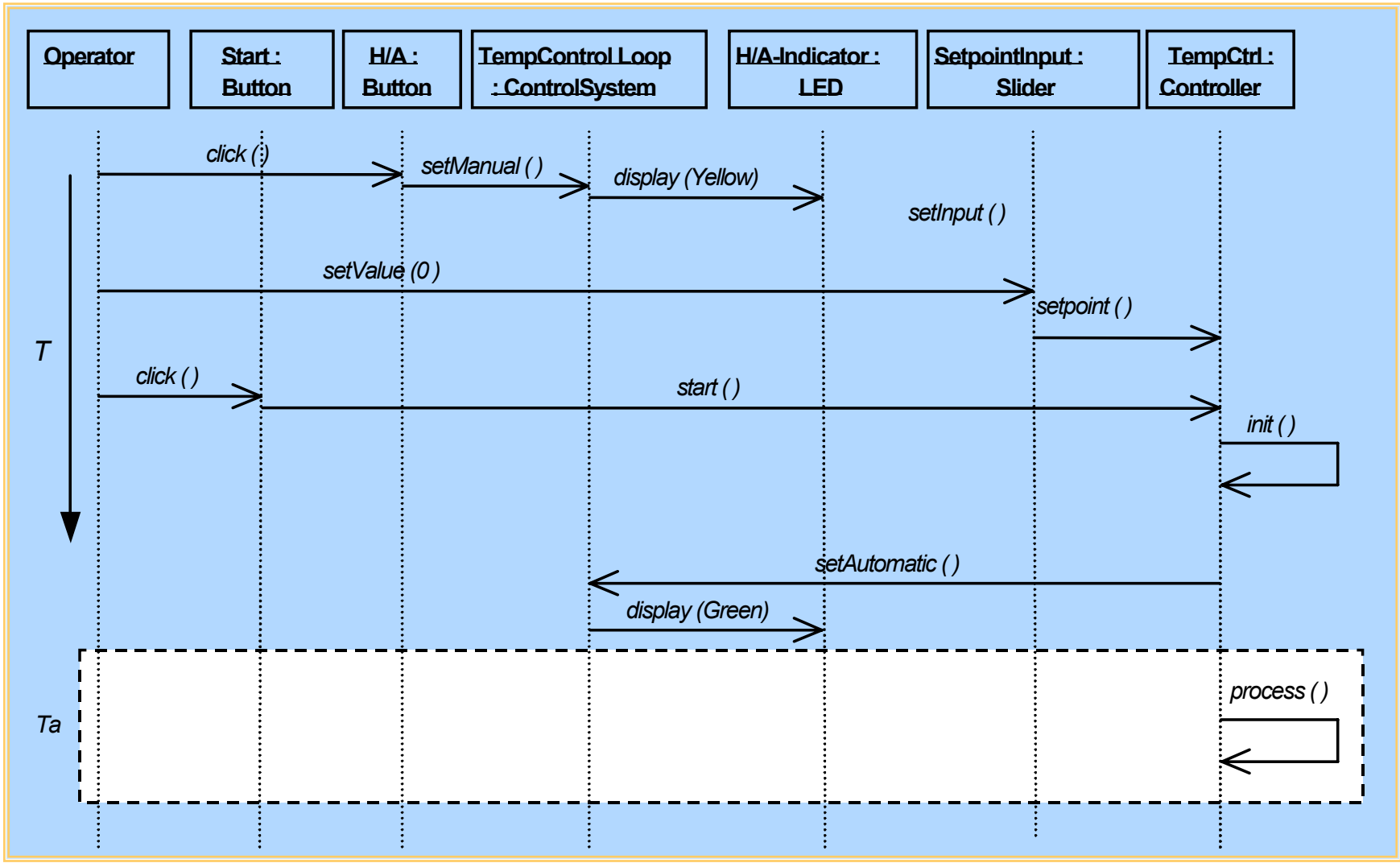
Biểu đồ hành động (calcOutput)



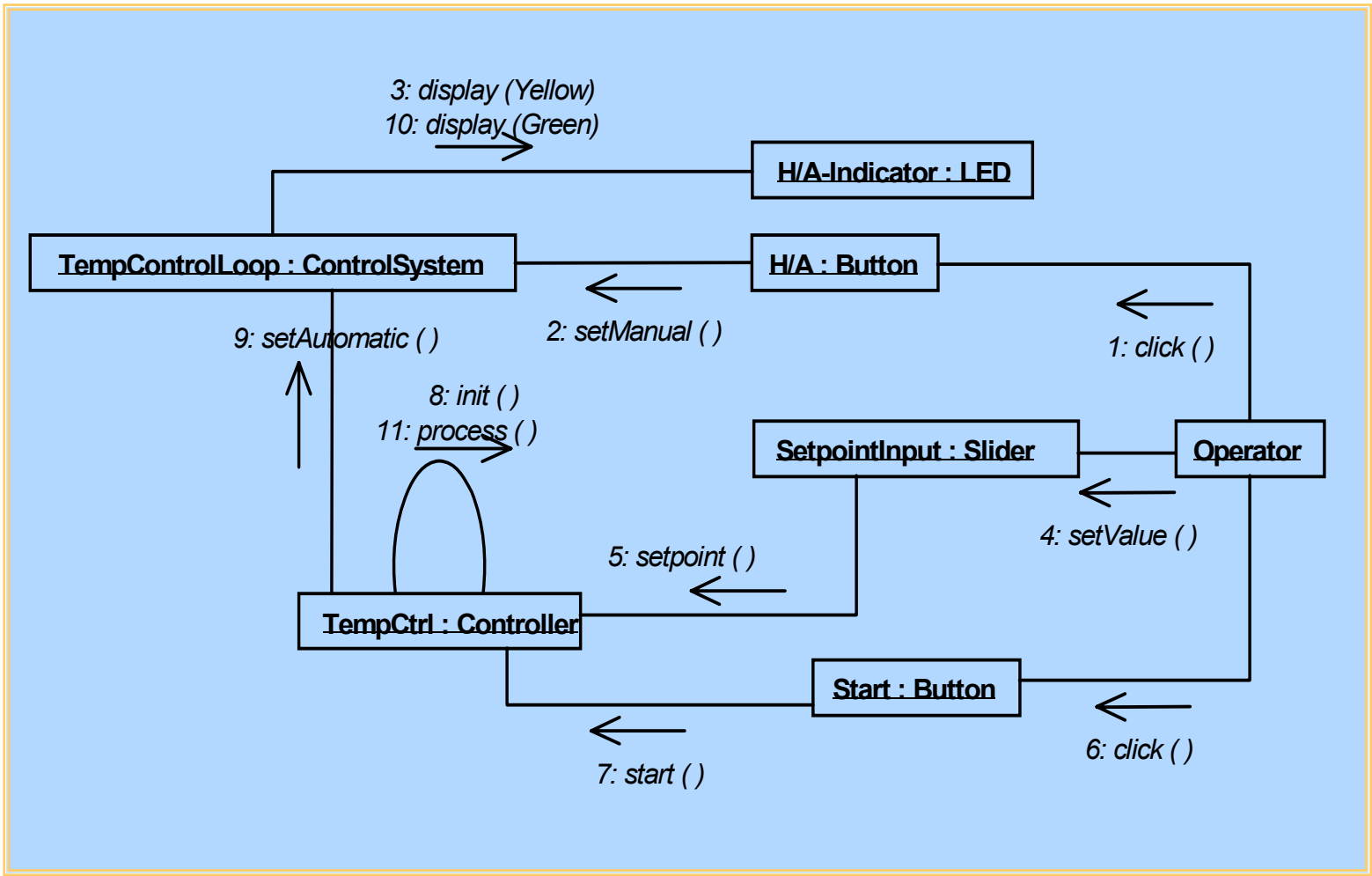
Tương tác đối tượng



Biểu đồ trình tự



Biểu đồ cộng tác



Tóm tắt

- Use case và kịch bản cho phân tích yêu cầu, phân tích hệ thống
- Biểu đồ lớp cho thiết kế cấu trúc
- Biểu đồ tương tác cho thiết kế giao diện và quan hệ tương tác giữa các đối tượng
- Biểu đồ trạng thái cho thiết kế hành vi đối tượng và thiết kế cụ thể các phép toán
- Biểu đồ hành động cho thiết kế thuật toán, chi tiết thực thi phép toán

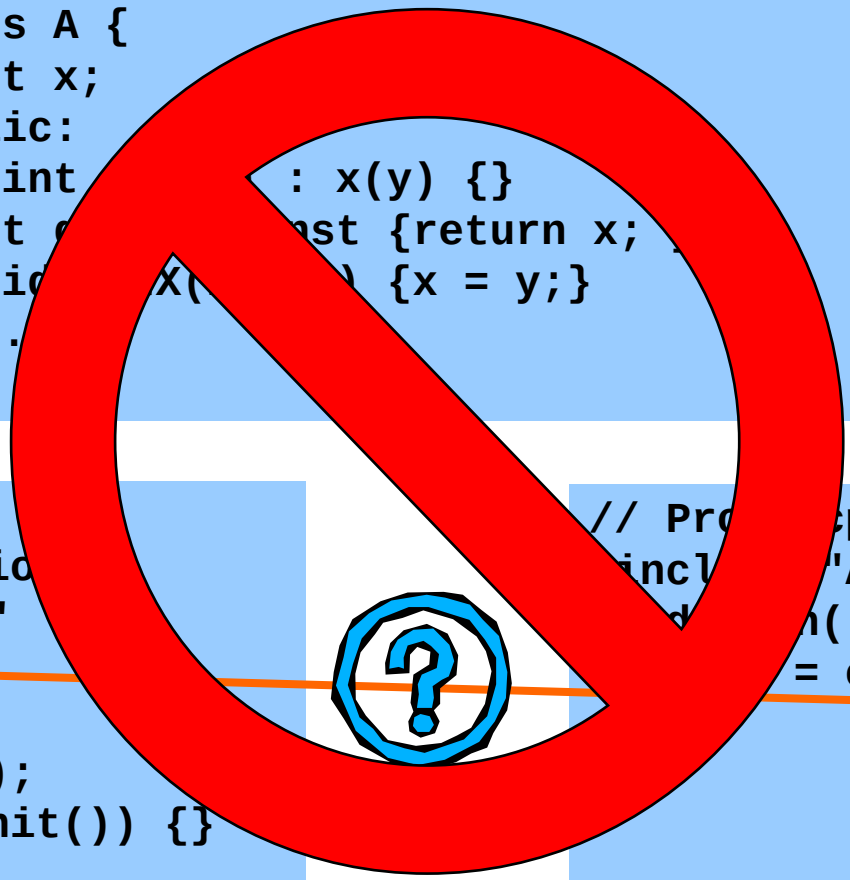
8.4 Khái niệm đối tượng phân tán

- Câu hỏi: Làm thế nào để gọi một hàm thành viên của một đối tượng viết trên C++ từ một chương trình khác?

```
// File A.h
class A {
  int x;
public:
  A(int y) : x(y) {}
  int getX() const {return x;}
  void setX(int y) {x = y;}
  ...
};
```

```
// Prog1.cpp
#include <conio.h>
#include "A.h"
A obj(5);
void main() {
   obj.setX(6);
   while (!kbhit()) {}
}
```

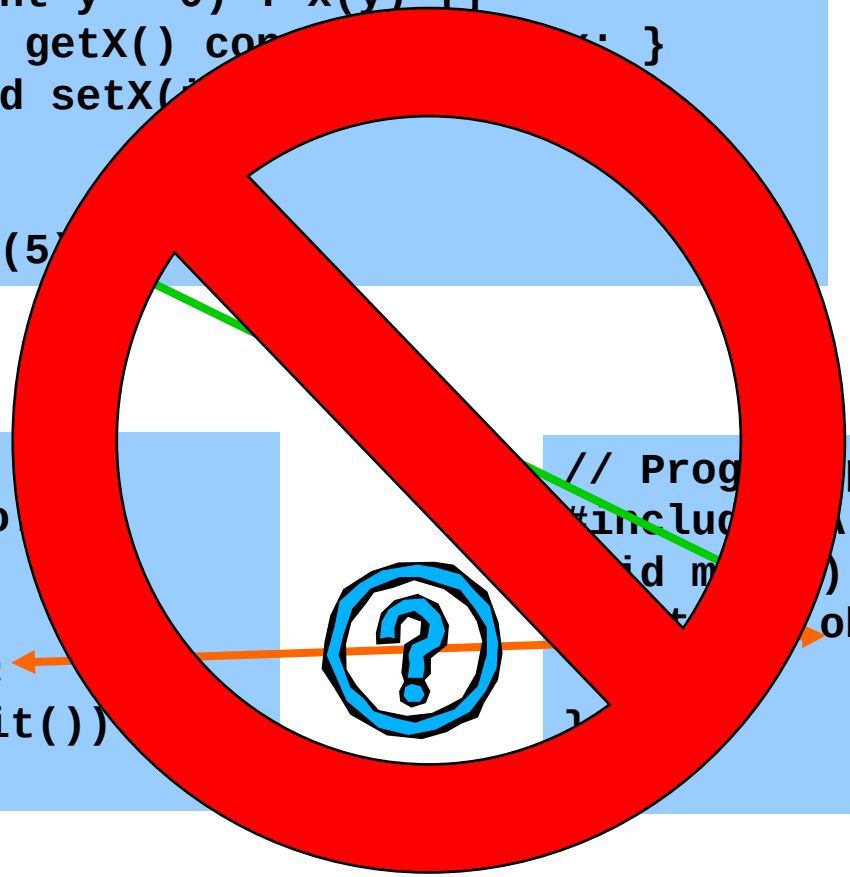
```
// Prog2.cpp
#include "A.h"
void main() {
   int x = obj.getX();
```



```
// File A.h
class A {
  int x;
public:
  A(int y = 0) : x(y) {}
  int getX() co
  void setX(i
  ...
};
A obj(5)
```

```
// Prog1.cpp
#include <conio
#include "A.h"
void main() {
   obj.setX(6);
   while (!kbhit())
}
```

```
// Prog
#inclu .h"
id m ) {
   obj.getX();
}
```



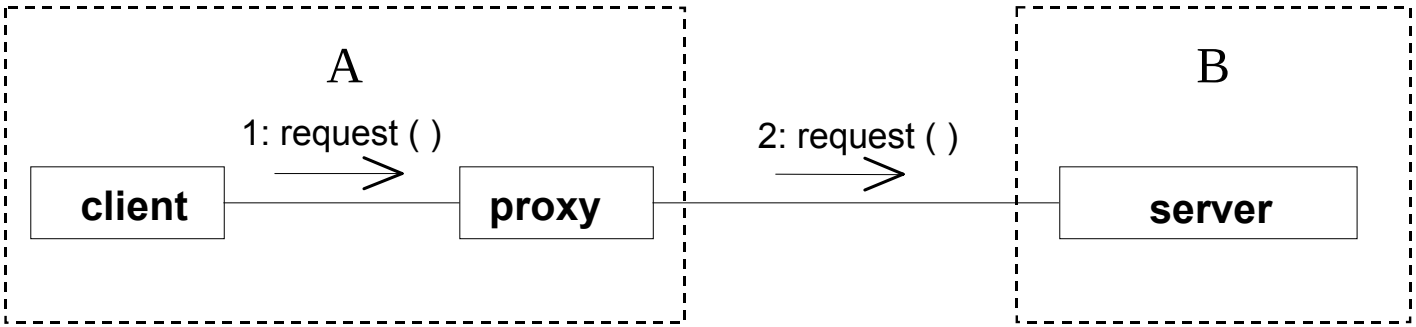
Đối tượng phân tán là gì?

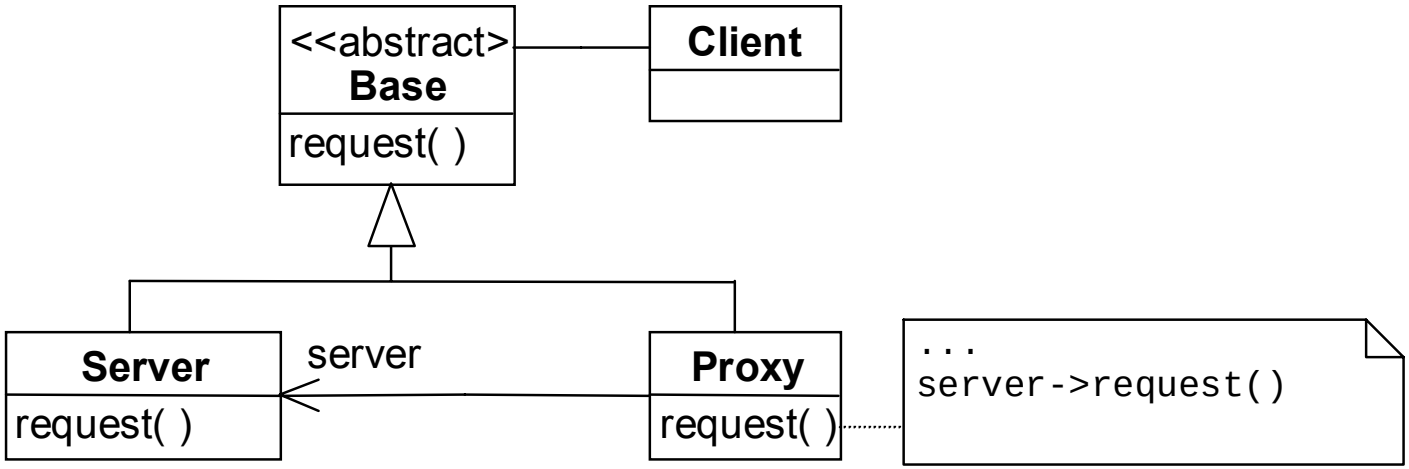
Đối tượng phân tán là các đối tượng phần mềm trong một hệ thống phân tán, có khả năng giao tiếp ngầm, có thể sử dụng từ xa (từ một chương trình khác, từ một nút mạng khác)

- Giống với đối tượng cổ điển
  - Có những đặc tính của đối tượng cổ điển (thuộc tính, phép toán, hành vi, trạng thái, căn cước)
- Khác với đối tượng cổ điển:
  - Không gắn với một ngôn ngữ lập trình
  - Không gắn với một nền cài đặt, nền mạng
  - Có thể tạo, hủy và gọi hàm từ xa

Mẫu thiết kế: Proxy

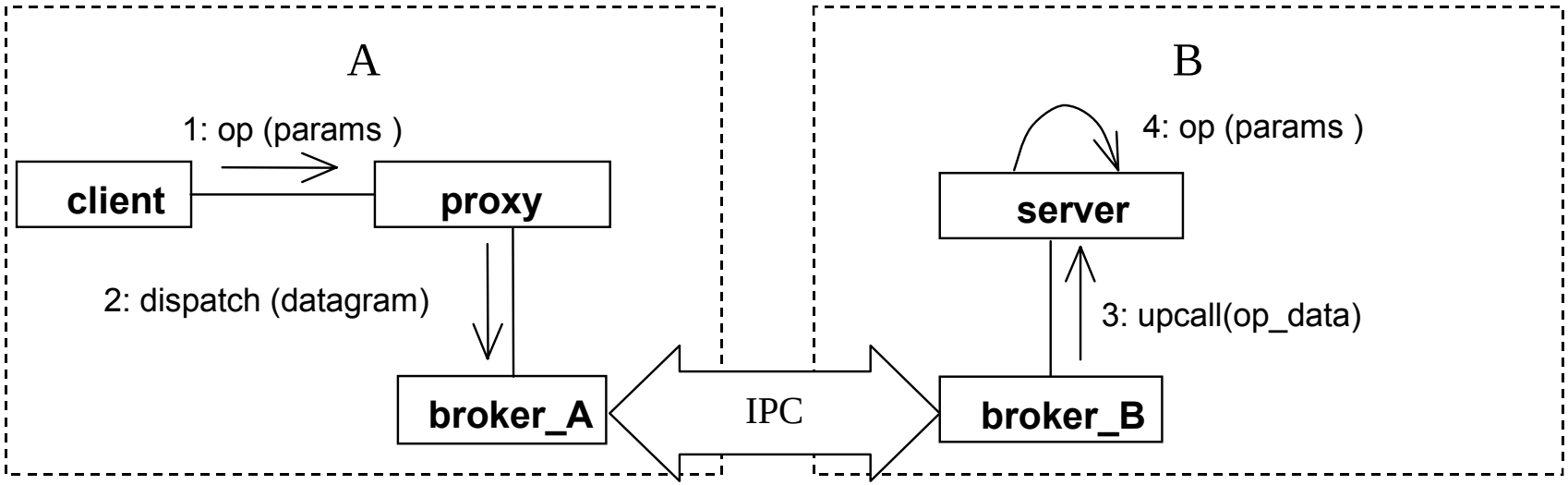
Sử dụng đối tượng từ xa thông qua một đại diện (Proxy)





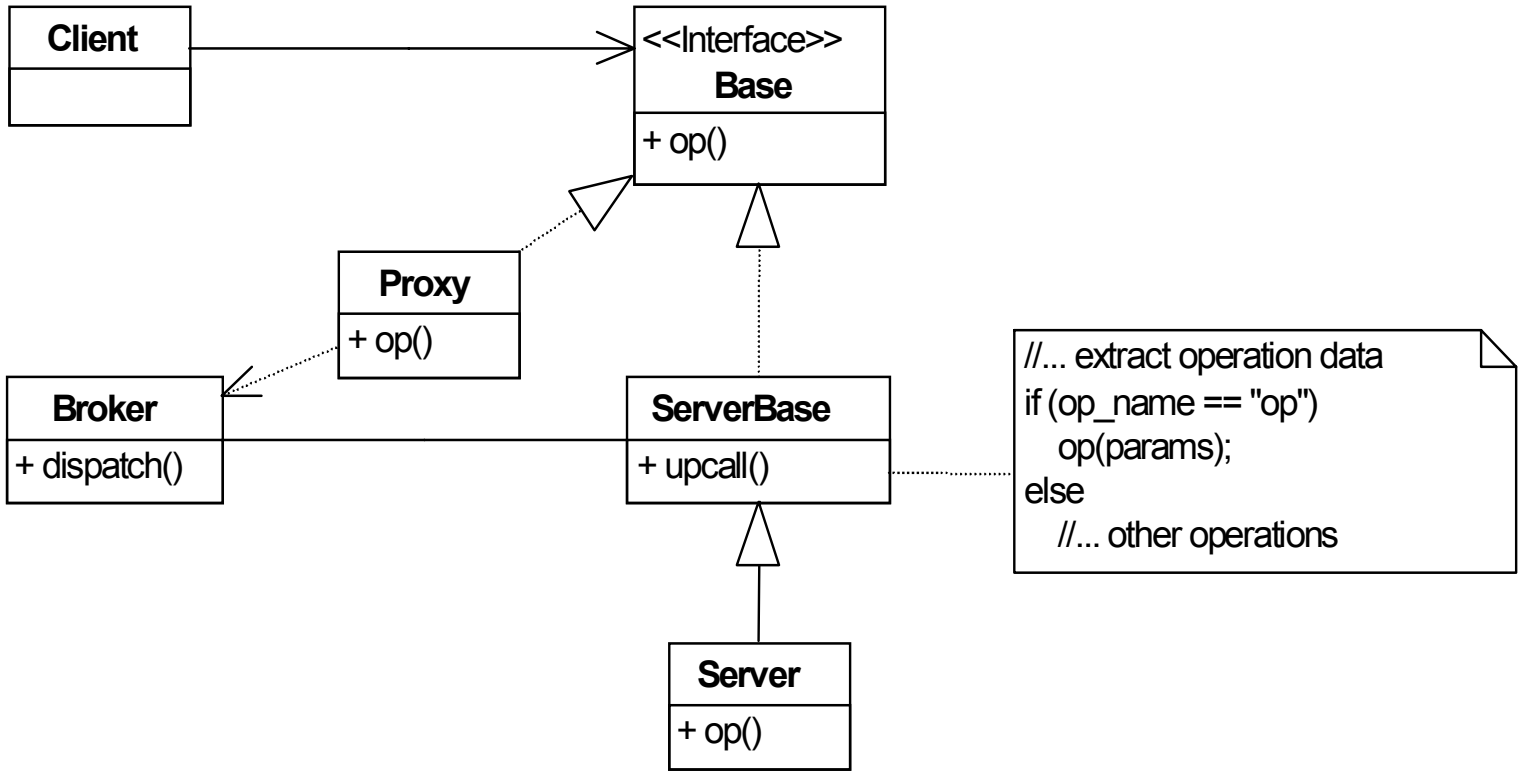
Mẫu thiết kế: Broker + Marshaling/Unmarshaling

Sử dụng đối tượng trung gian (Broker) để Proxy thực hiện giao tiếp với Server một cách trong suốt, không phụ thuộc cơ chế truyền thông cụ thể phía dưới



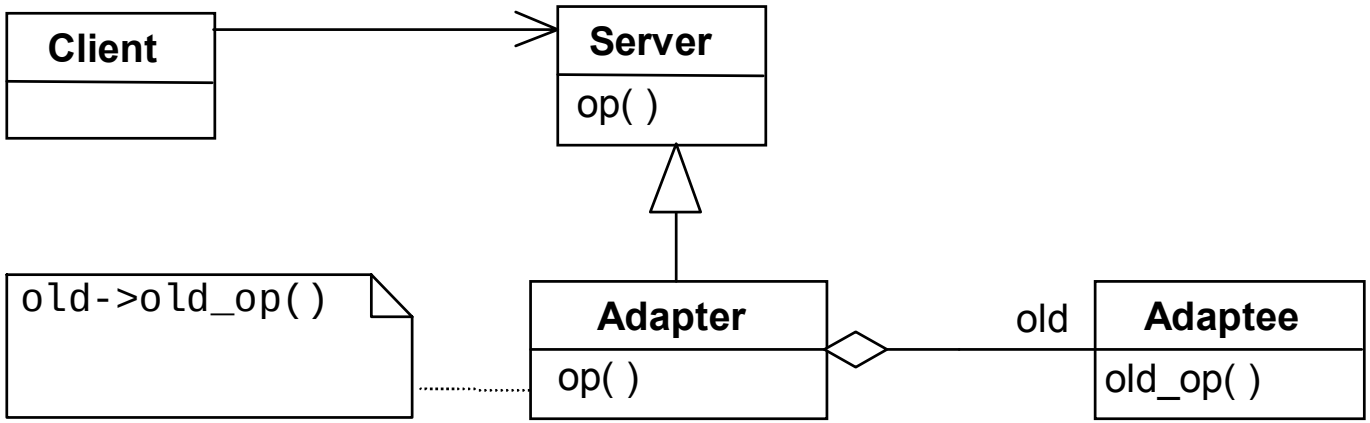
Marshaling: Đóng gói dữ liệu thành một bức điện, mã hóa căn cước đối tượng, tên hàm cần gọi và các tham số
Unmarshaling: Ngược lại với Marshaling.

Mẫu thiết kế: Broker + Marshaling/Unmarshaling



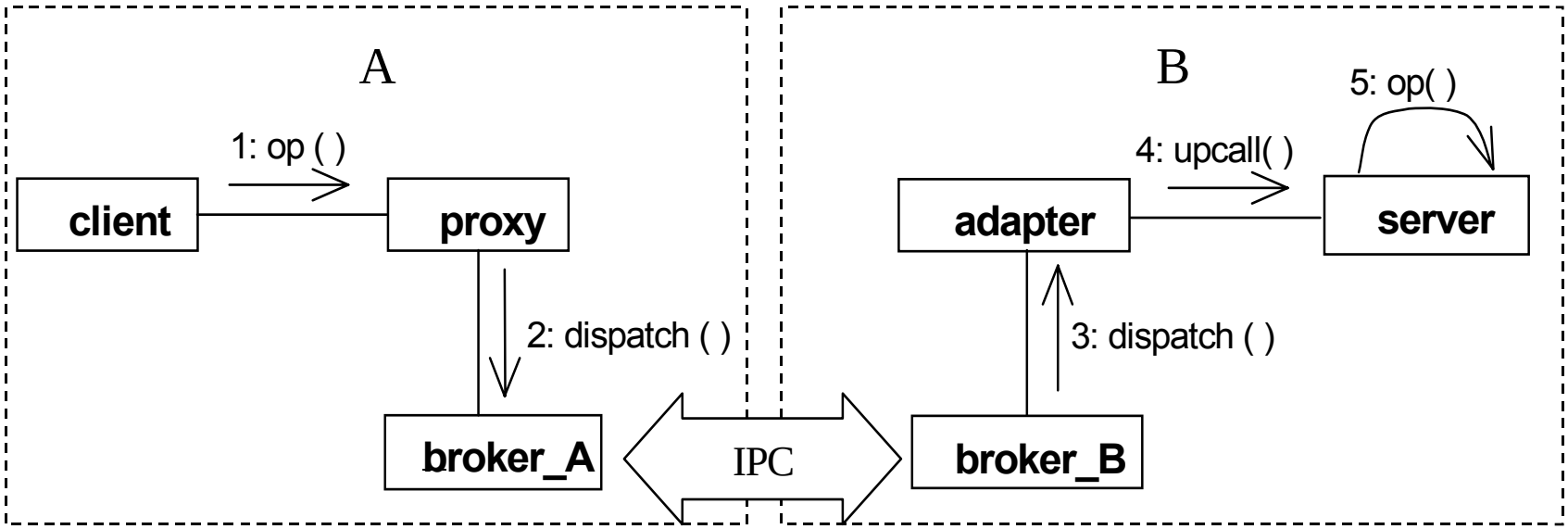
Mẫu thiết kế: Adapter

Sử dụng một đối tượng có sẵn thông qua giao diện thích ứng (Adapter)




```
class Adapter : public Server {
   Adaptee *old;
public:
  void op() { old->old_op(); }
  ...
};
```

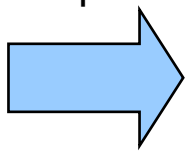
Kiến trúc tổng thể



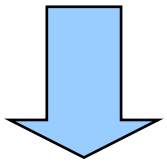
Mẫu thiết kế: Interface Mapping

```
// Server.idl
// Interface Definition
// Language
interface Server {
  bool op();
};
```

IDL-to-C++ Compiler



```
// Proxy.h
class Server_p {
public:
  boolean op() {
    // Marshaling Codes
  }
  ...
}
```



IDL-to-C++ Compiler

Broker

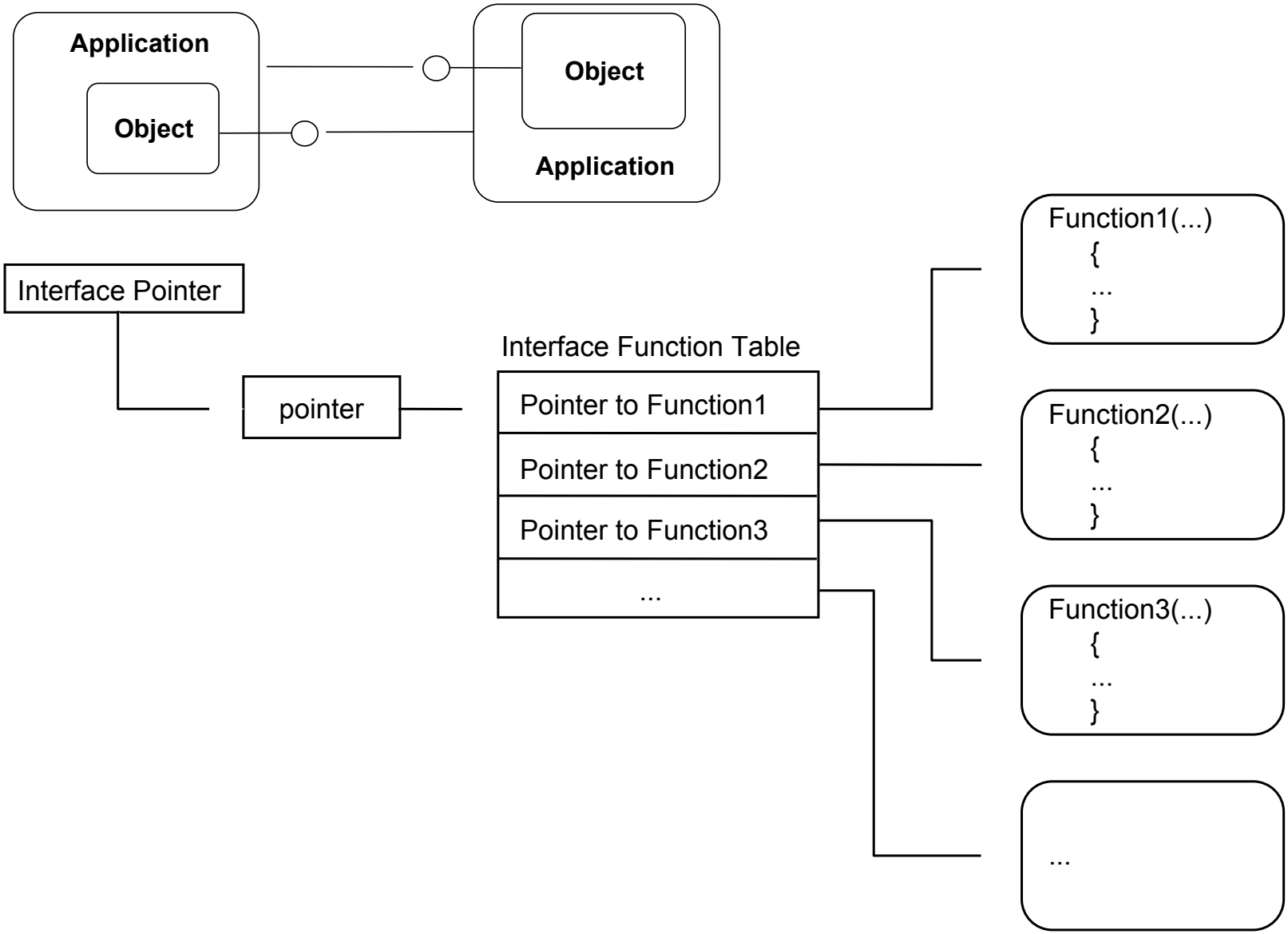
```
// ServerImp.cpp
class Server_s {
  virtual bool op()=0;
  virtual upcall() {
    ...
    op();
  }
};
class MyServer:
      public Server_s {
  bool op() { ... }
};
```

```
// Client Program
// Client.cpp
#include <Proxy.h>
void main() {
  Server_p *obj =
  NewObject("ServerID");
  bool result = obj->op();
  ...
  obj->DeleteObject();
}
```

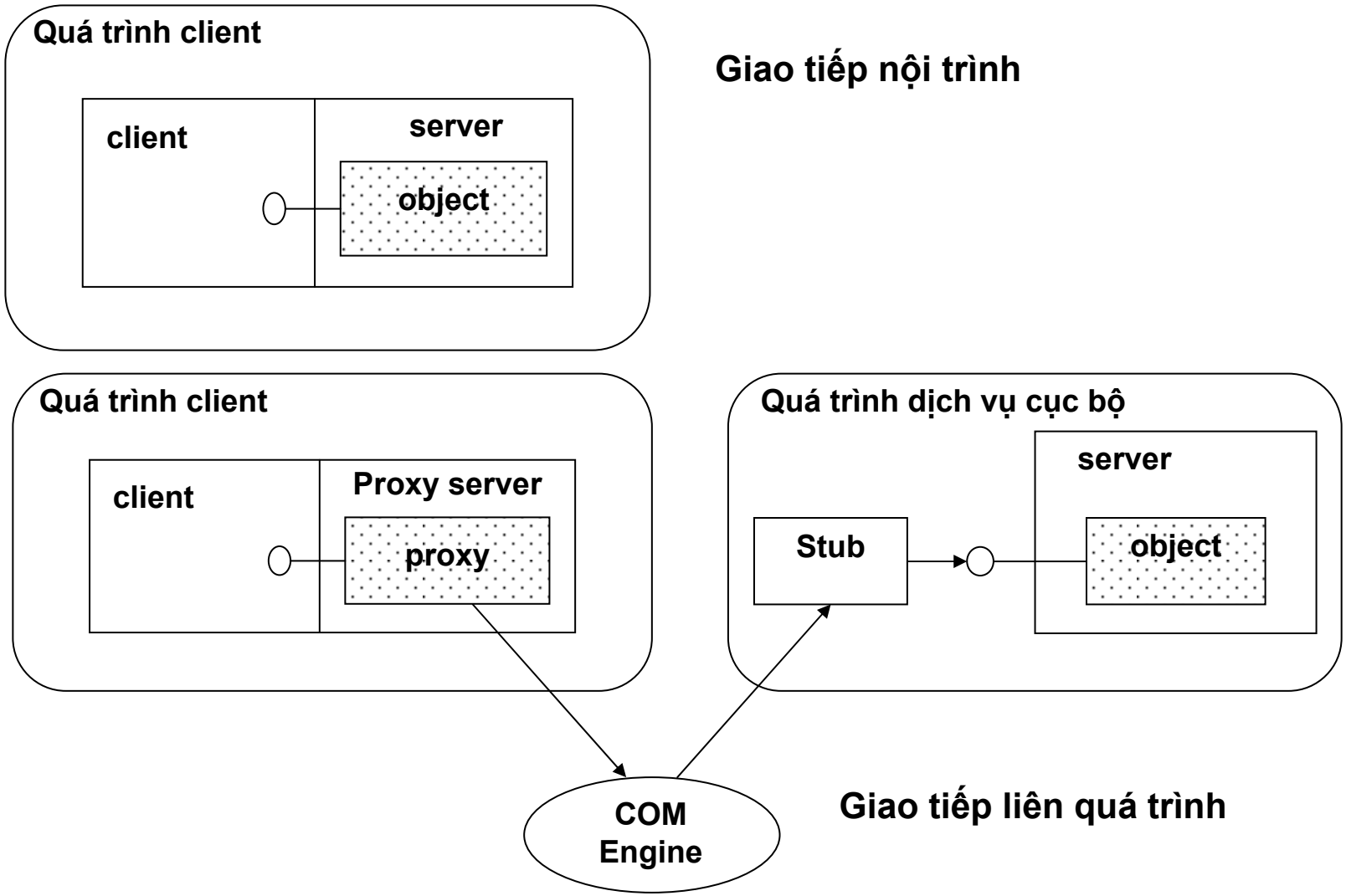
8.5 Mô hình COM và DCOM

- COM (Component Object Model)
  - Chuẩn của Microsoft, chủ yếu thực hiện trên nền Windows
  - Kiến trúc giao tiếp bậc cao giữa các thực thể phần mềm (đối tượng thành phần) trong hệ thống
  - Là nền tảng cho các công nghệ khác: OLE, ActiveX-Control, ASP, ADO, ...
  - Công nghệ then chốt trong các sản phẩm của Microsoft ngày nay
  - Hỗ trợ rất mạnh trong các sản phẩm phần mềm khác
- DCOM (Distributed COM)
  - Giao thức hỗ trợ giao tiếp với COM qua mạng
  - Kiến trúc đối tượng phân tán (so sánh với CORBA)

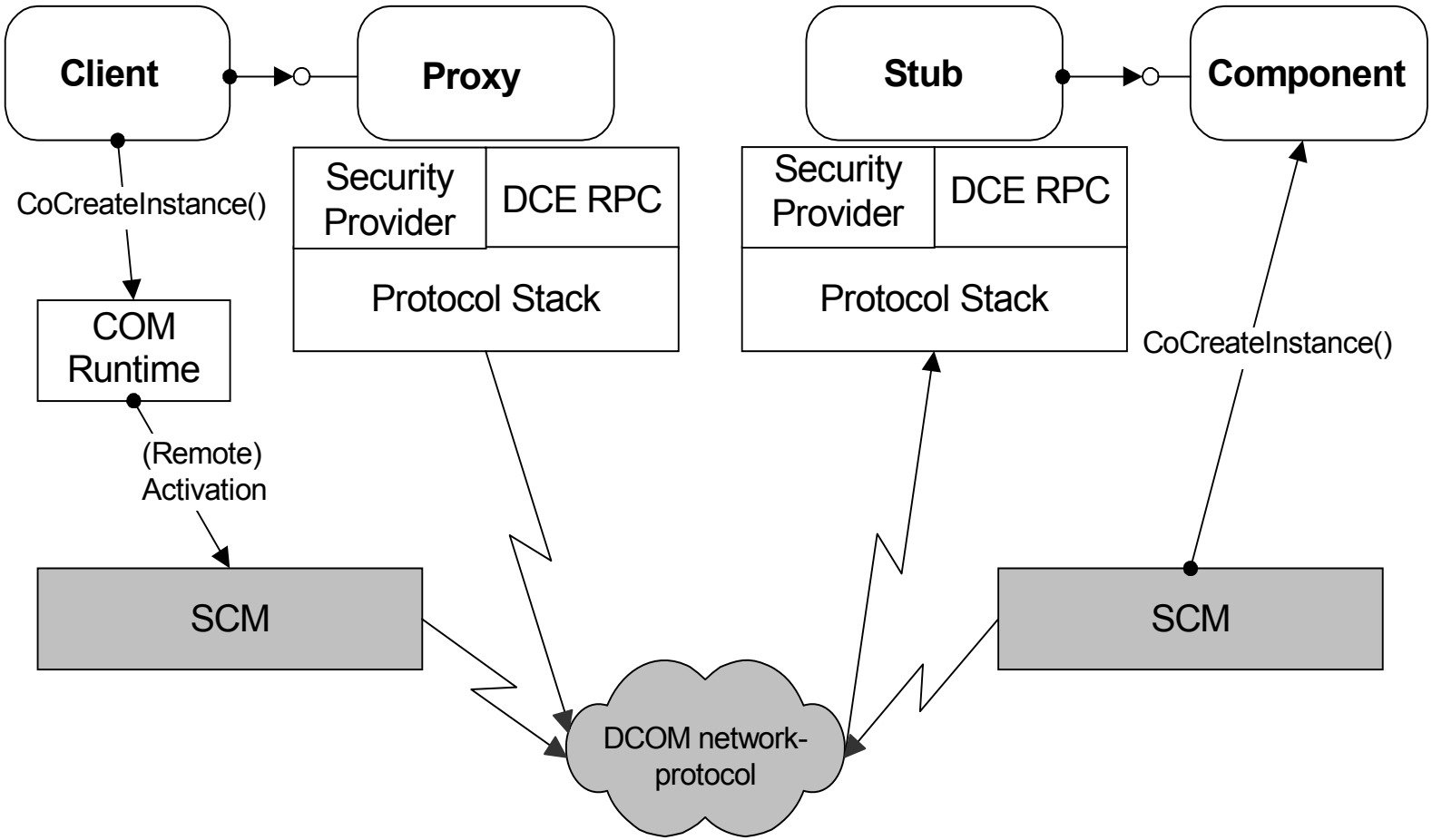
Đối tượng COM và giao diện COM



Giao tiếp với COM



Giao tiếp qua mạng với DCOM



8.6 Lập trình với COM/DCOM

- Tạo một đối tượng COM
- Sử dụng một đối tượng COM

Tạo một đối tượng COM

- Định nghĩa giao diện và căn cước đối tượng bằng IDL

```
/* COM-IDL */
[ object,
  uuid(793D8ABD-3E1B-11D3-A3E3-00A0C910AB98) ]
interface ISensor : IUnknown
{
// Eigenschaften
[propget] HRESULT status([out, retval] short *pVal);
[propget] HRESULT rangeMin([out, retval] double *pVal);
[propget] HRESULT rangeMax([out, retval] double *pVal);

// Methoden
HRESULT getValue([out, retval] double *pVal);
    // ...
};

[ uuid(793D8ABE-3E1B-11D3-A3E3-00A0C910AB98) ]
coclass Sensor
{
[default] interface ISensor;
};
```

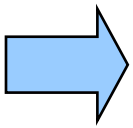
- Dịch sang C++ với IDL-Compiler

```
/* C++ */
class ISensor : public IUnknown
{
public:
virtual HRESULT STDMETHODCALLTYPE get_status(short *pVal) = 0;
virtual HRESULT STDMETHODCALLTYPE get_rangeMin(double *pVal) = 0;
virtual HRESULT STDMETHODCALLTYPE get_rangeMax(double *pVal) = 0;
virtual HRESULT STDMETHODCALLTYPE getValue(double *pVal) = 0;
... // Marshaling Codes
};
```

- Dẫn xuất lớp và thực thi đối tượng COM

```
/* C++ */
class Thermometer : public ISensor
{
public:
  virtual HRESULT get_status(short *pVal)     {...}
  virtual HRESULT get_rangeMin(double *pVal)  {...}
  virtual HRESULT get_rangeMax(double *pVal)  {...}
  virtual HRESULT getValue(double *pVal)      {...}
};
```

- Biên dịch mã thực thi đối tượng
- Đăng ký với hệ điều hành Windows trên các trạm cài đặt và trạm sử dụng



Các bước trên có thể thực hiện đơn giản với sự hỗ trợ của Visual C++ và ATL/COM Wizard

Sử dụng một đối tượng COM

```
/* C++ */
ISensor *pSensor;
HRESULT hr = CoCreateInstance(CLSID_Sensor, 0, CLSCTX_ALL,
  IID_ISensor,(void**)&pSensor);
if (SUCCEEDED(hr)) {
  double value = 0;
  pSensor->getValue(&value);
    //...
}
//...
pSensor->Release();
```

Hệ thống điều khiển phân tán

Chương 9: Chuẩn IEC 61131-3

13.09.06

Chương 9: Chuẩn IEC 61131-3

- Giới thiệu chung về IEC 61131
- Tiến trình chuẩn hóa IEC 61131
- Mô hình phần mềm
- Biến và kiểu dữ liệu
- Tổ chức chương trình
- Ngôn ngữ lập trình

IEC 61131 là gì?

- Tập chuẩn phần mềm quan trọng nhất cho các thiết bị điều khiển công nghiệp có khả năng lập trình (PLC, DCS, Soft PLC,...)
- Bao gồm nhiều phần:
  - Phần 1 (General Information)
  - Phần 2 (Equipment requirements)
  - **Phần 3 (Programming languages)**
  - Phần 4 (Guidelines for users)
  - Phần 5 (Communication)
  - Phần 7 (Fuzzy Control)
  - ...
- Hầu hết các hệ PLC và DCS hiện đại đều hỗ trợ chuẩn IEC 61131-3

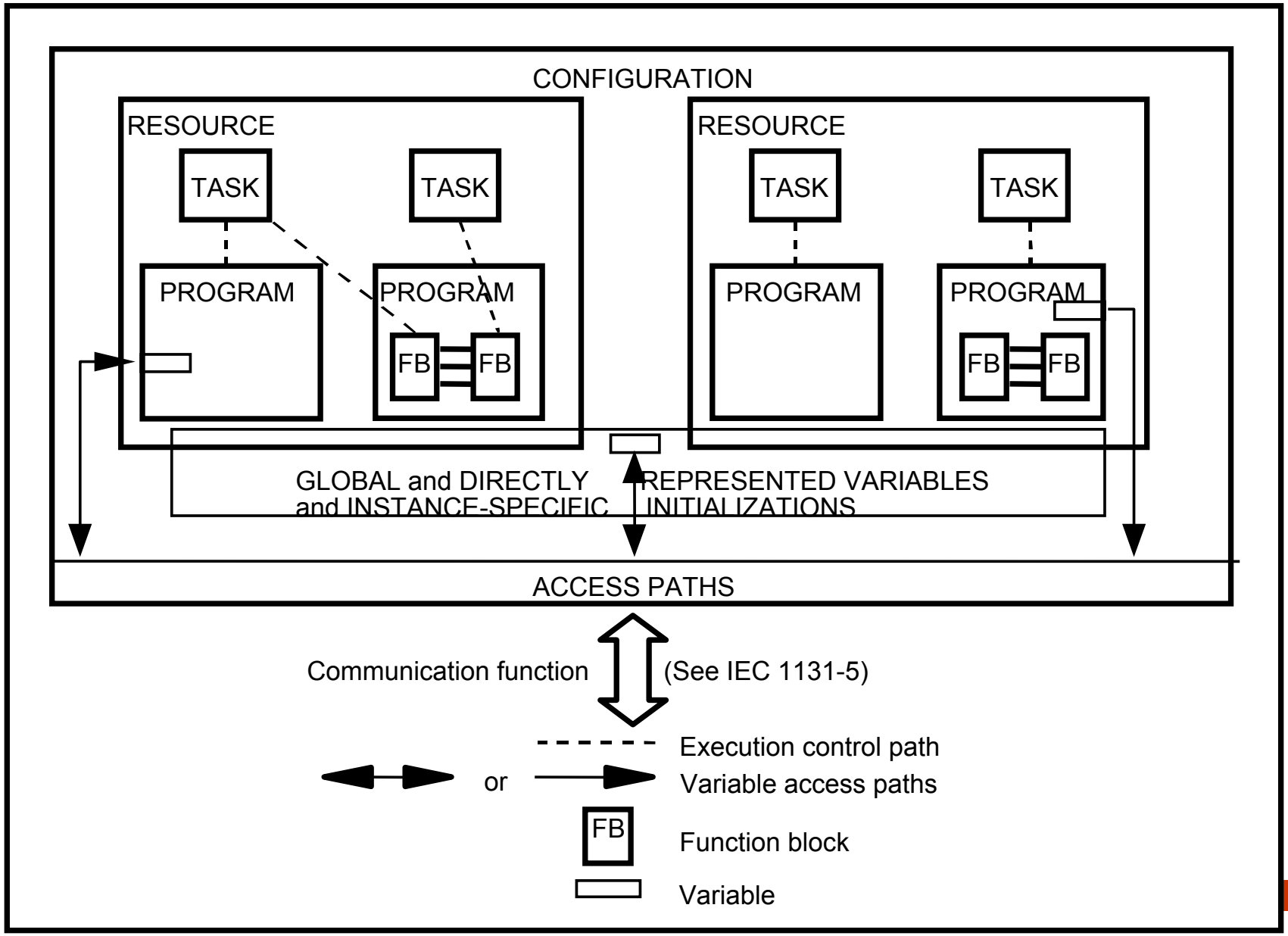
Tiến trình chuẩn hóa IEC 61131

- 1977: IEC 848
- 1979: Bắt đầu soạn bản thảo IEC 1131
- 1982: Hoàn thành bản thảo đầu tiên (5 nhóm làm việc)
- 1983: DIN 19239 PLC-Programming
- 1992: Chuẩn hóa quốc tế IEC 1131-1 và 1131-2
- 1993: Chuẩn hóa quốc tế IEC 1131-3
- 1995: Chuẩn hóa quốc tế IEC 1131-TR4
- 1994-1997: Đính chính IEC 1131-3 (Corrigendum)
- 1996-1999: Sửa đổi, bổ sung (Amendment)
- Từ 2000 -> IEC 61131-3 2nd Edition

Các tiến bộ của IEC 61131-3

- Các yếu tố cấu hình thống nhất (CONFIGURATION, TASK, RESOURCE), mô hình TASK và RESOURCE thích hợp cho nhiều hệ thống khác nhau
- Mô hình phần mềm thống nhất, hiện đại, với các khối tổ chức chương trình hợp lý (PROGRAM, FUNCTION BLOCK, FUNCTION)
- Các ngôn ngữ lập trình thống nhất, phát triển trên cơ sở chuẩn hóa các ngôn ngữ hiện có quen thuộc
- 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.

Mô hình phần mềm



Các yếu tố cấu hình

- Cấu hình (CONFIGURATION):
  - Tương ứng cho cả hệ PLC, có thể gồm nhiều CPU ghép nối
  - Mỗi PLC tại một thời điểm bất kỳ chỉ có một cấu hình.
  - Bao gồm một hay nhiều tài nguyên
- Tài nguyên (RESOURCE)
  - Tương ứng cho một CPU với các vào/ra và HMI (đơn giản) tương ứng
  - Bao gồm một hoặc nhiều chương trình hoạt động dưới sự điều khiển của một hoặc nhiều tác vụ
- Tác vụ (TASK)
  - Tác vụ tuần hoàn (*Periodic Task*)
  - Tác vụ sự kiện, task đơn (*Event Task, Single Task*)
  - Tác vụ rỗi (*Idle Task*)
- Biến toàn cục (Global Variables)
- Lối truy nhập (Access Path)

Các kiểu dữ liệu cơ bản

Kiểu Bool	BOOL
Kiểu nguyên có dấu	SINT, INT, DINT, LINT, INT
Kiểu nguyên dương	USINT, UINT, UDINT, ULINT
Số thực	REAL, LREAL
Khoảng thời gian	TIME
Ngày tháng	DATE
Thời gian trong ngày	TIME_OF_DAY, TOD
Ngày tháng và thời gian	DATE_AND_TIME, DT
Chuỗi ký tự	STRING, WSTRING
Chuỗi bit	BYTE, WORD, DWORD, LWORD

Các kiểu dữ liệu dẫn xuất

- Dẫn xuất trực tiếp:

  TYPE RU_REAL : REAL ; END_TYPE

- Liệt kê:

  TYPE ANALOG_SIGNAL_TYPE : (SINGLE_ENDED, DIFFERENTIAL) ;
  END_TYPE

- Dãy con:

  TYPE ANALOG_DATA : INT (-4095..4095) ; END_TYPE

- Mảng:

  TYPE ANALOG_16_INPUT_DATA : ARRAY [1..16] OF ANALOG_DATA ;
  END_TYPE

- Cấu trúc:

  TYPE ANALOG_CHANNEL_CONFIGURATION: STRUCT
  RANGE : ANALOG_SIGNAL_RANGE ;
  MIN_SCALE : ANALOG_DATA ;
  MAX_SCALE : ANALOG_DATA ;
  END_STRUCT;

Các kiểu dữ liệu tổng quát

```
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
```

Khai báo biến

- Kiểu của biến:
  - Kiểu cơ bản,
  - Kiểu dẫn xuất,
  - Kiểu tổng quát
  - Khối chức năng,
  - Khối chương trình
- Từ khóa
  - Bắt đầu với VAR, VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR_EXTERNAL, VAR_GLOBAL, VAR_ACCESS, VAR_TEMP hoặc VAR_CONFIG
  - Có thể kèm theo thuộc tính RETAIN, NON_RETAIN, CONSTANT, AT
  - Kết thúc với END_VAR

Ký hiệu biến trực tiếp

- Tiền tố
  - I — Biến đầu vào (Input)
  - Q — Biến đầu ra (Output)
  - M — Biến nhớ (Memory)
  - X hoặc không ghi — 1 bit, mặc định là BOOL
  - B — 8 bit, mặc định là BYTE
  - W — 16 bit, mặc định là WORD
  - D — 32 bit, mặc định là DWORD
  - L — 64 bit, mặc định là LWORD
- Ví dụ:
  - %QX75, %Q75 — Bit ra vị trí 75
  - %IW215 — Từ vào vị trí 215
  - %QB7 — Byte vào vị trí 7
  - %MD48 — Từ đúp vào tại vị trí ô nhớ 48
  - %IW2.5.7.1 — Từ vào kênh 1, slot 7, rack 5, station 2
  - %Q* — Đầu vào chưa định vị trí

Ví dụ khai báo biến

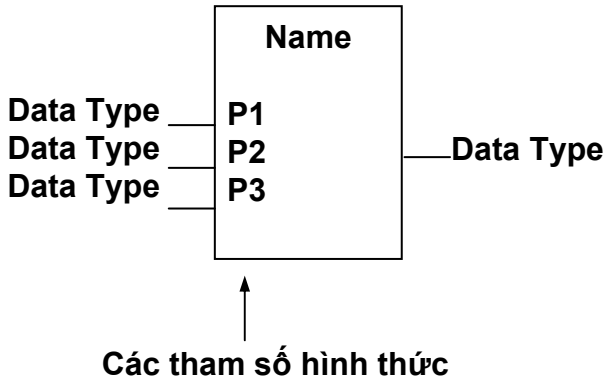
```
VAR RETAIN
   AT %IW6.2     : WORD;
   AT %MW6       :INT;
END_VAR
VAR_GLOBAL
   LIM_SW_S5 AT %IX27    :BOOL = TRUE;
   CONV_START AT %QX25 :BOOL;
   TEMPERATURE AT %IW28: INT;
   C2 AT %Q* : BYTE;
END_VAR
VAR INARY AT %IW6 :ARRAY [0..9] OF INT; END_VAR
VAR
   CONDITION_RED : BOOL = 1;
   IBOUNCE : WORD = 16#FF00;
   MYDUB :DWORD;
   AWORD, BWORD, CWORD : INT = 8;
   MYSTR: STRING[10];
END_VAR
```

Các khối tổ chức chương trình (POU)

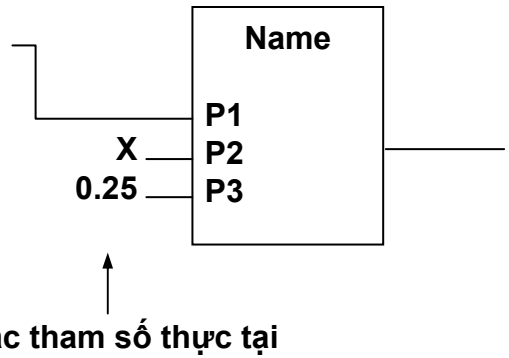
- Hàm (FUNCTION)
  - Tương tự hàm PASCAL, có thể nhiều vào, chính xác một ra
  - Như một hệ tĩnh, không có trạng thái
  - Có giá trị sử dụng lại
- Khối chức năng (FUNCTION BLOCK)
  - Tương tự lớp trong lập trình HĐT, có thể có nhiều đầu ra
  - Như một hệ động, có trạng thái
  - Phân biệt giữa kiểu và thể nghiệm theo ngữ cảnh
  - Có giá trị sử dụng lại
- Chương trình (PROGRAM)
  - Về cơ bản giống như khối chức năng
  - Truy cập được các biến trực tiếp (biến vào/ra, biến nhớ trực tiếp) và các biến toàn cục
  - Không có giá trị sử dụng lại

Khai báo và sử dụng hàm

KHAI BÁO HÀM



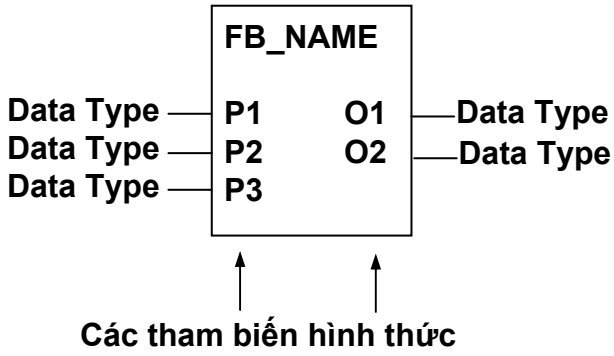
SỬ DỤNG HÀM



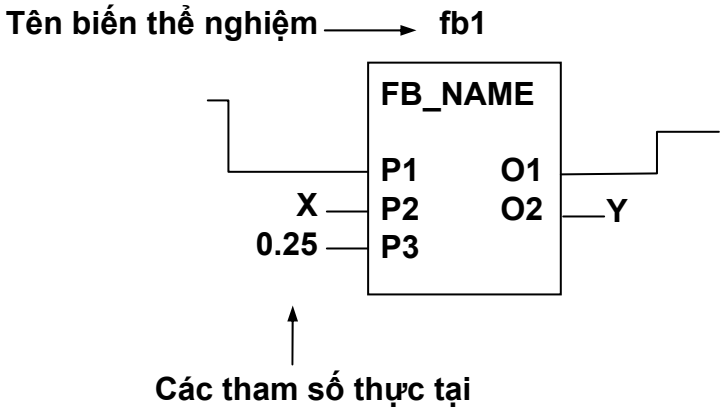

```
(* Khai báo hàm *)
FUNCTION fct1 : REAL
  VAR_INPUT
    a, b: REAL;
    c  : REAL:= 1.0;
  END_VAR
  fct1 := a*b/c;
END_FUNCTION

(* Gọi hàm *)
...
y := fct1(a:= x, b:= 2.0);
...
```

Khai báo và sử dụng khối chức năng

KHAI BÁO KHỐI CHỨC NĂMG



SỬ DỤNG KHỐI CHỨC NĂNG




```
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
```

Các ngôn ngữ lập trình

- Các ngôn ngữ lập trình văn bản (textual languages):
  - Instruction List (IL) : Một dạng hợp ngữ
  - Structured Text (ST): Giống PASCAL
  - Các thành phần SFC có thể sử dụng phối hợp
- Các ngôn ngữ đồ họa (graphical languages):
  - Ladder Diagram (LD): Giống mạch rơ le
  - Funtion Block Diagram (FBD): Giống mạch nguyên lý
  - Sequential Funtion Charts (SFC): Xuất xứ từ mạng Petri/Grafcet

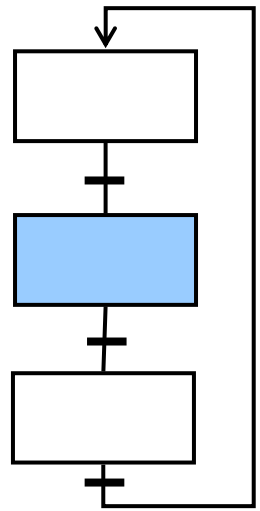
Instruction List (IL)

```
LD    A
ANDN  B
ST    C
```

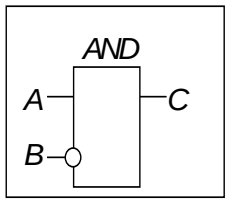
Structured Text (ST)

```
C:= A AND NOT B
```

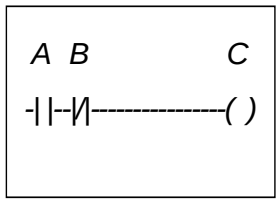
Sequential Function Charts



Function Block Diagram (FBD)



Ladder Diagram (LD)



Các ngôn ngữ văn bản: IL và ST

- Các yếu tố chung:

```
TYPE...END_TYPE
VAR...END_VAR
VAR_INPUT...END_VAR
VAR_OUTPUT...END_VAR
VAR_IN_OUT...END_VAR
VAR_EXTERNAL...END_VAR
VAR_TEMP...END_VAR
VAR_ACCESS...END_VAR
VAR_GLOBAL...END_VAR
VAR_CONFIG...END_VAR
FUNCTION ... END_FUNCTION
FUNCTION_BLOCK...END_FUNCTION_BLOCK
PROGRAM...END_PROGRAM
STEP...END_STEP
TRANSITION...END_TRANSITION
ACTION...END_ACTION
```

Instruction List (IL)

Cú pháp câu lệnh

```
NHÃN     TOÁN TỬ/HÀM     TOÁN HẠNG        CHÚ THÍCH
START:   LD              %IX1             (* PUSH BUTTON *)
         ANDN            %MX5             (* NOT INHIBITED *)
         ST              %QX2             (* FAN ON *)
         LD              2#00010001
         ST              %QB3
```

Lệnh phức hợp

```
AND(                          AND(    %IX1
LD      %IX1      hoặc        OR      %IX2
OR      %IX2                  )
)
```

Accu đa năng: chứa "giá trị tức thời"

- Thích hợp với các kiểu dữ liệu khác nhau
- Mã thực hiện cụ thể do trình biên dịch tạo ra
- Chuẩn không qui định về các cờ trạng thái accu

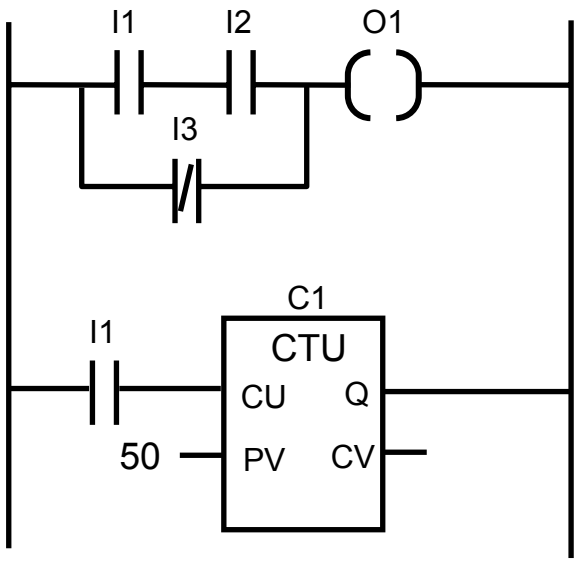
Structured Text (ST)

- Ngôn ngữ hoàn toàn mới, dựa trên PASCAL/C
- Ưu điểm: Đơn giản, mạnh
  - Lập trình ở mức cao
  - Dễ mô tả nhiệm vụ điều khiển
  - Lập trình có cấu trúc
  - Các lệnh điều khiển chương trình (IF, WHILE, FOR,..)
- Nhược điểm: Mã chậm, lớn
  - Phụ thuộc nhiều vào chất lượng của trình biên dịch
  - Không phải hệ PLC/DCS nào cũng hỗ trợ
- Lựa chọn hay không?
  - Qui mô ứng dụng
  - Tỉ lệ đầu tư phần cứng/phát triển phần mềm
  - Điều khiển đơn giản hay điều khiển cao cấp

Các ngôn ngữ đồ họa: LD, FBD và SFC

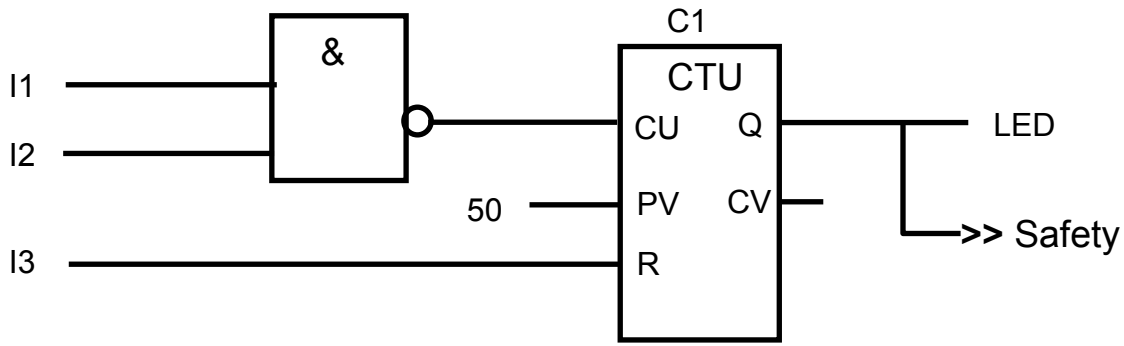
- Các yếu tố chung:
  - Ký hiệu mô tả các khối và đường nét:
  - Hướng của các dòng trong mạng
    - Power flow
    - Signal flow
    - Activity flow
  - Đánh giá mạng (network evaulation)
  - Các yếu tố điều khiển thực thi
    - Các ký hiệu nhảy
    - Các ký hiệu kết thúc
- Lựa chọn ngôn ngữ phù hợp:
  - LD cho mạch điều khiển logic
  - FBD cho điều khiển tương tự (ĐK quá trình) và điều khiển logic
  - SFC cho điều khiển trình tự, phối hợp sử dụng LD và FBD

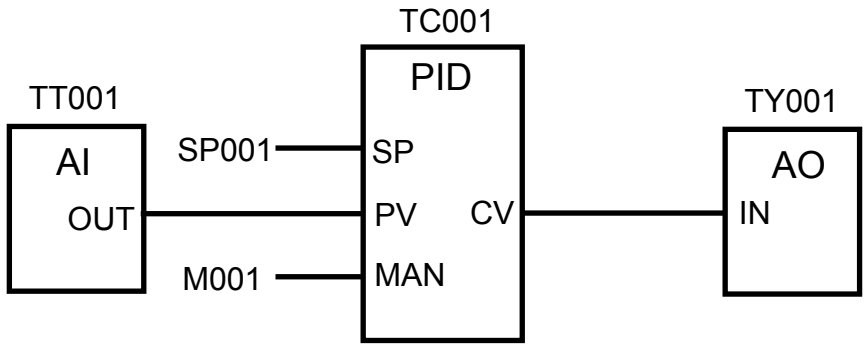
Ladder Diagram



| |	Tiếp điểm thường mở (NO)
|/|	Tiếp điểm thường đong (NC)
|P|	Tiếp điểm nhận biết sườn xung lên
|N|	Tiếp điểm nhận biết sườn xung xuống
( )	Cuộn dây (đầu ra)
(/)	Cuộn dây âm (đầu ra nghịch đảo)
(S)	Cuộn dây đặt
(R)	Cuộn dây xoá
(P)	Cuộn dây cảm nhận sườn xung lên
(N)	Cuộn dây cảm nhận sườn xung xuống

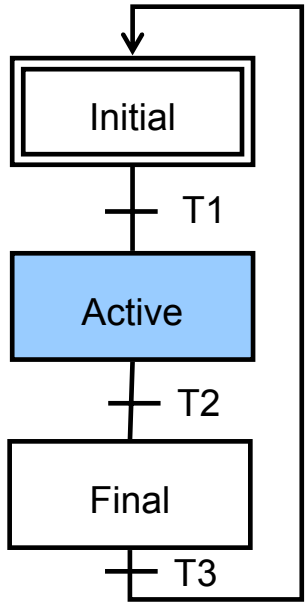
Function Block Diagram



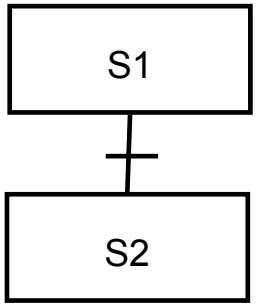


Sequential Function Chart (SFC)

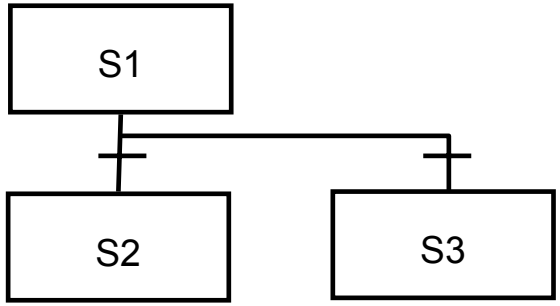
- Step: Một bước thực hiện trong điều khiển trình tự
  - Có thể bao gồm nhiều hành động đi kèm
  - Có ít nhất một bước tích cực
  - Trạng thái hệ thống được xác định qua các bước tích cực
- Transition: Chuyển tiếp, được thực hiện khi điều kiện chuyển tiếp thỏa mãn
  - Lập trình bằng ST, FBD, LD hoặc IL
- Action: Hành động đi với một bước
  - Nằm trong một "Action Block"
  - Được kiểm soát thực thi qua các "Qualifier"
  - Lập trình bằng ST, FBD, LD hoặc IL



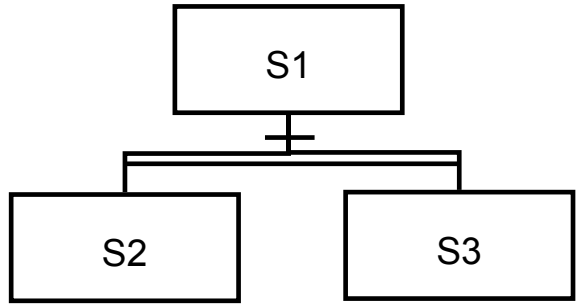
Các loại chuyển tiếp SFC



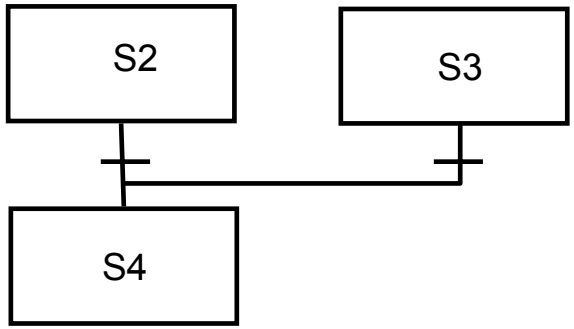
a) Đơn giản



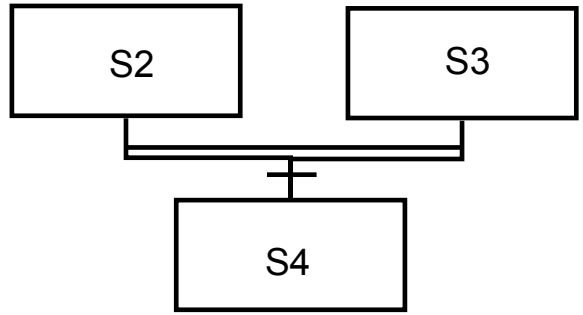
b) Phân nhánh cạnh tranh
(phân nhánh OR)



c) Phân nhánh song song
(phân nhánh AND)



d) Chuyển tiếp lựa chọn
Kết hợp kiểu OR



e) Chuyển tiếp đồng bộ
(Kết hợp kiểu AND)