MÔ HÌNH KHÁI NIỆM CHO HỆ THỐNG THU THẬP DỮ LIỆU VÀ ĐIỀU KHIỂN PHÂN BỐ

Trương Đình Châu, Nguyễn Đức Thành, Lương Văn Lăng
Trường Đại học Bách khoa, ĐHQG-HCM
(Bài nhận ngày 09 tháng 04 năm 2007, hoàn chỉnh sửa chữa ngày 21 tháng 01 năm 2008)

TÓM TẮT: *Một trong những thành phần của hệ thống điều khiển đa cấp được kể đến là hệ thống thu thập dữ liệu và điều khiển giám sát (SCADA – Supervisory Control And Data Acquisition). Thông qua việc đánh giá, phân tích các tính chất, ưu, khuyết điểm của các hệ thống SCADA trong thị trường tự động hóa công nghiệp bài báo đã xây dựng các khái niệm cơ bản để làm nền tảng cho việc xây dựng mô hình cấu trúc phần mềm thu thập dữ liệu và điều khiển, hành vi và sự tương tác giữa các thành phần trong mô hình. Cuối cùng, bài báo liệt kê các đặc tính mở của mô hình đặt ra.*

1. XU HƯỚNG PHÁT TRIỂN CỦA CÁC PHẦN MỀM THU THẬP DỮ LIỆU VÀ ĐIỀU KHIỂN

Với sự phát triển của lý thuyết và công nghệ thông tin, ngày nay các thành phần trong hệ thống thu thập dữ liệu và điều khiển (Data Acquisition And Control – DAQ&C) như HMI (Human-Machine Interface), SCADA (Supervisory Control And Data Acquisition), OPC (OLE for Process Control) Server đều có xu hướng được cải thiện lại (đối với những thành phần đã tồn tại) hoặc được thiết kế (đối với những thành phần mới) theo định hướng đối tượng [1] và cụ thể hơn nữa là hướng Component [2, 3, 4]. Tức là hệ thống thu thập dữ liệu và điều khiển được cấu thành từ một loạt tập hợp các Component. Với cấu hình này, các hệ thống DAQ&C sẽ được thể hiện mình là hệ thống mở (open system) và thừa kế tất cả những ưu điểm của phân tích, thiết kế và lập trình hướng đối tượng, hướng Component.

Hệ thống mở là hệ thống được hiểu rằng nếu đối với nó được xác định và mô tả bởi các dạng dữ liệu (data format) và giao diện (interface) để cho phép kết nối nó với các Component không phụ thuộc ở phía bên ngoài. Các hệ thống SCADA hiện đại tự bản thân nó được cấu thành từ nhiều Component khác nhau và cung cấp những interface cần thiết cho việc kết nối với các Component do các các nhà cung cấp khác sản xuất (third-party). Mỗi interface cung cấp một tập hợp các hàm truy cập (function) thực hiện những phần công việc khác nhau của interface. Ví dụ minh họa sự tương tác của các Component được thể hiện qua hình vẽ 1.

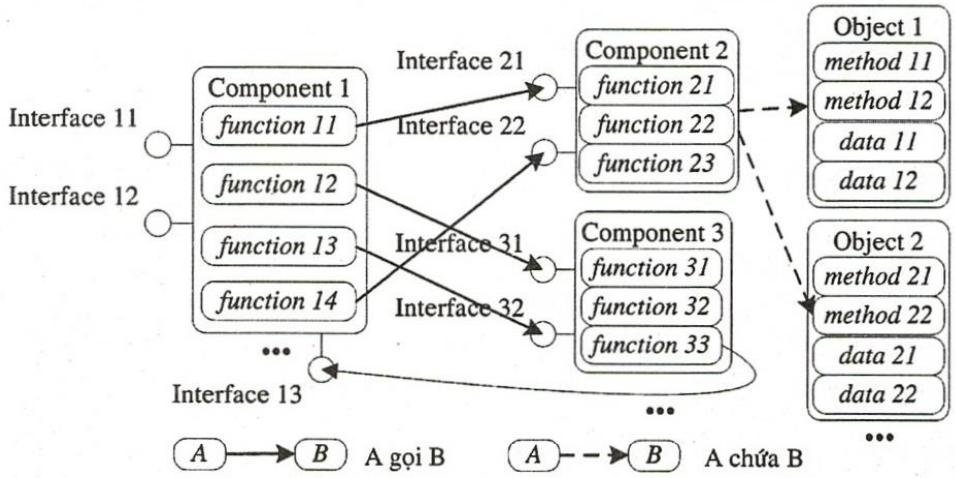
Các thành phần trong hình 1 được tác giả phân loại, xây dựng và phân tích rỏ hơn trong phần 2.

Tính chất mở của hệ thống SCADA là bài toán quan trọng nhất của nhà cung cấp hệ thống SCADA đó. Thực tế, tính chất mở của hệ thống SCADA có nghĩa là khả năng truy cập được đến những chi tiết kỹ thuật hệ thống (specification) để gọi những dịch vụ hệ thống khác nhau. Ví dụ như khả năng truy cập đến những hàm đồ họa, hàm làm việc với cơ sở dữ liệu, hàm thu thập dữ liệu và điều khiển đối tượng, v.v....

Tuy nhiên thực tế cho thấy rằng đa số các hệ thống SCADA đều có nguồn gốc là bản thiết kế đầu tiên của chúng không định hướng theo các đối tượng, các Component mà là áp dụng các phương pháp truyền thống như thiết kế, lập trình theo modular, thực hiện từ trên xuống, ... và đã được thừa kế từ các phiên bản 16 bit (theo dòng Microsoft Windows), cho nên ngày nay các hệ thống SCADA đó khi càng được mở rộng thì càng cồng kềnh. Các nhà sản xuất cố gắng phát triển hệ thống SCADA của mình trên mọi phương diện, trên mọi chức năng, nhưng những

thành phần của chúng, những module của chúng không được đồng nhất về mặt cấu tạo và tương tác, dẫn đến khó khăn trong vấn đề học tập và thao tác của người sử dụng (kỹ sư lập trình). Các hệ thống SCADA như Intouch (Wonderware, USA), Citect (Ci Technologies, Australia) được phát triển rất mạnh trên thị trường thế giới và Việt Nam. Những hệ thống SCADA này rất phong phú và chúng đã bao quát gần hết các công nghệ mới. Tuy nhiên chúng không thể tránh khỏi sự quá cồng kềnh, thiếu đồng nhất, gây phiền hà cho người sử dụng.

Vấn đề đặt ra trong bài báo là xây dựng một mô hình (model) cơ bản và trên cơ sở đó có thể xây dựng lên một hệ thống DAQ&C phức tạp. Để xây dựng model cơ bản này, trước hết chúng ta xem xét những khái niệm đặt ra bởi tác giả, thành phần và chức năng cơ bản trong một hệ thống DAQ&C (chỉ đề cập đến phần mềm).



Hình 1. Thí dụ về cấu trúc của một mảng phần mềm điều khiển hiện đại

2.XÂY DỰNG CÁC KHÁI NIỆM VÀ ĐỐI TƯỢNG CƠ BẢN

Cấu trúc của hệ thống (phần mềm) thu thập dữ liệu và điều khiển phân bố theo định hướng Component được cấu tạo từ nhiều Component khác nhau (xem hình 1). Component, theo tác giả, là một khối cung cấp các interface khác nhau cho các Component khác. Interface theo quan niệm của COM (Component Object Model) thì chứa các function để thực hiện các chức năng khác nhau của interface đó. Như vậy, các function có chức năng giống nhau gom lại và nằm trong một interface. Component chứa trong mình các đối tượng (object). Object theo lý thuyết thiết kế và lập trình hướng đối tượng thì chứa trong mình các hàm (method) và dữ liệu (data).

Chúng ta bắt đầu bằng một Component trống rỗng và lần lượt xây dựng các thành phần cho Component này.

Cấu tử phổ biến và cơ bản nhất trong một chiến lược (strategy – đối với GeniDAQ, Advantech) hay ứng dụng (application – đối với Intouch, Wonderware) là thông số quá trình công nghệ, hay còn được gọi là Tag (T). Bản chất của Tag là trừu tượng hóa thông số quá trình công nghệ. Cụ thể hơn Tag là nguồn thông tin như:

• dữ liệu thời gian thực từ các bộ cảm biến – nhiệt độ, áp suất, dòng chảy, mức chất lỏng, v.v...;

• trạng thái của các thiết bị, cơ cấu truyền động – đóng/mở, chạy/dừng, v.v…;

• giá trị của các bộ đếm, bộ tạo số ngẫu nhiên;

• thông báo từ các dụng cụ đo lường;

• dữ liệu thứ cấp, thu được trên cơ sở của các thông số sơ cấp nói trên.

Như vậy Tag là đối tượng được xuyên suốt trong toàn bộ hệ thống điều khiển – từ các thanh ghi, ô nhớ trong PLC, đến các Tag ở trong các OPC Server, đến các Tag ở trong trung tâm xử lý tín hiệu, trên màn hình hiển thị các thông số kỹ thuật của hệ thống SCADA, v.v…

Dưới gốc độ của lập trình thì Tag được hiểu là một cấu trúc hoặc lớp (structure và class ở trong ngôn ngữ lập trình C/C++) với một tập hợp các đặc tính xác định. Từ việc tổng hợp các tính chất về Tag của nhiều hệ thống SCADA khác nhau và theo tiêu chuẩn OPC của Hiệp hội tự động hóa thế giới OPC Foundation (www.opcfoundation.org) thì chúng ta có thể rút ra được Tag phải có 3 đặc tính sau [5, 6]:

• value (v): giá trị của Tag, có thể là bất kỳ dạng dữ liệu nào – analog, digital, integer, float, string, v.v…;

• quality (q): đánh giá mức độ tin cậy và chất lượng của giá trị v. Quality có thể là GOOD, BAD, UNCERTIAN, v.v…;

• timestamp *(t)*: thời gian đi kèm với giá trị v, đánh giá thời điểm mà Tag có giá trị là v.

Tất cả các các thao tác (Read, Write (Modify)) tác động lên Tag đều phải là truy cập loại trừ (exclusive access), tức là giá trị thu nhận được v với chất lượng là q phải đi kèm với thời gian t tương ứng với thời điểm mà v và q thu nhận được.

Vậy Tag là đối tượng có những tính chất sau:

• Tag là một đối tượng thụ động (passive) mà có thể thay đổi trạng thái của mình khi có sự tác động của đối tượng khác;

• Tag là đối tượng dùng để lưu trữ, thu nhận và chuyển tải dữ liệu từ/cho các đối tượng khác.

• Tag là đối tượng có các function Read/Write được mô tả như sau:

Read (s, v, q, t) – hàm thu thập dữ liệu $\{v, q, t\}$ từ thiết bị nếu s là Device, hoặc từ bản thân của Tag nếu s là Cache.

Write(s, v, q, t) – hàm ghi tín hiệu điều khiển $\{v\}$ ra thiết bị nếu s là Device, hoặc vào bản thân của Tag nếu s là Cache và thu nhận kết quả của điều khiển $\{q, t\}$.

Để thực hiện các quá trình cập nhật dữ liệu vào/ra cho các Tag theo một chu kỳ thời gian thực nhất định, để đồng bộ hoá các truy cập loại trừ từ bên ngoài đến các Tag, để và điều khiển các đối tượng kỹ thuật thông qua các thuật toán đã cho, v.v… thì phải đặt ra một đối tượng, được gọi là đối tượng Dispatcher – đối tượng điều phối. Vậy:

• Dispatcher (D) là đối tượng chủ động (active) và tự bản thân có thread điều khiển để thực hiện công việc điều phối, thu thập dữ liệu và điều khiển của Component. Dispatcher có thể thay đổi trạng thái của mình không cần sự tác động ở bên ngoài;

• thông qua Dispatcher có thể tạo hoặc xoá bỏ các đối tượng khác (ví dụ như các Tag).

Để chuẩn hoá Component hay nói cách khác, để mô tả chức năng của Component cho các đối tượng khác (Client, Client có thể là một Dispatcher) biết thì đòi hỏi Component phải có những giao diện dành cho client. Những giao diện này bản chất của nó cũng là những đối tượng, và được gọi là interface. Interface là đối tượng có các tính chất sau:

• thực hiện các chức năng truy cập bằng Client của Component mà Interface nằm trong đó;

• là một đối tượng thụ động mà có thể được tạo ra và hủy diệt bởi đối tượng client khác.

Như vậy Component là một khối gồm có Dispatcher, các Interface và nhiều đối tượng Tag. Chúng ta lần lượt giới thiệu và phân tích có những interface chuẩn, tổng hợp bởi tác giả, ở trong một Component.

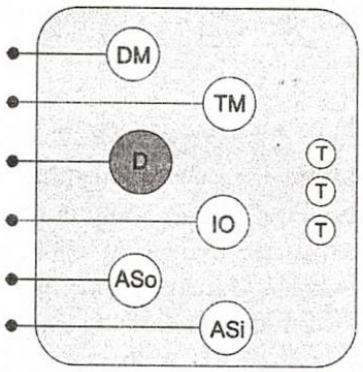
• Trước hết xin kể đến Interface IDispatcherStateMgt (DM) là interface điều khiển trạng thái Dispatcher của Component. Như vậy khi Client muốn sửa đổi cấu hình hay thay đổi thông số nào đó của Dispatcher thì phải thông qua đối tượng này.

• ITagMgt (TM) là Interface dùng để điều khiển các Tag của Component. Với ví dụ về function của Interface này có thể nói là function tạo/hủy diệt (Add/Delete) Tag.

• IDataIO (IO) dùng để trao đổi dữ liệu giữa đối tượng Client với Component. Các Function của Interface này phải kể đến Read (đọc) dữ liệu từ các Tag và Write (ghi) dữ liệu điều khiển vào các Tag. Các Function của IDataIO được chia ra thành 3 nhóm đó là SyncDataIO – đọc/ghi đồng bộ và AsyncDataIO – đọc/ghi không đồng bộ. Đối với phương pháp đọc/ghi đồng bộ – Client gửi cho Component yêu cầu gồm danh danh sách các Tag cần thu thập dữ liệu và chờ cho đến khi nó nhận được dữ liệu các Tag này từ Component. Trong trường hợp đọc/ghi không đồng bộ – Client gửi cho Component yêu cầu gồm danh sách các Tag cần thu thập dữ liệu và tiếp tục làm việc riêng của mình, khi Component hoàn thành yêu cầu này thì Component sẽ gữi (thông báo) kết quả lại cho Client thông qua Interface đăc biệt của Client và Interface này được gọi là IAdviseSink (ASi) – nguồn thu nhận thông báo.

• IAdviseSource (ASo) – hay là nguồn phát thông báo dùng để cho Client thông qua đó để đăng kí một danh sách các Tag mà Client muốn thu nhận dữ liệu một cách thường xuyên. Khi thông tin của một hoặc nhiều Tag trong danh sách này thay đổi thì Dispatcher sẽ thông báo lại cho Client thông qua IAdviseSink. Phương pháp truyền dữ liệu này gọi là Subscription.

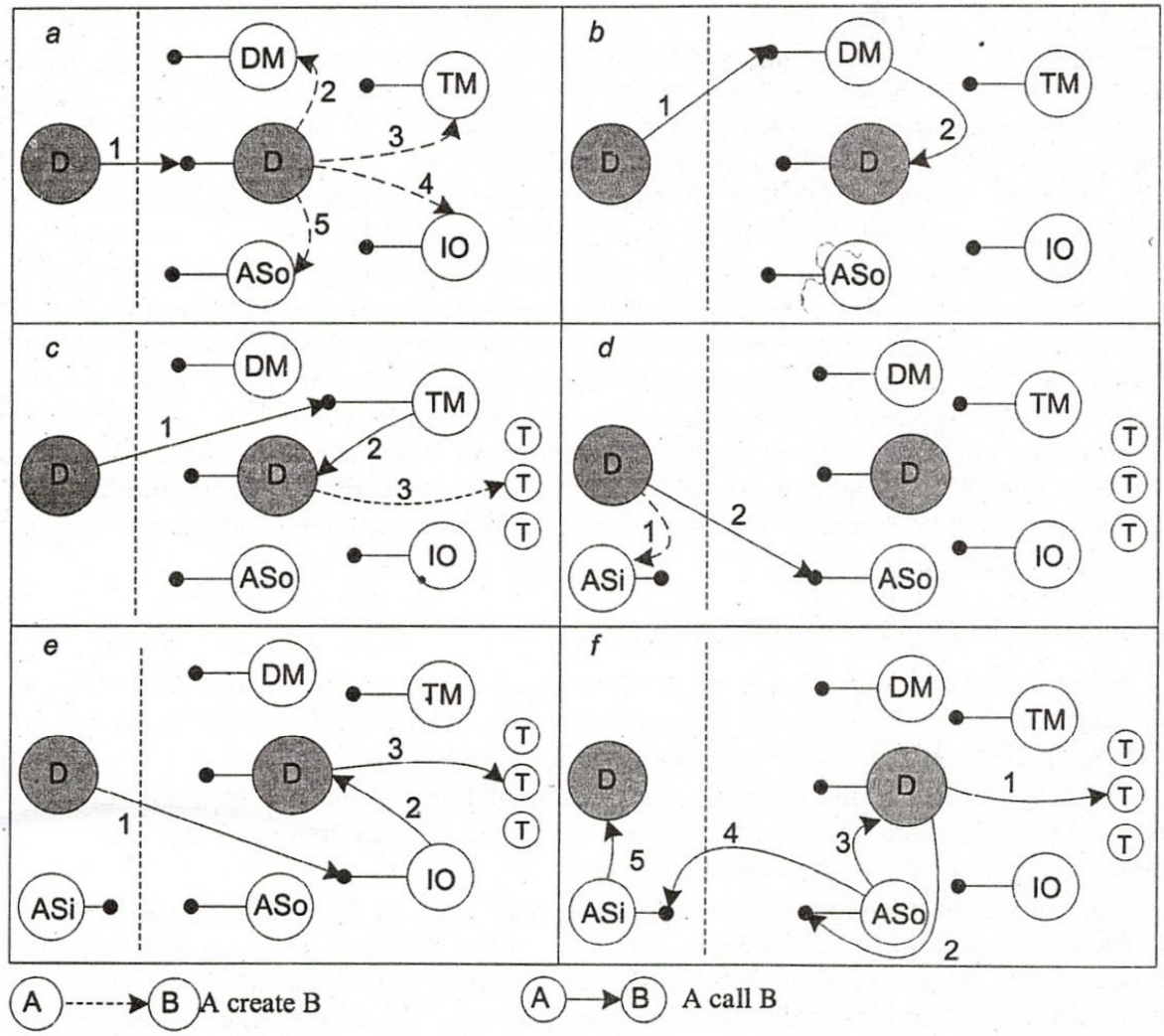
Như vậy mỗi Component trong hệ thống thu thập dữ liệu hoàn hão là một khối gồm có những thành phần cơ bản được mô tả như hình 2.



Hình 2. Component trong hệ thống DAQ&C

3. MÔ HÌNH HÀNH VI VÀ TƯƠNG TÁC GIỮA CÁC COMPONENT TRONG HỆ THỐNG

Bây giờ chúng ta xem xét quá trình tương tác giữa Component Client và Component Server trong hệ thống thu thập dữ liệu phân bố. Giả sử, ban đầu, trong hệ thống phía Client tồn tại Dispatcher của mình (gọi tắt là CD) và phía Server cung vậy (gọi tắt là SD). Dưới đây sẽ mô tả chuỗi làm việc của hệ thống.



Hình 3. Mô hình khái niệm

• Thông qua hệ thống (hệ điều hành, COM, ...) Client tìm thấy được SD. Thông qua SD (hình 3a, giai đoạn 1) Client có thể tạo các Interface: DM (giai đoạn 2), TM (giai đoạn 3), IO (giai đoạn 4) và ASo (giai đoạn 5) và nhận địa chỉ của các Interface này.

• Thông qua Interface DM Client có thể thay đổi trạng thái hay cấu hình của SD (hình 3b).

• Thông qua Interface TM Client có thể gọi các Function của Interface này (hình 3c, giai đoạn 1) và lần lượt các Function sẽ truy cập vào thư viện của SD (giai đoạn 2) để thực hiện công việc tạo các Tag và thay đổi cấu hình các Tag này.

• Client tạo Interface ASi của mình (hình 3d, giai đoạn 1) và đăng kí với Server địa chỉ ASi mà Client vừa tạo (giai đoạn 2). Như vậy, theo hình vẽ, địa chỉ của ASi phía Client được lưu trữ ở ASo phía Server chứ không phải là ở đối tượng D của Server.

• Đối với chế độ đọc/ghi đồng bộ thì thông qua Interface IO của Server, Client gọi những function Read/Write của Interface này (hình 3e, giai đoạn 1). Tiếp theo Read/Write của IO gọi những hàm trong thư viện của đối tượng D hoặc chuyển giao công việc đọc/ghi cho đối tượng D (giai đoạn 2). Tiếp theo, để thực hiện công việc của mình, đối tượng D gọi những hàm Read/Write của các Tag nằm trong danh sách yêu cầu của Client (giai đoạn 3). Đối với chế độ đọc/ghi không đồng bộ thì quá trình này chỉ xãy ra đến thao tác 2 (giai đoạn 2), có nghĩa là Client thông qua Interface IO của Server, Client đưa ra yêu cầu về việc đọc/ghi một danh sách Tag nào đó và Client tiếp tục thực hiện những công việc khác.

• Hình vẽ 3f mô tả việc thực hiện các yêu cầu đọc/ghi không đồng bộ và đăng ký (Subscription) của Client. Đối với chế độ đọc/ghi không đồng bộ thì D gọi hàm Read/Write của các Tag (giai đoạn 1) và thu nhận kết quả của các phép toán này. Tiếp theo D thông báo cho Interface ASo (giai đoạn 2) là phải thực hiện công việc chuyển giao kết quả cho Client. Để thực hiện công việc của mình ASo gọi những hàm thư viện của D (giai đoạn 3) để nhận kết quả đọc/ghi. Tiếp theo, ASo gọi Interface ASi (giai đoạn 4) để chuyển giao kết quả đọc/ghi, và sau đó, ASi gọi thư viện đối tượng D của Client để chuyển kết quả này để Client có thể hiển thị, xữ lý hay lưu trữ, ... Đối với chế độ Subscription thì D gọi hàm Read/Write của các Tag (giai đoạn 1) để kiểm tra sự thay đổi giá trị của các Tag, nếu giá trị của các Tag thay đổi thì D sẽ đóng gói những kết quả thay đổi này và tiếp tục thực hiện các bước tiếp theo như trong chế độ không đồng bộ.

4. CÁC ĐẶC TÍNH MỞ CỦA MÔ HÌNH

Mô hình đã được đưa ra ở trên, trước hết, mang đặc tính câu trúc của một phần mềm hiện đại - ứng dụng là một hệ thống bao gồm nhiều Component khác nhau nằm rãi rác trên các máy trong mạng. Như vậy, vấn đề khoảng cách đã được giải quyết.

Áp dụng mô hình có thể xây dựng một hệ thống tính toán song song hiệu quả. Ví dụ, đối tượng ASo có thể được bố trí ở trên một máy tính khác với máy tính đang chạy Component Server, và cũng có thể chạy trên máy tính Client.

Cấu trúc đã nêu ra thực hiện theo nguyên tắc phân cách các Component thuộc về thiết bị với các Component thuộc về Software, như vậy, khi thay đổi trong hệ thống thiết bị thu thập dữ liệu và điều khiển thì không cần phải sữa đổi các Component thuộc về Software và ngược lại khi nâng cấp phiên bản (ví dụ, thêm vào Component các Interface mới) của Component thì không ảnh hưởng gì đến hoạt động của thiết bị.

Vấn đề mở của mô hình được bộc lộ rỏ nét nhất là phần mềm được cấu thành từ nhiều Component khác nhau và các Component đều cung cấp các Interface để cho các thành phần khác có thể truy cập và điều khiển Component.

5.KẾT LUẬN

Mô hình khái niệm đã được đưa ra ở trong bài báo là một mô hình mang tính đa chức năng. Trên cơ sở mô hình này có thể xây dựng lên các Component của các phần mềm HMI, SCADA và Driver cho các thiết bị thu thập dữ liệu và điều khiển. Tức là từ cấp thấp nhất cho đến cấp cao nhất trong hệ thống điều khiển đa cấp hiện đại. Mỗi Component phải chứa trong mình một Dispatcher và chính nó điều khiển tất cả các truy cập từ bên ngoài. Các truy cập đều thông qua các Interface mở ra cho thế giới bên ngoài biết, thích nghi và hoà nhập. Có như vậy thì công việc tích hợp hệ thống của các chuyên gia trong lĩnh vực tự động hóa công nghiệp từ đây được dể dàng và nhẹ nhỏm hơn.

A CONCEPTUAL MODEL FOR DISTRIBUTED SYSTEMS OF DATA ACQUISITION AND CONTROL

Truong Dinh Chau, Nguyen Duc Thanh, Luong Van Lang
University of Technology, VNU-HCM

ABSTRACT: *The Supervisory Control and Data Acquisition (SCADA) is one of components of hierarchical control systems. This article focuses on offering a conceptual model for distributed systems of data acquisition and control on the basis of functional estimations, structural, advantageous and disadvantageous analyses of SCADA systems existing in industrial automation. The behavior and the interaction between components in the model are also described. Lastly, the open features of offered model in the article are listed.*

Keywords: *SCADA, OPC, component-oriented programming, open system*

TÀI LIỆU THAM KHẢO

[1]. G. Booch, *Object-Oriented Analysis and Design with Applications*, 2nd ed., Benjamin-Cummings, Redwood City, California, 604 p, (1993).

[2]. C. Szyperski, *Emerging Component software technologies – a strategic comparison*, Software – Concepts & Tools, №19, pp. 2-10, (1998).

[3]. A. Wang, K. Qian, *Component-oriented programming,* Wiley-Interscience, 334 p, (2005).

[4]. L. Wang, K. Tan, *Modern industrial autotomation software design,* Wiley-Interscience, 349 p, (2006).

[5]. F. Iwanittz and J. Lange, *OPC: fundamentals, implementation, and application,* [Softing], Heidelberg: Huthig, 221 p, (2002).

[6]. M. Santory, *OPC: OLE for Process Control,* Real-Time Magazine, №4, pp. 78-81, (1997).