

ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC BÁCH KHOA

PHAN DUY ANH

NGHIÊN CỨU VÀ XÂY DỰNG PHẦN MỀM SCADA

Chuyên ngành: Tự động hóa

LUẬN VĂN THẠC SĨ

TP. HỒ CHÍ MINH, tháng 12 năm 2010

**CÔNG TRÌNH ĐƯỢC HOÀN THÀNH TẠI
TRƯỜNG ĐẠI HỌC BÁCH KHOA
ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH**

Cán bộ hướng dẫn khoa học: Tiến Sĩ Trương Đình Châu

Cán bộ chấm nhận xét 1: Tiến Sĩ Nguyễn Đức Thành

Cán bộ chấm nhận xét 2: Tiến Sĩ Hoàng Minh Trí

Luận văn thạc sĩ được bảo vệ tại Trường Đại học Bách Khoa, ĐHQG Tp.HCM
ngày 04 tháng 01 năm 2011

Thành phần Hội đồng đánh giá luận văn thạc sĩ gồm:

1. Tiến Sĩ Nguyễn Đức Thành
2. Tiến Sĩ Huỳnh Thái Hoàng
3. Tiến Sĩ Hoàng Minh Trí
4. Tiến Sĩ Nguyễn Thiện Thành
5.

Xác nhận của Chủ tịch Hội đồng đánh giá LV và Bộ môn quản lý chuyên ngành sau
khi luận văn đã được sửa chữa (nếu có).

Chủ tịch Hội đồng đánh giá LV

Bộ môn quản lý chuyên ngành

Tp. HCM, ngày 04 tháng 01 năm 2011

NHIỆM VỤ LUẬN VĂN THẠC SĨ

Họ tên học viên: PHAN DUY ANH

Phái: Nam

Ngày, tháng, năm sinh: 20, 04, 1984

Nơi sinh: Quảng Ngãi

Chuyên ngành: Tự động hóa

MSHV: 01507309

I – TÊN ĐỀ TÀI:

NGHIÊN CỨU VÀ XÂY DỰNG PHẦN MỀM SCADA

II- NHIỆM VỤ VÀ NỘI DUNG:

- Tìm hiểu cấu trúc chung của các phần mềm SCADA
- Tìm hiểu kiến trúc phần mềm hướng dịch vụ (SOA)
- Tìm hiểu kỹ thuật lập trình hướng thành phần (COP)
- Xây dựng phần mềm SCADA trong môi trường Visual Studio.Net

III- NGÀY GIAO NHIỆM VỤ:

IV- NGÀY HOÀN THÀNH NHIỆM VỤ:

V- CÁN BỘ HƯỚNG DẪN: Tiến Sĩ Trương Đình Châu

CÁN BỘ HƯỚNG DẪN
(Họ tên và chữ ký)

CHỦ NHIỆM BỘ MÔN
QUẢN LÝ CHUYÊN NGÀNH
(Họ tên và chữ ký)

KHOA QL CHUYÊN NGÀNH
(Họ tên và chữ ký)

LỜI CẢM ƠN

Tác giả xin gửi lời cảm ơn chân thành đến:

Các Thầy, Cô giảng dạy Bộ môn Điều khiển Tự động – Khoa Điện
Điện tử - Trường Đại học Bách khoa Tp.HCM đã cho tác giả những tư
duy nền tảng về Tự động hóa và Điều khiển.

Thầy giáo, Tiến Sĩ Trương Đình Châu – Người đã cho tác giả kiến thức
về SCADA từ cái nhìn đầu tiên, đến nghiên cứu chuyên sâu về xây
dựng phần mềm SCADA hiện đại.

Ba, Mẹ, Vợ và em gái đã động viên, cỗ vũ tinh thần để tác giả có thể
hoàn thành tốt đề tài luận văn.

TÓM TẮT

Phần mềm điều khiển giám sát công nghiệp (SCADA) phù hợp với các hệ thống điều khiển phân tán (Distributed Control System) là yêu cầu và động lực thúc đẩy đối với các nhà phát triển hệ thống điều khiển giám sát bởi vì tính năng của các bộ phần mềm SCADA lớn hiện có trên thị trường khó có thể đáp ứng được các yêu cầu cao của các hệ thống điều khiển phân tán. Dựa trên thế mạnh của các công nghệ phần mềm mới: kiến trúc hướng dịch vụ (*Service Oriented Architecture*), lập trình hướng component (*Component Oriented Programming*), và môi trường phát triển tích hợp tốt của nền tảng .NET framework, luận văn đề xuất một phần mềm SCADA mới nhằm thỏa mãn nhu cầu của các hệ thống điều khiển phân tán. Công nghệ OPC (*OLE for Process Control*) cũng được sử dụng làm cơ sở truyền thông thời gian thực để tăng thêm tính mở cho hệ thống.

Từ khóa: *Service Oriented Architecture, Component Oriented Programming, Distributed Systems, Visual Studio, Real-time Systems, SCADA, System Architecture, SCADA System.*

MỤC LỤC

Chương	Nội dung	Trang
Chương 1	Giới thiệu	8
Chương 2	Tổng quan về hệ thống SCADA	10
	Mô hình lý thuyết của hệ thống SCADA	10
	Mô hình ứng dụng của hệ thống SCADA	12
Chương 3	Cấu trúc cơ bản của phần mềm SCADA	16
	Cấu trúc phần mềm SCADA WinCC	16
	Cấu trúc phần mềm SCADA InTouch	18
	Cấu trúc phần mềm SCADA GeniDAQ	21
	Các thành phần cơ bản của một phần mềm SCADA	23
	Giao thức truyền thông công nghiệp OPC	26
Chương 4	Lập trình hướng thành phần (COP) và kiến trúc phần mềm hướng dịch vụ (SOA)	33
	Định nghĩa và tính chất của Software Component	33
	Lập trình tạo ra Component	35
	Kiến trúc phần mềm hướng dịch vụ (Service Oriented	36

	Architechture)	
	Môi trường phát triển phần mềm tích hợp Visual Studio.NET (Visual Studio Environment – VSE)	40
	Các thành phần của Visual Studio.Net Environment	40
	Các tính năng của Visual Studio.Net Environment	43
Chương 5	Xây dựng phần mềm SCADA trong môi trường Visual Studio	46
	So sánh phần mềm Visual Studio.Net và phần mềm SCADA	46
	Phân tích và thiết kế phần mềm SCADA trong Visual Studio.Net	49
	Lập trình phần mềm SCADA trong Visual Studio.Net	60
Chương 6	Kết luận	73
	Danh mục công trình công bố của tác giả	74
	Tài liệu tham khảo	75

CHƯƠNG 1: GIỚI THIỆU

SCADA được định nghĩa là quy trình thu thập dữ liệu từ các thiết bị vật lý để giám sát, lưu trữ trên các máy tính và đưa các lệnh từ máy tính để điều khiển các thiết bị này. Hệ thống SCADA bao gồm các quy trình này chủ yếu dựa trên hệ thống máy tính (có thể trên một máy tính đơn lẻ hoặc một mạng máy tính) và phần mềm SCADA được cài đặt trên các máy tính này.

Theo nhu cầu về sản xuất hàng loạt, yêu cầu về năng suất sản xuất ngày càng cao, về chất lượng sản xuất và độ an toàn cao, các hệ thống sản xuất hiện tại có quy mô rất lớn và phức tạp. Các thiết bị của những hệ thống sản xuất này được lắp đặt trên các khu vực diện rộng và các trạng thái của chúng được giám sát và điều khiển từ rất nhiều phòng ban, bộ phận. Các hệ thống này được gọi là hệ thống phân tán (*Distributed System*). Hệ thống năng lượng điện được đề cập ở [1] là một ví dụ của hệ thống này. Chúng cần một phần mềm SCADA cao cấp với các tính năng đặc trưng: không lệ thuộc khoảng cách, vận hành mềm dẻo, nâng cấp dễ dàng và chi phí đầu tư hợp lý để điều khiển giám sát tất cả các thiết bị phân tán diện rộng.

Tuy nhiên, các gói phần mềm SCADA phổ thông trên thị trường tự động hóa hiện nay được phát triển dần lên từ các phiên bản cũ dựa trên các phân tích và thiết kế theo hướng cấu trúc hoặc hướng đối tượng, rất hạn chế trong

việc hỗ trợ tương tác qua mạng. Những điều này rất khó thỏa mãn những yêu cầu của hệ thống phân tán.

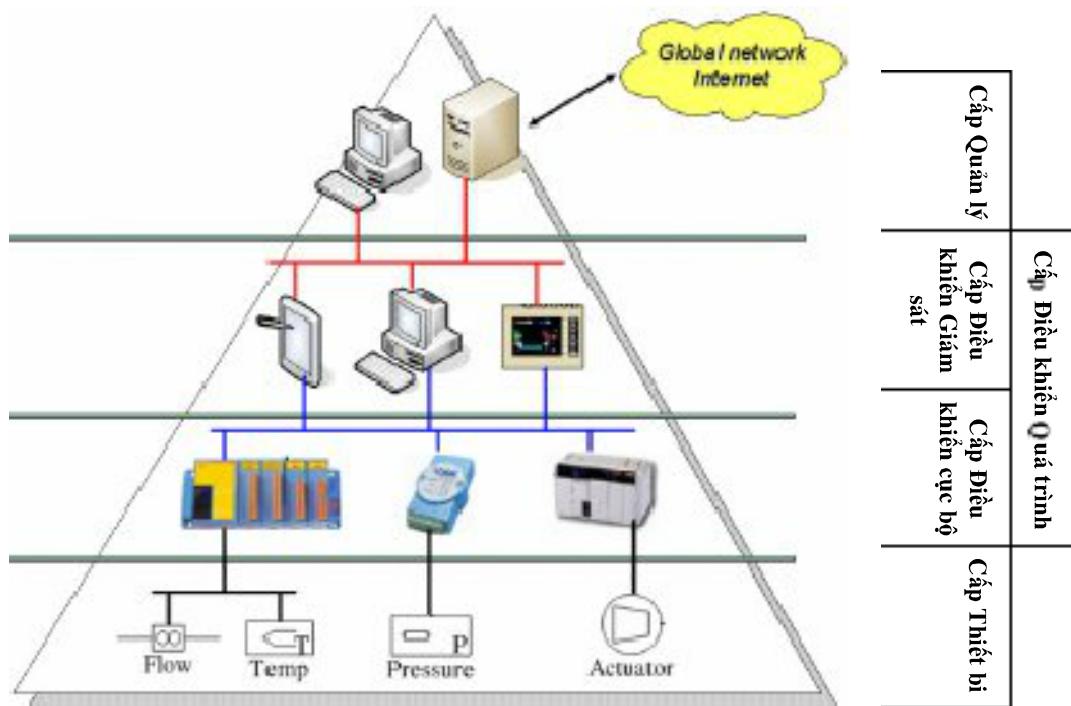
Với sự phát triển của *kỹ thuật lập trình hướng thành phần* (*Component Oriented Programming*), *kiến trúc phần mềm hướng dịch vụ* (*Service Oriented Architecture*), cùng với sự hỗ trợ tốt của nền tảng lập trình .NET framework, các phần mềm hiện tại và tương lai được phân tích và thiết kế theo hướng này để đạt được các mục đích: *dễ sử dụng, độ bảo mật cao, tính mở, tính tái sử dụng tốt, kiểm soát được độ phức tạp, và tương tác qua mạng dễ dàng*. Phần mềm SCADA hiện đại cũng nên theo xu hướng này để đạt được mục đích là giải pháp tốt cho các yêu cầu của hệ thống phân tán.

Đề tài này trình bày một phần mềm SCADA có kiến trúc hướng dịch vụ (*Service Oriented Architecture*) được lập trình theo hướng thành phần (*Component Oriented Programming*) nhờ được xây dựng trong môi trường Visual Studio.NET của Microsoft để làm nó phù hợp hơn với hệ thống điều khiển phân tán. Để có được phần mềm SCADA này, **chương 2** trình bày tổng quan cơ bản về hệ thống SCADA trong công nghiệp, **chương 3** sẽ nêu rõ kiến trúc của một phần mềm SCADA truyền thống, các đối tượng chính và trình bày về giao thức truyền thông công nghiệp OPC. **Chương 4** tổng quan kỹ thuật lập trình hướng thành phần (*COP*) và kiến trúc phần mềm hướng dịch vụ (*SOA*), đồng thời trình bày tính mở và tính đa năng của môi trường Visual Studio.NET. **Chương 5** tác giả đi xây dựng phần mềm SCADA trong môi trường Visual Studio.NET. **Chương 6** là kết luận và kiến nghị phát triển.

CHƯƠNG 2: TỔNG QUAN VỀ HỆ THỐNG SCADA

2.1. MÔ HÌNH LÝ THUYẾT CỦA HỆ THỐNG SCADA

Hệ thống SCADA định nghĩa chi tiết hơn là hệ thống thu thập dữ liệu thời gian thực từ các đối tượng để xử lý, biểu diễn, lưu trữ, phân tích và có khả năng điều khiển được những đối tượng này. Để có thể phản ứng linh hoạt với các đối tượng thiết bị, thì hệ thống SCADA cần là hệ thống thời gian thực (luồng dữ liệu vận hành trong hệ thống thay đổi liên tục) tạo nên một khối lượng lớn thông tin lưu trữ dư thừa trong cơ sở dữ liệu để phục vụ cho việc phân tích và báo cáo trong khoảng thời gian dài.



Hình 2.1 : Cấu trúc phân cấp cơ bản của hệ thống SCADA

Theo hình 2.1, hệ thống SCADA được phân chia thành 4 cấp:

Cấp thiết bị: bao gồm các đối tượng thiết bị được giám sát và điều khiển như các valves, động cơ, cảm biến nhiệt, cảm biến áp suất, ... Cấp thiết bị có nhiệm vụ chấp hành tín hiệu điều khiển thời gian thực từ cấp trên, trả lại thông số vận hành cho cấp trên.

Cấp điều khiển cục bộ: là các thiết bị có CPU như vi điều khiển, PLC dùng để xử lý tín hiệu thời gian thực, và đưa tín hiệu điều khiển cho cấp thiết bị, cấp này chủ yếu dùng để điều khiển thiết bị và đưa dữ liệu tươi cho cấp trên, nó có ít khả năng lưu trữ dữ liệu.

Cấp điều khiển giám sát: để giám sát và điều khiển đối tượng thiết bị từ màn hình hoặc máy tính, cấp điều khiển cục bộ đưa dữ liệu lên để phân tích, hiển thị và lệnh từ cấp này được đưa xuống cấp dưới để điều khiển thiết bị một cách tự động hoặc bằng tay (qua con trỏ chuột).

Cấp quản lý: sử dụng các số liệu đã được lưu trữ và phân tích để lập kế hoạch sản xuất, điều khiển điều hành để tối ưu hóa các chỉ số kinh tế kỹ thuật của công ty. Dữ liệu tại đây có thể chia sẻ để theo dõi và phân tích thông qua mạng toàn cầu internet.

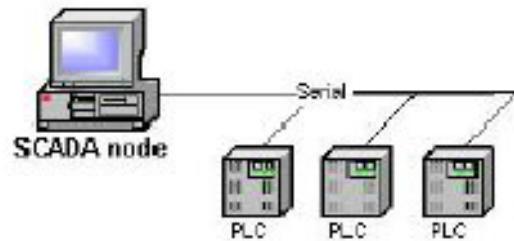
Để thực hiện việc giám sát và điều khiển các đối tượng thiết bị, trên các máy tính trong hệ thống SCADA cần thiết phải cài đặt các phần mềm SCADA/HMI. Các máy tính trong cấp điều khiển giám sát muốn giao tiếp được với các thiết bị điều khiển cục bộ thì cần có các Drivers mềm cài đặt bên trong gọi là IODrivers, các drivers này trước đây do các nhà sản xuất thiết bị cung cấp riêng rẽ kèm theo thiết bị điều khiển, hiện nay, các drivers đã được chuẩn hóa và tập hợp tạo thành một loại Driver Server có tên OPC (OLE for Process Control), OPC server được trình bày rõ hơn ở chương 3.

Từ khi có chuẩn truyền thông của OPC server, việc kết nối giữa máy tính và các thiết bị điều khiển cục bộ như PLC trở nên đơn giản và ‘trong suốt’ hơn.

2.2. CÁC MÔ HÌNH ỨNG DỤNG CỦA HỆ THỐNG SCADA

Hệ thống SCADA làm việc với dữ liệu lưu trữ lớn nên cần thiết phải có cơ sở dữ liệu tích hợp trong hệ thống. Đồng thời phần mềm SCADA có thêm các tính năng Alarm và Report để xuất hiện cảnh báo và báo cáo về các sự kiện của thiết bị dưới dạng trực quan.

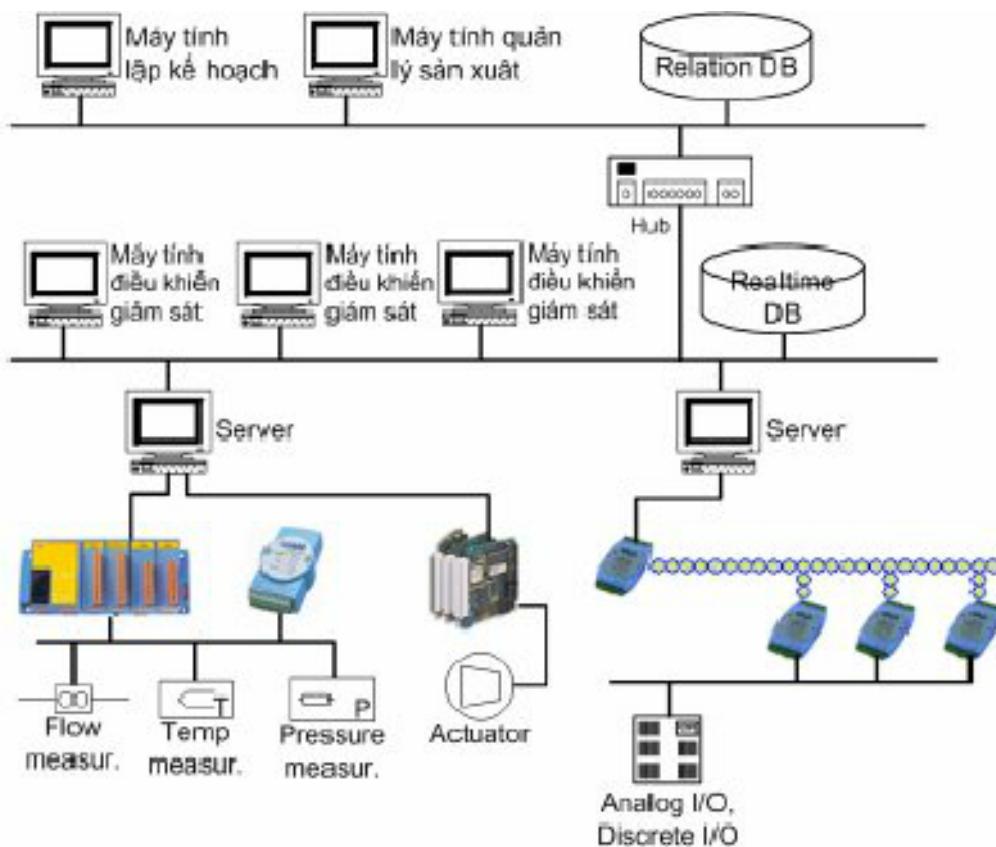
Mô hình đơn giản nhất bao gồm một hoặc nhiều PLC kết nối với một máy tính SCADA để vừa giám sát được, điều khiển được cũng như lập kế hoạch sản xuất ngay tại máy tính này. Mô hình được mô tả như hình 2.2. Mọi tính năng kể trên của phần mềm SCADA nằm trọn vẹn trong một máy tính SCADA.



Hình 2.2: Hệ thống SCADA đơn giản

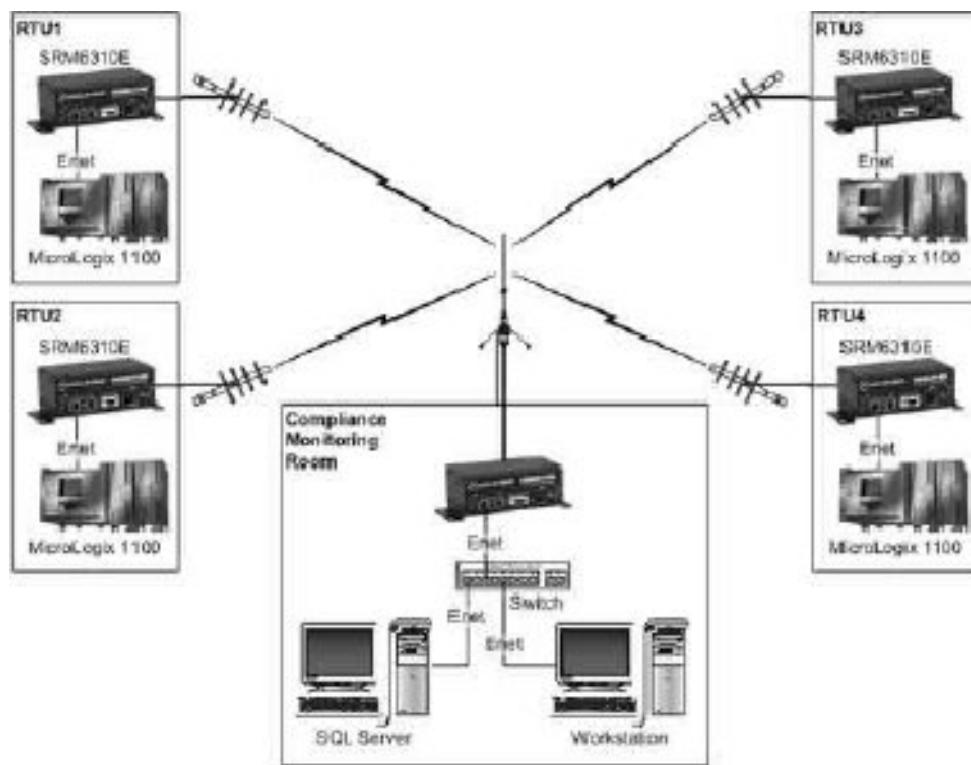
Hình 2.3 mô tả loại mô hình ứng dụng tầm cao hơn của hệ thống SCADA, trong đó có IODriver Server riêng, DataBase quan hệ (Relation DB) và cả Realtime DB. Hệ thống SCADA như mô hình 2.3 được áp dụng cho các hệ

thông sản xuất lớn, trong đó các IODriver Server chuyên dụng vào việc tạo giao diện vào ra cho luồng dữ liệu giữa cấp điều khiển giám sát (máy tính) và điều khiển cục bộ (PLC). Các máy tính điều khiển giám sát sẽ thông qua IODriver Server để điều khiển giám sát các đối tượng vật lý và tương tác (lưu trữ, đọc) dữ liệu thời gian thực với Realtime DB. Các máy tính cấp cao hơn (máy tính lập kế hoạch và quản lý sản xuất) sẽ dựa vào Realtime DB để thu thập số liệu và đưa ra quyết định. Các dữ liệu quan hệ phát sinh trên cấp máy tính này được lưu trữ vào Relation DB. Các dữ liệu trong Relation DB phục vụ cho hạ tầng công nghệ thông tin tại nhà máy cũng như các truy vấn từ xa.



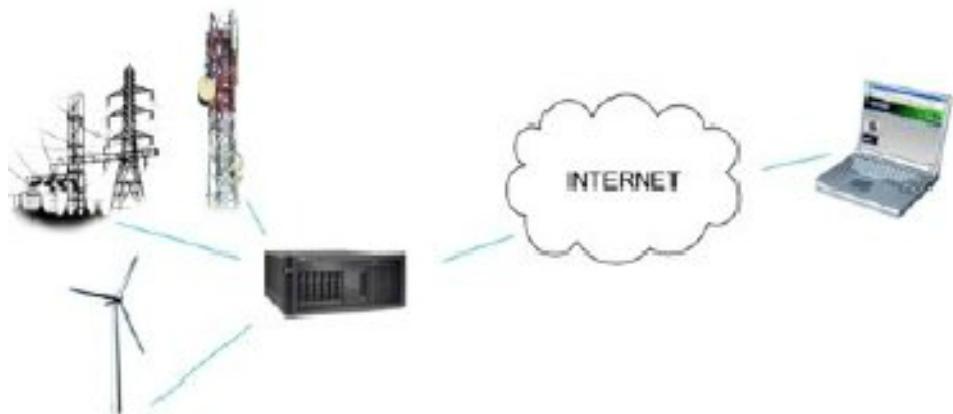
Hình 2.3: Mô hình hệ thống SCADA quy mô lớn

Cơ sở truyền thông của hệ thống SCADA mô tả trong hình 2.3 có thể là hệ thống có dây theo các giao thức CAN, PROFIBUS, ENTHERNET, ... Loại này được ứng dụng chủ yếu đối với các đối tượng nằm tập trung và trong một quy mô địa lý nhỏ. Cơ sở truyền thông của hệ thống SCADA áp dụng để điều khiển giám sát các đối tượng phân tán, nằm ở nhiều địa hình, vị trí địa lý cách xa nhau thường là hệ thống không dây như hình 2.4. Hệ thống không dây có thể vận hành dựa theo sóng radio, qua vệ tinh, ... Và yêu cầu về tính bảo mật đối với hệ thống SCADA này được chú ý đến kỹ lưỡng.



Hình 2.4: Hệ thống SCADA không dây

Để loại bỏ sự bất tiện về khoảng cách trong hệ thống phân tán, cũng như tận dụng nền tảng truyền thông tầm quốc tế của mạng internet, ngày nay hệ thống SCADA nền tảng web đang được phát triển và ứng dụng rộng rãi (hình 2.5). Tuy nhiên, vấn đề bảo mật cũng bị đặt ra gay gắt.



Hình 2.5: Mô hình Web based SCADA

Trong hệ thống SCADA nền tảng web, dữ liệu nhà máy từ các bộ điều khiển cục bộ (PLC) được tập trung về một Web Server. Máy server này vận hành ứng dụng SCADA nền tảng web (có đồ họa thời gian thực, vẽ đồ thị thời gian thực, báo cáo, cảnh báo) và cung cấp dịch vụ SCADA đến các máy clients truy cập đến nó.

CHƯƠNG 3: CẤU TRÚC CƠ BẢN CỦA PHẦN MỀM SCADA

3.1. CẤU TRÚC PHẦN MỀM SCADA WINCC

WinCC là phần mềm SCADA thuộc hãng Siemens (Đức), là phần mềm SCADA khá thông dụng tại Việt Nam. Cấu trúc hệ thống của WinCC được thiết kế theo hướng modular.

Hệ thống phần mềm SCADA WinCC bao gồm các hệ thống con:

- Hệ thống đồ họa (Graphics system)
- Ghi nhận cảnh báo (Alarm logging)
- Hệ thống lưu trữ (Archiving System)
- Hệ thống báo cáo (Report system)
- Truyền thông (Communication)
- Quản trị người dùng (user administration)

Hệ thống WinCC gồm có phần mềm cấu hình (Configuration Software) và phần mềm thực thi (Runtime software).

- Phần mềm cấu hình được sử dụng để tạo ra project

- Phần mềm thực thi dùng để thực thi project trong khi xử lý.

Hình 3.1 mô tả các thành phần và sự tương tác giữa các thành phần con trong hệ thống WinCC.

Theo hình 3.1, đầu tiên, dựa vào các trình biên tập trong phần mềm cấu hình, ta tạo ra project. Tất cả các trình biên tập đều lưu trữ thông tin của project trong cơ sở dữ liệu cấu hình (CS database).

Khi thực thi, thông tin của project được đọc ra từ CS database bởi phần mềm thực thi và project được thực thi. Dữ liệu hiện tại của project được lưu tạm tại cơ sở dữ liệu thực thi (RT database).

- Hệ thống đồ họa hiển thị graphic trên màn hình, đồng thời cũng nhận các thiết lập (input) từ người vận hành như khi người vận hành nhấp nút nhấn hay nhập giá trị.
- Việc giao tiếp giữa WinCC và hệ thống tự động được thực hiện bởi driver giao tiếp hay còn gọi là các kênh (channels). Các kênh này có nhiệm vụ thu thập các giá trị quá trình cần thiết cho các thành phần trong WinCC, đọc giá trị các tag từ hệ thống tự động và ghi các giá trị mới trở lại hệ thống tự động.
- Việc trao đổi dữ liệu giữa WinCC và các ứng dụng khác có thể được thực hiện bằng OPC, OLE hay ODBC.
- Hệ thống lưu trữ lưu giá trị của quá trình vào nơi lưu trữ giá trị quá trình. Các giá trị quá trình được lưu trữ được dùng để vẽ đồ thị (trend), đưa ra báo cáo (report)...
- Các giá trị của quá trình được theo dõi bởi tính năng ghi nhận cảnh báo (Alarm logging). Nếu một giá trị giới hạn bị tràn, Alarm logging sẽ tạo ra một

message để cảnh báo. Và hệ thống message cũng nhận các xác nhận (acknowledgements) từ người vận hành và quản lý các trạng thái của message. Alarm logging lưu trữ tất cả các message vào nơi lưu trữ message.

- Quá trình sẽ được báo cáo bởi hệ thống báo cáo (Report system) theo yêu cầu hoặc theo thời điểm định trước. Nơi lưu trữ giá trị quá trình và message được sử dụng cho mục đích này.

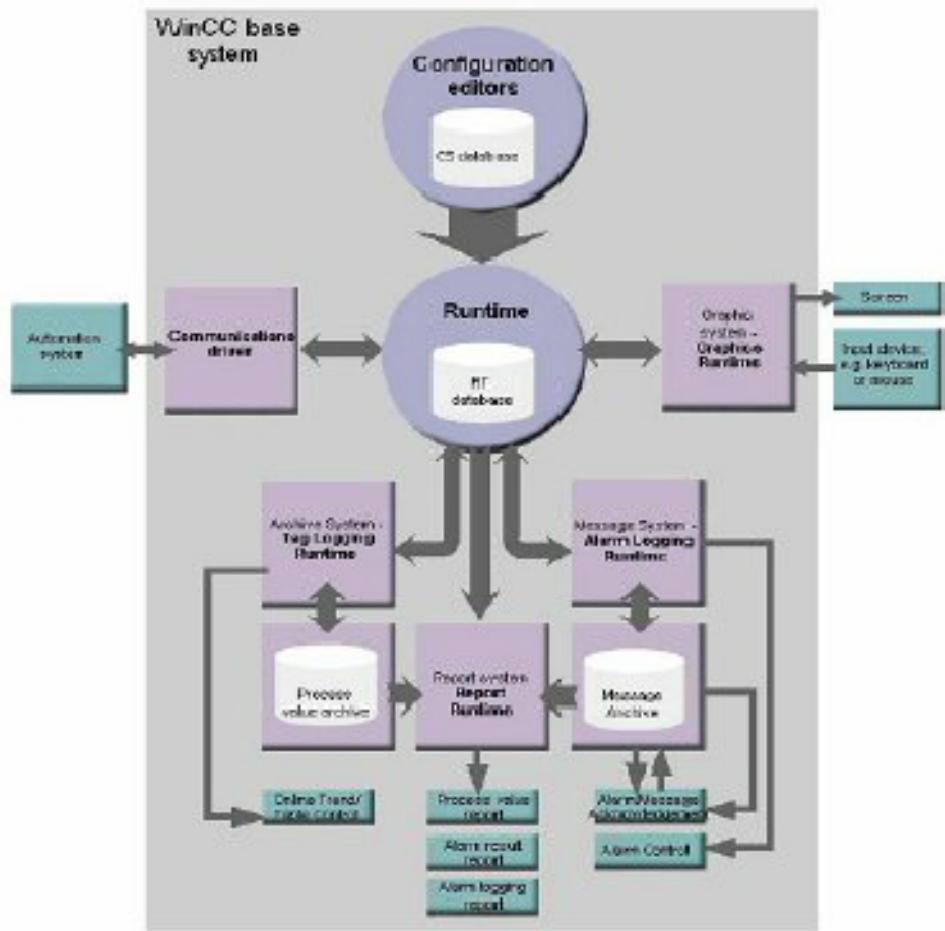
3.2 CẤU TRÚC PHẦN MỀM SCADA INTOUCH

Intouch là phần mềm SCADA của hãng Wonderware. Intouch có cấu trúc bên trong được mô tả như hình 3.2.

Hình 3.2 là mô hình các thành phần cơ bản của Intouch mà ta sử dụng để xây dựng và thực thi các ứng dụng SCADA.

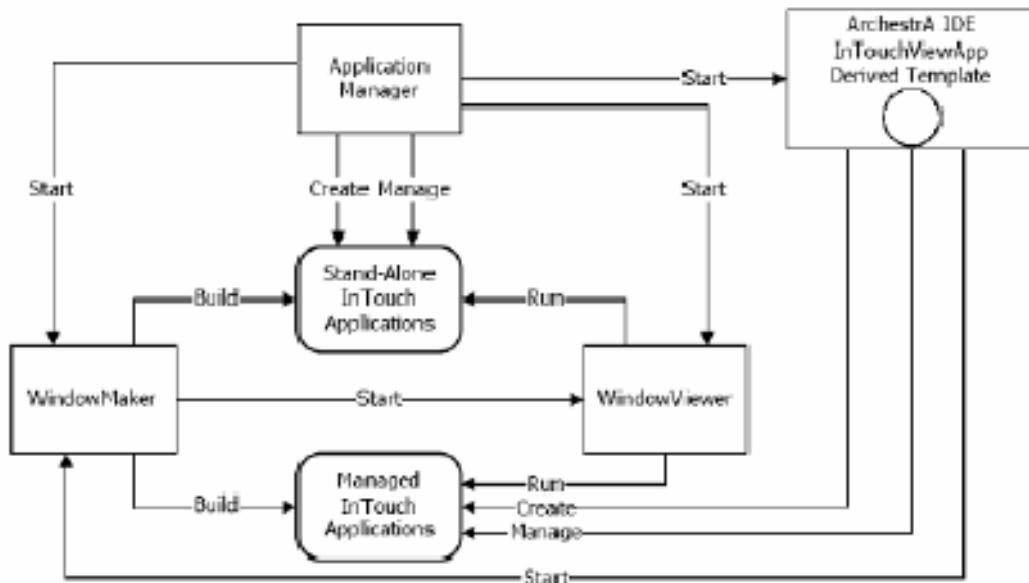
Thành phần Application Manager được dùng để tạo và quản lý các ứng dụng trên Intouch.

WindowMaker là môi trường phát triển ứng dụng bao gồm một tập các graphic và các công cụ phát triển để xây dựng các ứng dụng SCADA. Các công cụ phát triển này là ngôn ngữ script, các tính năng quản lý tag để tạo nên các đặc tính của các đối tượng trong các cửa sổ ứng dụng.



Hình 3.1: Các thành phần và sự tương tác giữa các thành phần bên trong WinCC

WindowViewer được sử dụng để chạy ứng dụng được tạo ra từ Windowmaker. Nó thực thi các đặc tính của các đối tượng trong các cửa sổ ứng dụng đã được tạo ra từ WindowMaker. Đồng thời nhờ các dữ liệu thu được từ các tags mà WindowViewer cung cấp dữ liệu cho các tính năng cảnh báo (alarm logging), xuất báo cáo (report generating), và vẽ đồ thị (trending).



Hình 3.2: cấu trúc phần mềm SCADA của Intouch

Các ứng dụng đơn lẻ: Stand-alone applications được tạo ra và quản lý bởi Application manager. Chúng được xây dựng toàn bộ trên Windowmaker và thực thi bởi Windowviewer, không có mối liên hệ nào đến Archestra IDE. Tuy nhiên chúng vẫn triển khai được trên tất cả các node trong cùng một mạng.

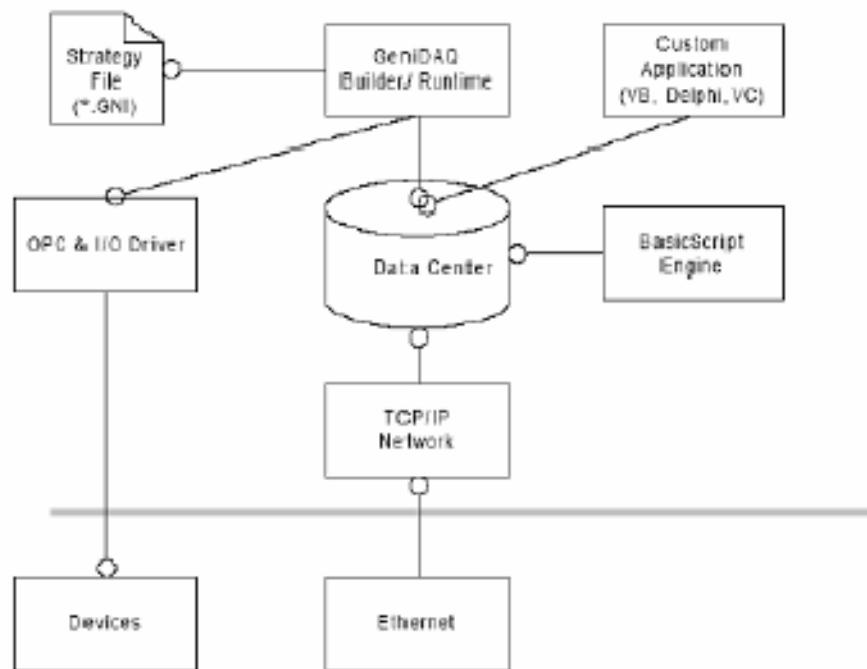
Archestra IDE khi được cài đặt cùng Intouch sẽ giúp tạo ra các graphic cao cấp, nhằm giúp phát triển các đối tượng giao diện cho ứng dụng.

Các ứng dụng được quản lý: managed applications được tạo ra, quản lý bởi Archestra IDE, sử dụng các symbols cao cấp của Archestra Symbol Editor.

3.3. CẤU TRÚC PHẦN MỀM SCADA GENIDAQ

GeniDAQ là phần mềm SCADA thuộc hãng Advantech (Đài Loan). Cấu trúc bên trong phần mềm được mô tả như sau:

GeniDAQ được thiết kế theo hướng module, kiến trúc tích hợp mở nên dễ tích hợp với các ứng dụng khác để chia sẻ dữ liệu thời gian thực. Khả năng làm việc và số lượng khối I/O mà GeniDAQ hỗ trợ được tăng lên đáng kể thông qua kiến trúc này. Kiến trúc này được mô tả như hình 3.3.



Hình 3. 3: cấu trúc phần mềm SCADA của GeniDAQ

GeniDAQ Builder là bao gồm 3 phần: Task Designer, Display designer and script designer. Được dùng để hiển thị các task, các trang thiết kế và script.

GeniDAQ Builder: là phần mềm phát triển ứng dụng cho phép người dùng tạo các ứng dụng HMI. Môi trường phát triển bao gồm cấu hình task, display, script.

GeniDAQ Builder cung cấp một giao diện đồ họa làm đơn giản hóa quá trình thiết kế đồ họa và thiết kế chương trình. Việc thiết kế chỉ cần chọn các khối biểu tượng từ toolbox, kết nối chúng lại với nhau, cấu hình thông số, và vẽ các màn hình hiển thị mà không cần phải lập trình.

GeniDAQ Runtime: cung cấp một môi trường thực thi thời gian thực cho ứng dụng của GeniDAQ. Không có một thay đổi nào trong ứng dụng này.

Basic Script engine: là một tập các DLL giúp thực hiện việc biên dịch mã nguồn ở build-time và thực thi script ở run-time. Và ngôn ngữ được dùng ở đây là VB for application. Ta có thể tính toán, đọc, viết files, DDE, và ODBC. Ta còn có thể giao tiếp với các ứng dụng khác như Microsoft Access, Microsoft Excel.

OPC client: dùng để kết nối với các thiết bị theo chuẩn OPC thông qua OPC server. Với chuẩn OPC, GeniDAQ dễ dàng tích hợp với các hệ thống.

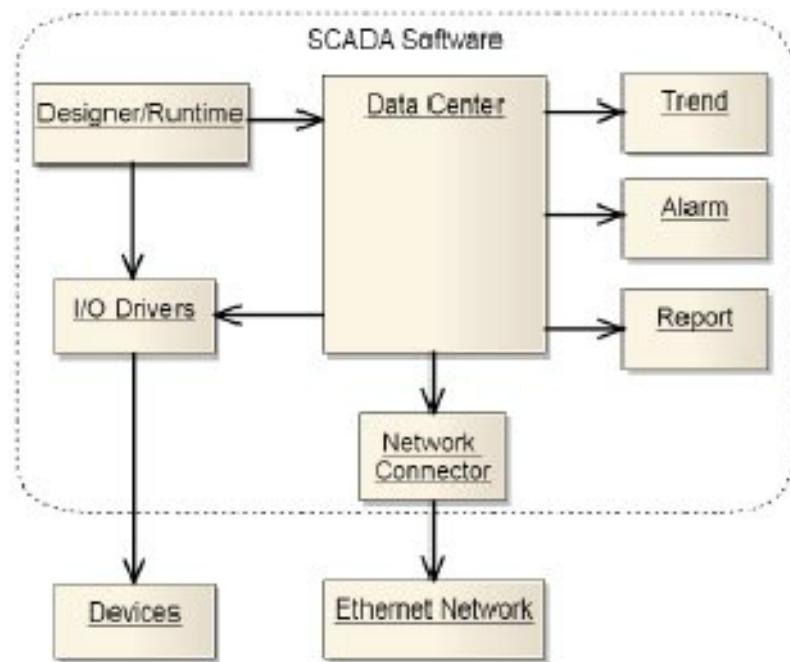
TCP/IP network: chức năng này được dùng để truyền thông giữa các máy tính cài GeniDAQ trong mạng. Module này cho phép một máy tính trong mạng hiển thị dữ liệu thu thập được bởi các máy tính khác trong mạng hay ngược lại thông qua giao thức TCP/IP.

Data Center: là nơi thu thập và kiểm soát dữ liệu. Là nơi kiểm soát toàn bộ dữ liệu thời gian thực và là nơi cung cấp hai tập giao diện: DDE và OLE Automation cho các ứng dụng khác truy cập hay gán dữ liệu cho GeniDAQ.

I/O Driver: thu thập dữ liệu thời gian thực từ phần cứng. GeniDAQ I/O driver bao hàm tất cả các phần cứng của Advantech, bao gồm các DA&C card, các bộ điều khiển MIC-200, các remote I/O module ADAM-4000 và các module phân tán ADAM 5000.

3.4. CÁC THÀNH PHẦN CƠ BẢN CỦA PHẦN MỀM SCADA

Sau khi đi vào phân tích một số phần mềm SCADA thông dụng, bùa tranh tổng quát về các đối tượng trong phần mềm SCADA và mối tương quan giữa chúng được thể hiện như **hình 3.4**.



Hình 3.4: mô hình phần mềm SCADA truyền thống

Phần mềm SCADA căn bản gồm có bốn đối tượng chính: *I/O driver*, *data center*, *designer*, *runtime* và bốn đối tượng con khác: *trend*, *alarm*, *report* và *network connector*.

I/O Driver lấy dữ liệu từ các thiết bị vật lý gửi đến các đối tượng khác trong phần mềm. Dữ liệu thời gian thực được đặc trưng bởi giá trị của các tags trong phần mềm SCADA.

Data Center lưu trữ dữ liệu thô hoặc giá trị đã được xử lý của các tags được chỉ định trong phần mềm SCADA.

Trend sử dụng dữ liệu trực tiếp từ đối tượng I/O driver hoặc dữ liệu lưu trữ trong cơ sở dữ liệu để vẽ đồ thị.

Alarm sử dụng dữ liệu từ I/O driver để cảnh báo các trạng thái vượt quá ngưỡng giá trị cho phép của các tags tới người dùng và đồng thời các trạng thái này cũng được lưu trữ vào cơ sở dữ liệu.

Report sử dụng dữ liệu từ cơ sở dữ liệu để in các báo cáo hay hiển thị danh sách dạng số lên màn hình.

Các tính năng của các đối tượng trend, alarm, report được cấu hình và hiển thị trong *Designer* và *Runtime* bởi các trend-viewer, alarm-viewer, report-viewer theo trình tự.

Network Connector, đối tượng này cho phép các truy cập từ bên ngoài hệ thống SCADA hoặc từ các ứng dụng của third-party vào cơ sở dữ liệu để thu thập dữ liệu hoặc sử dụng tài nguyên của data center.

Designer là giao diện người dùng, bao gồm nhiều công cụ thiết kế như các labels, textboxes, alarm-viewer, report-viewer, trend-viewer, etc. để hiển thị các giá trị của các tags. Buttons, switches, etc. để thay đổi giá trị của tags. Nó có thể chỉ định tags nào sẽ ghi dữ liệu vào cơ sở dữ liệu hay được sử dụng trong các đối tượng trend, alarm, hoặc report. Designer được sử dụng bởi các kỹ sư hệ thống để thiết kế các mô hình nhà máy để giám sát và điều khiển các

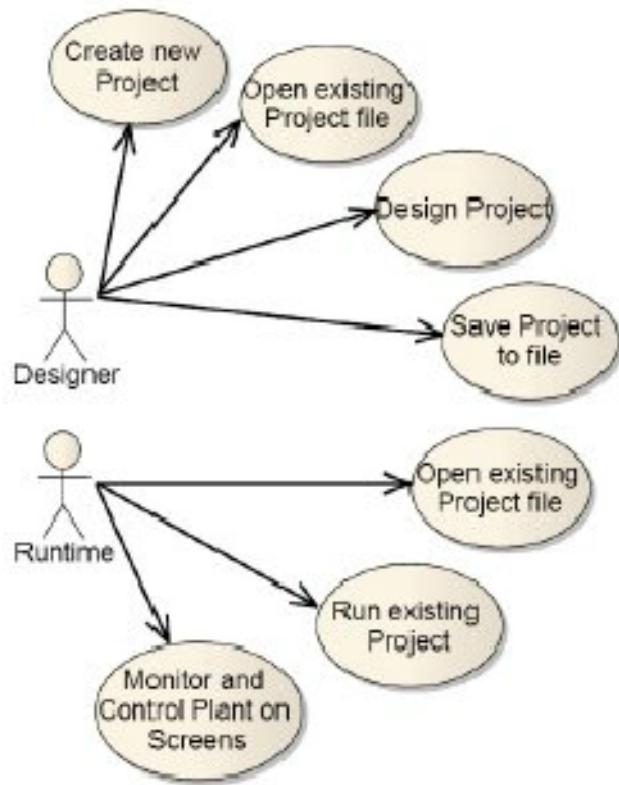
quá trình trong nhà máy. Sau khi thiết kế trong Designer, các kỹ sư hệ thống phải lưu lại công việc của họ vào file dự án. File này sẽ được đọc và thực thi bởi đối tượng Runtime.

Runtime là đối tượng thực thi file dự án được tạo ra bởi Designer. Nó được dùng để giám sát và điều khiển các quá trình trong nhà máy. Các tính năng của Designer và Runtime được thể hiện trong hình 3.5.

Trong gói phần mềm GeniDAQ (Advantech, Đài Loan), GeniDAQ builder và GeniDAQ runtime tương ứng là designer và runtime. Trong Intouch (Wonderware, USA), Window maker là designer và Window viewer là runtime.

Sau khi sử dụng các gói phần mềm SCADA thông dụng trên thị trường như WinCC (Siemens, Đức), Intouch, GeniDAQ vào nhiều dự án nhà máy, tác giả nhận ra rằng chúng tích hợp gần như toàn bộ công nghệ mới, nhưng chúng lại rườm rà gò ghè vì dựa trên phương pháp lập trình modular [5-7]- một kỹ thuật lập trình cũ.

Chúng được phát triển lên từ phiên bản đầu tiên, phiên bản này đã được phân tích và thiết kế dựa trên các công nghệ phần mềm cũ và rất nhiều bản vá cập nhật. Do vậy, chúng ngày càng lớn dần lớn dần và trở thành một khối phức tạp. Việc tương tác qua mạng để giám sát điều khiển phức tạp và bị hạn chế. Do vậy yêu cầu về một phần mềm SCADA hiện đại, loại bỏ hết các điểm nhược trên là cần thiết.



Hình 3.5: tính năng của Designer và Runtime

3.5. GIAO THÚC TRUYỀN THÔNG CÔNG NGHIỆP OPC

3.5.1. TỔNG QUAN VỀ OPC

OPC (*OLE for Process Control*) được hội định chuẩn OPC (OPC Foundation) định nghĩa là “*kết nối mở* (*Open Connectivity*) thông qua các *tiêu chuẩn mở* (*Open Standards*), thỏa mãn được yêu cầu trong lĩnh vực tự động hóa giống như các drivers máy in được tích hợp trong Windows”. OPC là chuẩn để các máy tính kết nối và truyền thông dữ liệu với các thiết bị điều khiển lập trình (PLC).

OPC là kiểu kết nối mở trong lĩnh vực tự động hóa công nghiệp. Tính tương tác được đảm bảo thông qua việc hình thành và phát triển của các tiêu chuẩn mở. Ở thời điểm hiện tại có bảy tiêu chuẩn đã hoàn thiện cũng như đang phát triển.

Dựa trên nền tảng các tiêu chuẩn cơ bản và công nghệ hiện có trên thị trường kỹ thuật, tổ chức OPC Foundation tạo ra các tiêu chuẩn thỏa mãn các yêu cầu của ngành công nghiệp. OPC sẽ tiếp tục tạo ra các chuẩn mới cho các yêu cầu ngày càng tăng và đáp ứng các yêu cầu hiện có đồng thời tạo nên các công nghệ mới.

OPC là một loạt các tiêu chuẩn mở. Tiêu chuẩn đầu tiên (thuộc tính cơ bản: thuộc tính truy cập dữ liệu – *Data Access Specification*) là kết quả của việc kết hợp nhiều nhà cung cấp thiết bị tự động hóa hàng đầu thế giới với Microsoft thành tổ chức để tạo ra một chuẩn kết nối giữa các thiết bị tự động hóa và máy tính. Dựa trên công nghệ OLE COM (Component Object Model) và DCOM (Distributed Component Object Model), một tập các chuẩn bao gồm các đối tượng, giao diện và phương thức sử dụng trong điều khiển quá trình và các ứng dụng tự động hóa sản xuất được hình thành để tạo ra các thuộc tính tương tác được (interoperability). Công nghệ COM/ DCOM cung cấp nền tảng (framework) để các sản phẩm phần mềm phát triển. Hiện tại có hàng trăm OPC Data Access servers và clients có mặt trên thị trường.

Yêu cầu của việc xuất hiện OPC cũng giống như drivers máy in trong DOS và sau đó là trong Windows. Dưới nền DOS, các nhà phát triển của mỗi một ứng dụng đều phải viết driver cho mỗi máy in. AutoCAD phải viết ứng dụng AutoCAD và các drivers máy in. WordPerfect phải viết ứng dụng WordPerfect và drivers cho các máy in. Họ phải viết một driver riêng biệt cho mỗi một máy in khi muốn hỗ trợ: một driver cho Epson FX-80, và một cho

HP LaserJet và cứ như thế cho các loại máy in khác. Trong ngành tự động hóa công nghiệp, Intellution viết phần mềm HMI của họ, và các drivers tương ứng cho mỗi một thiết bị công nghiệp (bao gồm tất cả các dòng PLC). Rockwell viết HMI của họ, và các drivers tương ứng cho các thiết bị công nghiệp (các dòng PLC, không chỉ riêng của họ).

Windows đã giải quyết vấn đề driver máy in bằng cách tích hợp hỗ trợ driver máy in vào hệ điều hành. Và giờ đây, một gói driver máy in hỗ trợ tất cả các ứng dụng phần mềm in ấn. Gói các drivers máy in này do các nhà sản xuất máy in (phân cứng) viết chứ không phải do các nhà phát triển ứng dụng phần mềm viết. Windows cũng cung cấp cấu trúc mở cho các drivers của thiết bị công nghiệp cũng giống như driver máy in.Thêm các thuộc tính OPC vào Microsoft's OLE Technology trong Windows cho phép tạo ra tiêu chuẩn OPC. Giờ đây, các nhà sản xuất thiết bị công nghiệp có thể viết ra OPC DA Servers và phần mềm HMI trở thành OPC Clients.

Từ đây, các nhà phát triển phần mềm công nghiệp không tốn nhiều thời gian vào việc nghiên cứu, thực hiện kết nối truyền thông mà dành nhiều thời gian vào phát triển các đặc tính đặc trưng của phần mềm. Đối với người dùng, lợi ích họ trở nên được mở rộng hơn. Người dùng có thể chọn nhà cung cấp phần mềm dựa trên các đặc tính tính năng cần thiết thay vì quan tâm đến việc họ hỗ trợ driver cho thiết bị người dùng đang có hay không. Người dùng không cần phải bỏ ra một khoản phí lớn để cập nhật lại driver khi họ thay đổi thiết bị. Người dùng cũng được bảo đảm về chất lượng truyền thông tốt nhờ vào đặc tính tiêu chuẩn của OPC DA. Các sản phẩm OPC được tạo ra một lần và tái sản xuất nhiều lần, do vậy, chúng trải qua việc kiểm soát chất lượng và cải tiến liên tục.

Đặc tính cơ bản nhất là Data Access đã được chuẩn hóa việc thu thập dữ liệu. Các kiểu truyền thông dữ liệu khác cũng sẽ có lợi hơn khi được chuẩn hóa. Các chuẩn của Alarms & Events, Historical Data, và Batch Data cũng đã được thiết kế, đưa vào sử dụng.

3.5.2. CÁC LOẠI OPC HIỆN TẠI

OPC Data Access: là loại cơ bản nhất. Được dùng để truyền dữ liệu từ PLCs, DCSs và các thiết bị điều khiển khác qua HMIs và các clients hiển thị khác. Hiện tại phiên bản Data Access 3 đã ra đời, nó được cải tiến khả năng hiển thị và tích hợp XML-DA Schema.

OPC Alarm & Events: cung cấp các cảnh báo và sự kiện theo yêu cầu (khác với dòng dữ liệu liên tục của Data Access). Loại này bao gồm các cảnh báo quá trình, các hành động của người vận hành, các thông điệp mang thông tin, các thông điệp tracking/auditing.

OPC Batch: loại này mang quan điểm OPC vào các yêu cầu của các quá trình vận hành mẻ (batch processes). Nó cung cấp giao diện cho việc thay đổi thông số vận hành thiết bị (dựa theo model s88.01) và các điều kiện vận hành hiện tại.

OPC Data Exchange: loại này cung cấp cơ sở truyền thông server-to-server dựa trên mạng ethernet fieldbus, tạo ra sự tương tác giữa các nhà sản xuất và thêm vào các dịch vụ remote configuration, diagnostic và monitoring/management.

OPC Historical Data Access: trong khi **OPC Data Access** cung cấp các truy cập đến đối tượng là dữ liệu thời gian thực, thay đổi liên tục thì **OPC Historical Data Access** cung cấp các truy cập đến data đã được lưu trữ. Từ

một hệ thống ghi dữ liệu nối tiếp đến hệ thống SCADA phức tạp, các lưu trữ lược sử có thể được thu thập theo một phương thức chuẩn.

OPC Security: cung cấp cách thức điều khiển việc truy cập dữ liệu đến các servers để bảo vệ các thông tin sản xuất cần bảo mật và bảo vệ servers khỏi các thiết lập chưa được cấp phép.

OPC XML-DA: cung cấp các luật mềm dẻo, nhất quán để thu thập dữ liệu từ nhà máy sử dụng XML, kế thừa công nghệ từ Microsoft về SOAP và Web Services.

OPC Complex Data: là sự kết hợp giữa OPC Data Access với XML-DA để cho phép khai thác và mô tả nhiều loại dữ liệu phức tạp như các cấu trúc binary và các XML documents.

OPC Commands: một nhóm làm việc đã được thành lập để phát triển một tập các giao diện cho phép các OPC Clients và Servers xác nhận, gửi và giám sát các lệnh trong khi thực thi trên thiết bị.

Trong đề tài này, giao thức OPC Data Access được tác giả sử dụng để xây dựng I/O Driver cho phần mềm SCADA mới, giúp phần mềm SCADA có được một chuẩn truyền thông mở, uyển chuyển với các thiết bị điều khiển công nghiệp và hỗ trợ hầu hết các thiết bị điều khiển công nghiệp hiện đại.

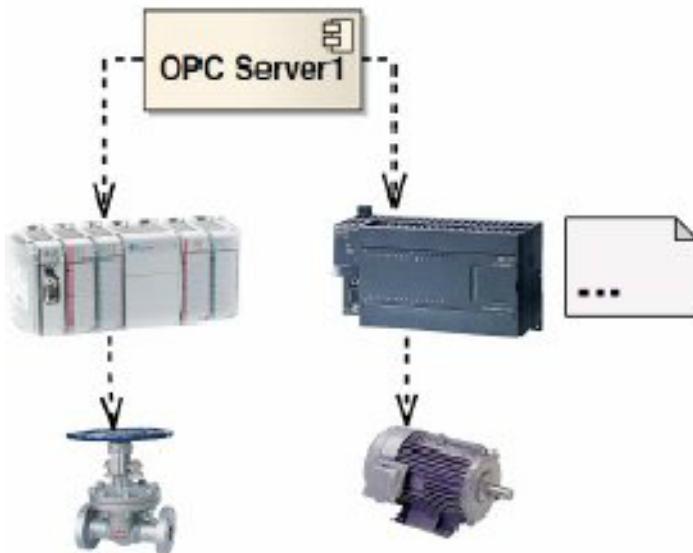
3.5.3. CHI TIẾT VỀ OPC DATA ACCESS

Chuẩn OPC Data Access bao gồm một OPC Server Component hoạt động theo chuẩn truyền thông OPC và OPC Client Component có thể kết nối tới OPC Server Component để thu thập và giám sát dòng dữ liệu thời gian thực. Mỗi thành viên của OPC Foundation có thể sở hữu một OPC Server riêng của mình sau khi tích hợp nhiều loại drivers riêng vào OPC Server Component do

OPC Foundation tạo ra. Có một số nhà sản xuất không chế tạo PLC nhưng họ lại là nhà tích hợp OPC, nên OPC Server của họ có hầu hết các driver của các họ PLC trên thị trường. Điển hình cho nhà sản xuất này là Kepware Inc.

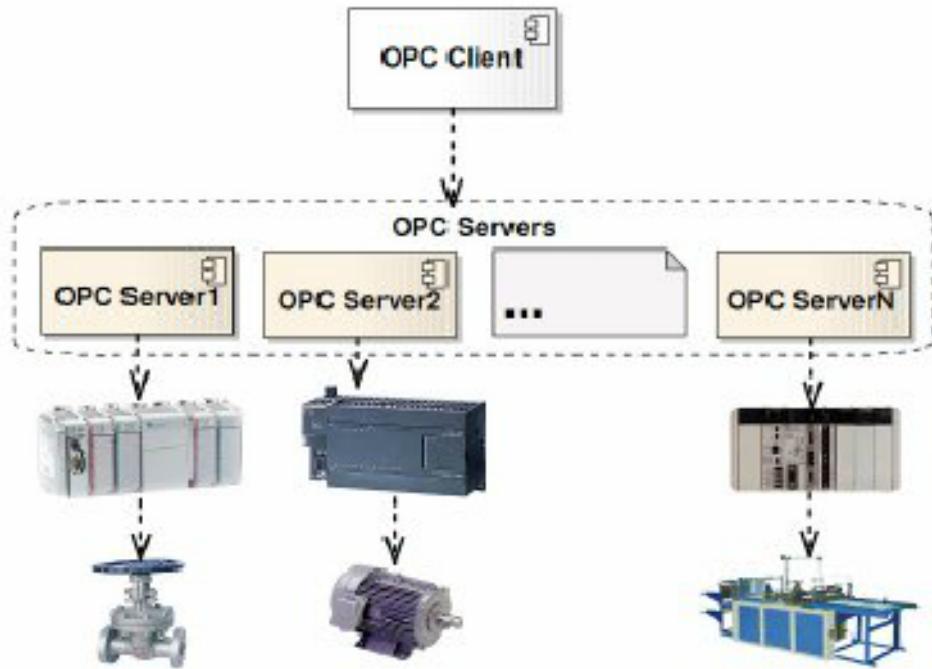
OPC Client Component được tích hợp vào các ứng dụng phần mềm để thu thập dữ liệu và giám sát các thiết bị công nghiệp kết nối với OPC Server mà nó kết nối.

Hình 3.6 Mô tả kết nối giữa OPC Server và thiết bị điều khiển lập trình. Một OPC Server một lúc có thể kết nối và làm việc với nhiều PLC của nhiều hãng khác nhau, mở rộng được tầm điều khiển và giám sát của hệ thống SCADA so với các driver cổ điển trước đó (chỉ có thể kết nối một máy tính với một PLC).



Hình 3.6: Một OPC Server có thể làm việc với nhiều PLC

Hình 3.7 mô tả tương tác giữa OPC Client và OPC Server. Do được lập trình theo hướng Component nên việc tương tác giữa OPC Client và OPC Server



Hình 3.7: Một OPC Client có thể làm việc với nhiều OPC Server

rất mềm dẻo, một OPC Client có thể kết nối được với một hoặc nhiều OPC Server. Điều này càng mở rộng hơn nữa tầm kiểm soát đối tượng thiết bị của chuẩn truyền thông OPC, làm cho hệ thống SCADA càng phù hợp hơn với hệ thống công nghiệp phân tán.

CHƯƠNG 4: LẬP TRÌNH HƯỚNG THÀNH PHẦN (COP) VÀ KIẾN TRÚC PHẦN MỀM HƯỚNG DỊCH VỤ (SOA)

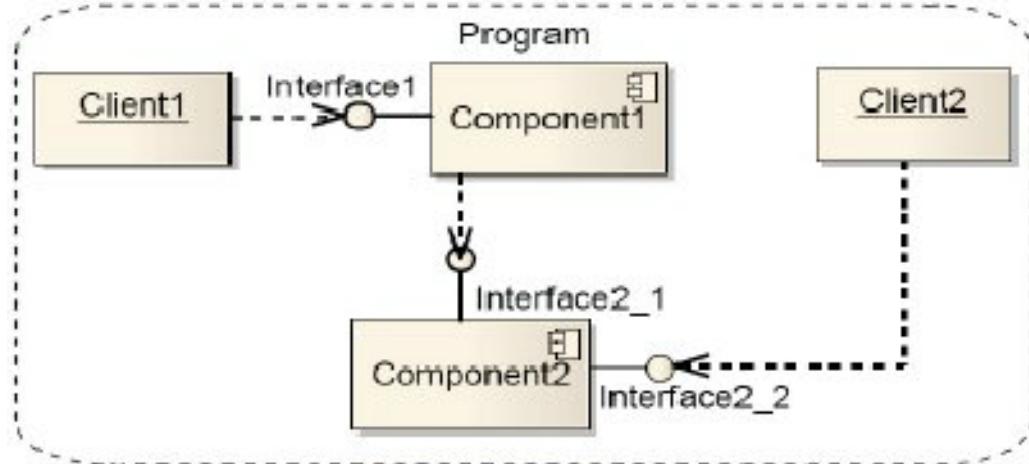
4.1. ĐỊNH NGHĨA VÀ TÍNH CHẤT CỦA SOFTWARE COMPONENT

Lập trình hướng thành phần (Component Oriented Programming) là một kỹ thuật lập trình hiện đại cho phép các chương trình được tạo nên bởi các thành phần phần mềm (Software Component). Software Component là một mã code máy tính có khả năng *tự chứa đựng* (*seft-contained*), *tự vận hành* (*seft-deployed*) với các tính chất được định nghĩa tốt và có thể được kết nối với các thành phần phần mềm khác thông qua các giao diện (Interfaces)[4].

Hình 4.1. cho thấy một chương trình được xây dựng bởi các thành phần. Tính năng của chương trình được cung cấp bởi *client1* và *client2*. Client1 kết nối với *component1* thông qua *interface1* để lấy dịch vụ. Client2 kết nối với *component2* thông qua giao diện *interface2_2*. *component2* cũng cung cấp dịch vụ cho *component1* thông qua *interface2_1*.

Bởi vì một component là một mã máy tính có tính tự chứa “seft-contained”, nó chỉ liên hệ với bên ngoài thông qua các giao diện interfaces. Vì vậy, khi sử dụng một thành phần phần mềm, chúng ta không cần biết bên trong nó vận hành thế nào và sự thật, ta không thể biết điều này nếu thành phần được viết

bởi một hãng thứ ba. Điều này đặc trưng cho việc dễ sử dụng và độ an toàn cao về bản quyền của thành phần phần mềm.



Hình 4.1: một chương trình hướng component

Nếu khách hàng sử dụng phần mềm hướng thành phần, họ có thể tự cập nhật hay nâng cấp phần mềm bằng cách thay thế các thành phần hiện có bằng các thành phần mới có cùng giao diện interfaces. Điều này không làm ảnh hưởng hoạt động của các component khác. Với cách này, ta cũng có thể thao tác cho sản phẩm phần mềm của chính mình. Điều này đặc trưng cho tính mở và tính tái sử dụng của các chương trình được xây dựng từ components.

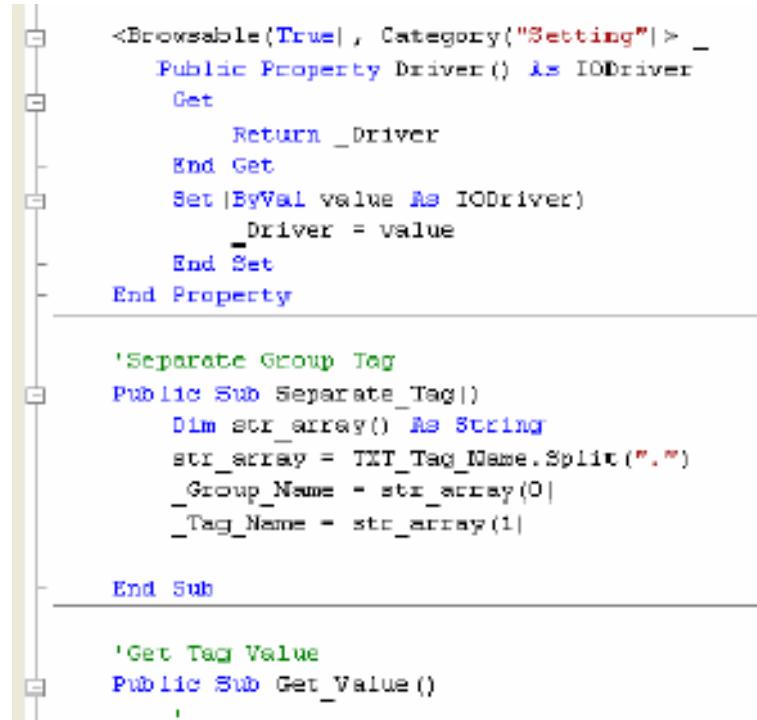
Một component phần mềm còn là một mã máy tính có khả năng tự vận hành. Nó có thể được cài đặt và thực thi một cách độc lập với các components khác. Vì vậy, khi cung cấp dịch vụ cho các đối tượng khác, nó đóng vai trò là server, và các đối tượng sử dụng dịch vụ của nó được gọi là các clients. Trong hình 4.1, component1 là server của client1 và là client của component2. Component2 là server của client2 và component1. Với sự hỗ trợ tốt của nền tảng .NET, components có thể vận hành như những servers trên nhiều máy tính khác nhau, chúng có thể tương tác qua lại với nhau từ xa [2]. Chương

trình bao gồm các components chạy trên nhiều máy tính khác nhau được gọi là ứng dụng phân tán. Loại chương trình như thế này dùng làm giải pháp cho các hệ thống phân tán phức tạp. Điều này còn tượng trưng cho tính mở của phần mềm hướng component.

Tất cả các thuộc tính: *dễ sử dụng, độ bảo mật cao, tính mở, tính tái sử dụng, kiểm soát được độ phức tạp* làm cho phần mềm hướng thành phần tốt hơn những công nghệ phần mềm trước đó (structured programming, object-oriented programming).

4.2. LẬP TRÌNH TẠO COMPONENT

Hiện tại nhiều ngôn ngữ và môi trường lập trình phần mềm hỗ trợ lập trình hướng component. Trong số này, các ngôn ngữ .NET trong môi trường Visual Studio (VSE) của Microsoft hỗ trợ lập trình hướng Component một cách trọn vẹn.



```
<Browsable(True), Category("Setting")> _
    Public Property Driver() As IODriver
        Get
            Return _Driver
        End Get
        Set(ByVal value As IODriver)
            _Driver = value
        End Set
    End Property

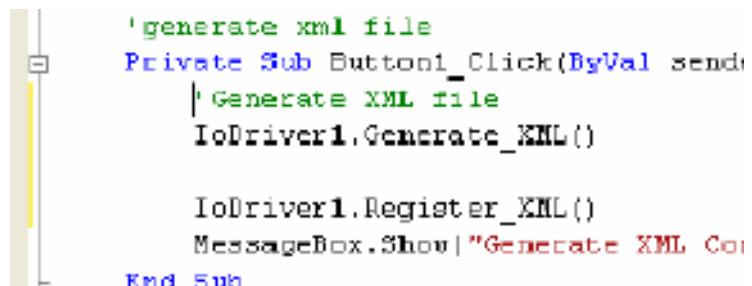
    'Separate Group Tag
    Public Sub Separate_Tag()
        Dim str_array() As String
        str_array = TXT_Tag_Name.Split(".")
        _Group_Name = str_array(0)
        _Tag_Name = str_array(1)
    End Sub

    'Get Tag Value
    Public Sub Get_Value()
    
```

Hình 4.2: Các lập trình các interfaces khi tạo Component

Trong ngôn ngữ .NET, Các interfaces là các properties và các methods được tạo ra bên trong một Class (hình 4.2). Với cách lập trình này, một đối tượng sẽ là một component độc lập với các đối tượng khác, giao tiếp với bên ngoài nhờ vào các properties và methods có thuộc tính public.

Nếu Client muốn sử dụng Component thì chỉ cần gọi các interfaces của component đó để tương tác (hình 4.3).



**Hình 4.3: Sử dụng Component IODriver1 bằng cách gọi hai interfaces
Generate_XML() và Register_XML()**

4.3. KIẾN TRÚC PHẦN MỀM HƯỚNG DỊCH VỤ (SERVICE ORIENTED ARCHITECTURE)

Kiến trúc phần mềm hướng dịch vụ (SOA) được định nghĩa là “một kiến trúc nhằm mục đích xây dựng các ứng dụng phần mềm là một tập hợp các components có mối liên hệ tương tác để tạo nên các dịch vụ được định nghĩa rõ ràng”[21]. Nói cách khác, SOA là kiến trúc để các Components tương tác với nhau hiệu quả tối ưu nhất nhằm tạo ra một hệ thống dịch vụ hoàn chỉnh nhất.

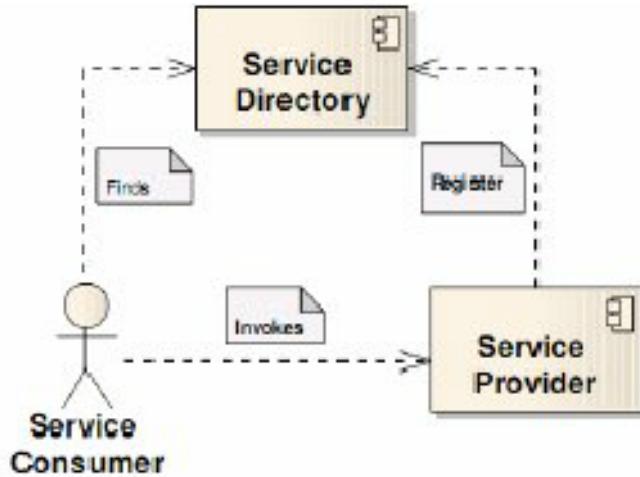
SOA là kiến trúc component hộp đen. SOA giúp che đi các phức tạp tại bất cứ vị trí nào nếu có thể, và ý tưởng hộp đen được tích hợp vào SOA. Hộp đen cho phép tái sử dụng các ứng dụng phần mềm hiện có bằng cách thêm vào một tính năng chuyển đổi phù hợp mà không cần quan tâm đến việc nó được xây dựng thế nào.

Các SOA components là “*loosely coupled*”. Cụm từ “*loosely coupled*” dùng để chỉ cách thức hai components tương tác bên trong SOA. Component này truyền dữ liệu qua component kia và tạo ra một yêu cầu, component kia sẽ thực hiện yêu cầu, và nếu cần thiết, chuyển dữ liệu ngược trở lại cho component thứ nhất. Cần nhấn mạnh đây là quy trình này được thực hiện trên nguyên tắc đơn giản hóa và tự động hóa. Mỗi một component sẽ tạo ra các dịch vụ đơn giản tới các components khác. Một tập các “*loosely coupled*” components ngoài khả năng làm cùng một công việc quen thuộc bên trong các ứng dụng, các components còn có thể kết hợp hoặc tái kết hợp theo nhiều cách khác nhau một cách cơ động. Điều này làm cho cơ sở hạ tầng IT trở nên tổng quát hóa, và mềm dẻo hơn.

Các SOA components phân tán trong ứng dụng phần mềm SOA liên kết với nhau thông qua các quy trình để tạo ra dịch vụ được định nghĩa rõ. SOA tạo ra một trật tự đơn giản các components có thể tạo ra dịch vụ kinh doanh phức tạp. Cấp dịch vụ được tạo ra gắn liền với các hoạt động kinh doanh tốt nhất – nói chung, đây là việc quản lý quy trình sản xuất.

Hình 4.4 mô tả kiến trúc hướng dịch vụ, bao gồm: *Service Component*, *Service Consumer Component*, và *Service Directory*. Service Component là component dịch vụ, nó chỉ hoạt động hợp lệ khi và chỉ khi thông tin của nó được đăng ký tại Service Directory. Consumer Component sử dụng dịch vụ của Service Component bằng cách tìm kiếm trong Service Directory xem có

dịch vụ Service Component mà nó cần hay không, nếu có, nó kích hoạt và sử dụng dịch vụ này.



Hình 4.4. Kiến trúc hướng dịch vụ SOA

Thể mạnh của việc sử dụng SOA được khẳng định nhờ vào khả năng của các components.

Khả năng tái sử dụng (reuse) và *tính kết hợp (composition)*. Điều này rất hữu ích cho việc tạo nên các quy trình nhanh chóng và đáng tin cậy.

Tính tái kết hợp (recomposition). Khả năng tháo lắp các quy trình sẵn có hoặc các ứng dụng khác dựa trên tập hợp dịch vụ.

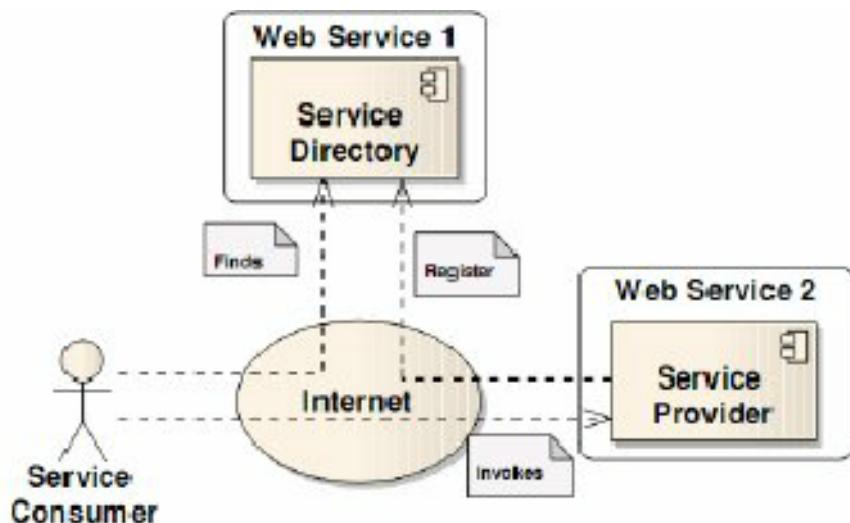
Khả năng thay đổi hệ thống một cách nhanh chóng, *chuyển đổi các nhà cung cấp dịch vụ, mở rộng dịch vụ, bổ sung các nhà cung cấp dịch vụ và khách hàng*. Tất cả các điều này có thể được thực thi một cách chính xác dựa trên khả năng kết nối tốt.

Khả năng xây dựng hệ thống một cách nhanh chóng. Điều này có được nhờ vào khả năng tích hợp hệ thống của nền tảng SOA.

Kiến trúc SOA có thể sử dụng internet làm giao thức truyền thông giữa các service components. Do vậy, SOA có thể định hướng kiến trúc cho Web Service[22]. Hình 4.5. mô tả mô hình tương tác của Web Service sử dụng kiến trúc SOA.

Từ định nghĩa, các tính chất, thế mạnh của kiến trúc hướng dịch vụ, tác giả rút ra được một số kết luận sau:

SOA là kiến trúc của phần mềm được lập trình theo hướng component. Tất cả các dịch vụ trong ứng dụng đều được xây dựng dưới dạng các components. Mỗi component là một dịch vụ, chúng tương tác để tạo nên kiến trúc SOA.



Hình 4.5. Mô hình tương tác trên SOA Web Service.

Do sự phù hợp với các ứng dụng có độ phức tạp cao, các ứng dụng thường xuyên phải thay đổi về thông số, có thể tương tác tốt trong môi trường Web. SOA hoàn toàn phù hợp để xây dựng một hệ thống SCADA từ cấp kết nối

thiết bị điều khiển đến cấp điều khiển giám sát, cho đến cấp hoạch định chiến lược kinh doanh.

Đặc biệt, với sự kết hợp SOA – Web Service, phần mềm SCADA không lệ thuộc khoảng cách vật lý, không lệ thuộc vào hệ điều hành mà các máy tính client đang sử dụng. Và kiến trúc SOA được hỗ trợ sâu rộng trong môi trường Visual Studio .NET

4.4. MÔI TRƯỜNG PHÁT TRIỂN PHẦN MỀM TÍCH HỢP VISUAL STUDIO.NET

4.4.1 CÁC THÀNH PHẦN CỦA VISUAL STUDIO.NET ENVIRONMENT

Visual Studio Environment (VSE) là môi trường phát triển phần mềm tích hợp (IDE) của Microsoft. VS dùng để biên tập (lập trình), biên dịch, kiểm lỗi và xuất bản các ứng dụng phần mềm trên nền tảng .NET. Toàn bộ cấu trúc của VS được Microsoft phát triển hướng Component và kiến trúc SOA. VS hỗ trợ tốt các ứng dụng phân tán (distributed application) nhờ tích hợp sẵn giao thức web có các công nghệ Hyper Text Transfer Protocol (HTTP), Extensible Markup Language (XML), và Simple Object Access Protocol (SOAP)[26].

Các loại ứng dụng có thể được xây dựng từ VSE bao gồm: console, giao diện người dùng, các ứng dụng winform, website, ứng dụng web hỗ trợ bởi Microsoft Windows, Windows Mobile, Windows CE, .NET Framework và Microsoft Silverlight.

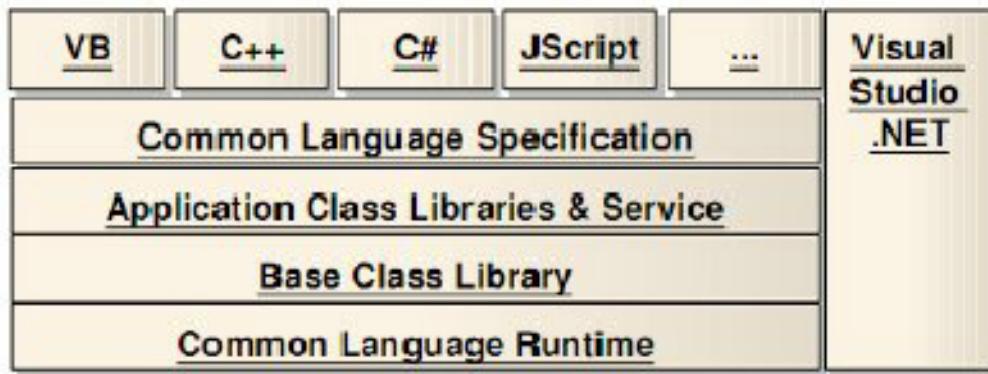
Môi trường VSE bao gồm các ba thành phần chính **Designer**, **Debug**, và **Runtime**.

Designer được người dùng sử dụng để thiết kế, cấu hình và lập trình để tạo ra ứng dụng. Designer bao gồm cửa sổ thiết kế giao diện *UI Design* để thiết kế giao diện đồ họa cho phần mềm, giao diện đồ họa có thể là Winform hoặc là Webform. *Code Editor* là cửa sổ hiển thị lập trình, nơi các nhà lập trình có thể viết các đoạn mã bằng các ngôn ngữ mà VSE hỗ trợ (VSE hỗ trợ nhiều loại ngôn ngữ VB.NET, C#, C++ trên cùng một môi trường) để quy định các tính năng và hành động của sản phẩm ứng dụng. *Toolbox* chứa các công cụ đồ họa điều khiển tính năng như button, label, textbox, ... hỗ trợ việc thiết kế User Interface (UI). Ngoài các điều khiển sẵn có, VSE còn hỗ trợ người dùng tự tạo ra các bộ tool riêng và tích hợp vào toolbox của VSE. Dựa vào đặc điểm này, người dùng có thể tạo các bộ công cụ phù hợp nhất cho công việc của mình có thể sử dụng đi sử dụng lại nhiều lần cho các dự án.

Debug là công cụ mạnh của VSE hỗ trợ việc bắt lỗi, bẫy lỗi và các hướng dẫn mà VSE hỗ trợ người lập trình.

Runtime là tính năng thực thi ứng dụng được tạo ra bởi Designer. Runtime trong VSE hỗ trợ cả các ứng dụng Winform và Webform.

Hình 4.6. mô tả cấu trúc của .NET framework, là nền tảng của VSE.



Hình 4.6. Mô hình .NET Framework

.NET framework được cấu thành từ nhiều components như hình 4.6. Trong đó component quang trọng bậc nhất là Common Language Runtime (CLR). Tất cả các ứng dụng .NET đều lệ thuộc vào CLR. CLR cung cấp một tập các loại dữ liệu, là nền tảng của VB, C#, và các ngôn ngữ khác trên nền .NET framework. Vì nền tảng này là duy nhất cho tất cả các ngôn ngữ lập trình .NET.

Common Language Specification (CLS) là tập các luật để các ngôn ngữ lập trình .NET giao thoa nhau. Ví dụ CLS đảm bảo kiểu interger trong VB giống như ở C#. Bởi hai ngôn ngữ đã thống nhất về kiểu dữ liệu, do vậy chúng có thể chia sẻ dữ liệu lẫn nhau. CLS định nghĩa không chỉ kiểu dữ liệu mà còn cách gọi method, cách kiểm soát lỗi, và nhiều tính năng khác. Một class định nghĩa trong VB có thể sử dụng được ở C#, một method định nghĩa ở C# cũng có thể được gọi từ VB.

Tầng giữa của mô hình là Application Class Libraries and Services. Tầng này khai báo tập hợp các thư viện và APIs đã được tạo sẵn để hỗ trợ trực quan việc thiết kế giao diện ứng dụng đồ họa APIs hoặc Windows.Forms, database APIs thông qua ADO.NET, XML.

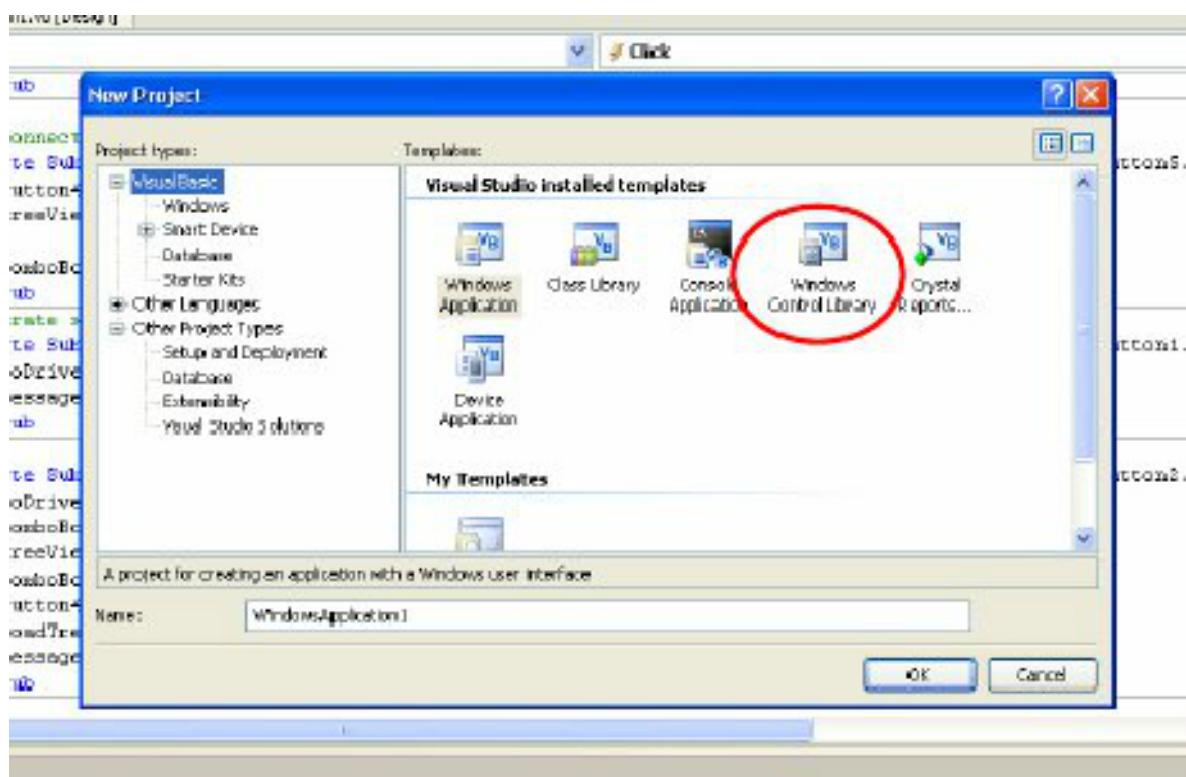
Visual Studio.NET là một phần quan trọng của .NET framework bởi nó cung cấp các phương tiện để người lập trình tiếp cận framework ở bất cứ tầng nào.

Trên cùng bên trái là các ngôn ngữ được Microsoft hỗ trợ trong VSE để người lập trình viết code tạo ứng dụng, bao gồm: VB.NET, C#, C++, Jscript, ...

4.4.2 CÁC TÍNH NĂNG CỦA VISUAL STUDIO.NET ENVIRONMENT

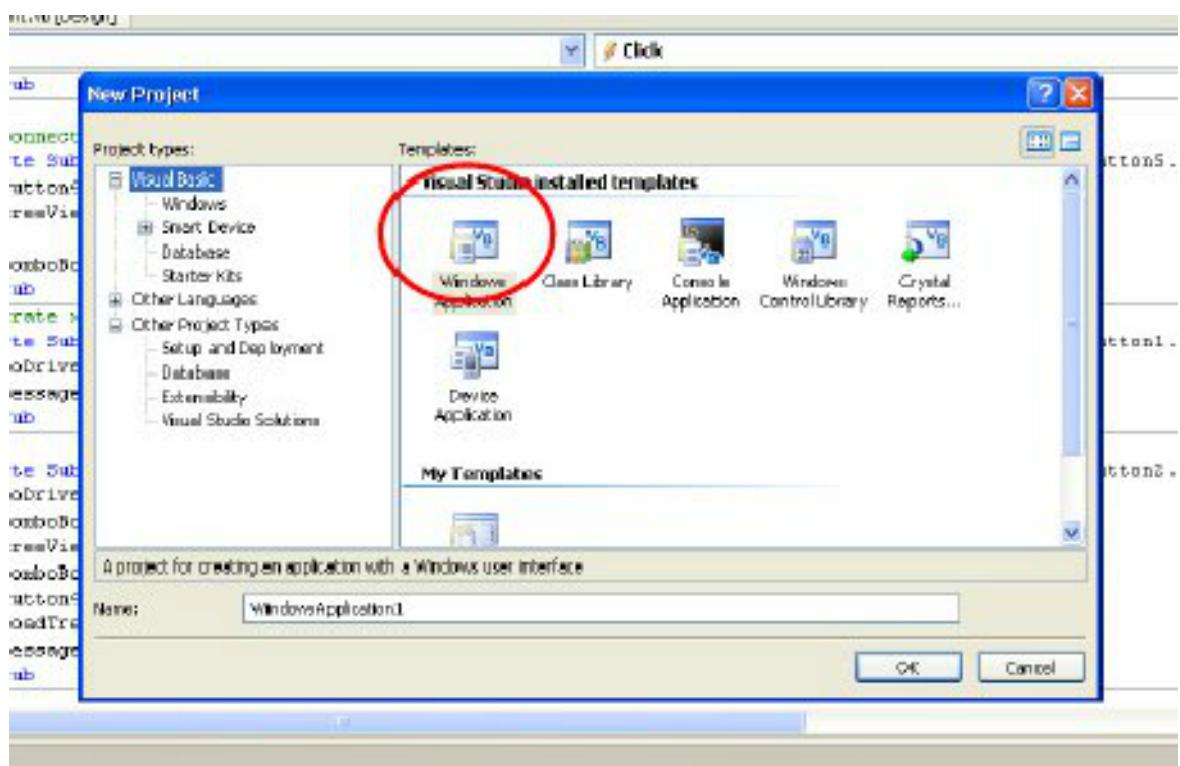
Môi trường lập trình VSE có thể xem như môi trường đa tính năng, nó hỗ trợ việc “chế tạo” ra các component, hỗ trợ việc liên kết các components để tạo các ứng dụng desktop, ứng dụng web, và hỗ trợ việc đóng gói phần mềm.

Để xây dựng nên các component ứng dụng cho Windows Form, VSE tạo ra project dạng “Windows Control Library” (hình 4.7), trong project này, người dùng có thể tạo nên các Control Components chuyên dụng để sử dụng cho mục đích riêng.



Hình 4.7: Tạo mới Windows Control Library Project để xây dựng các Windows Control Component

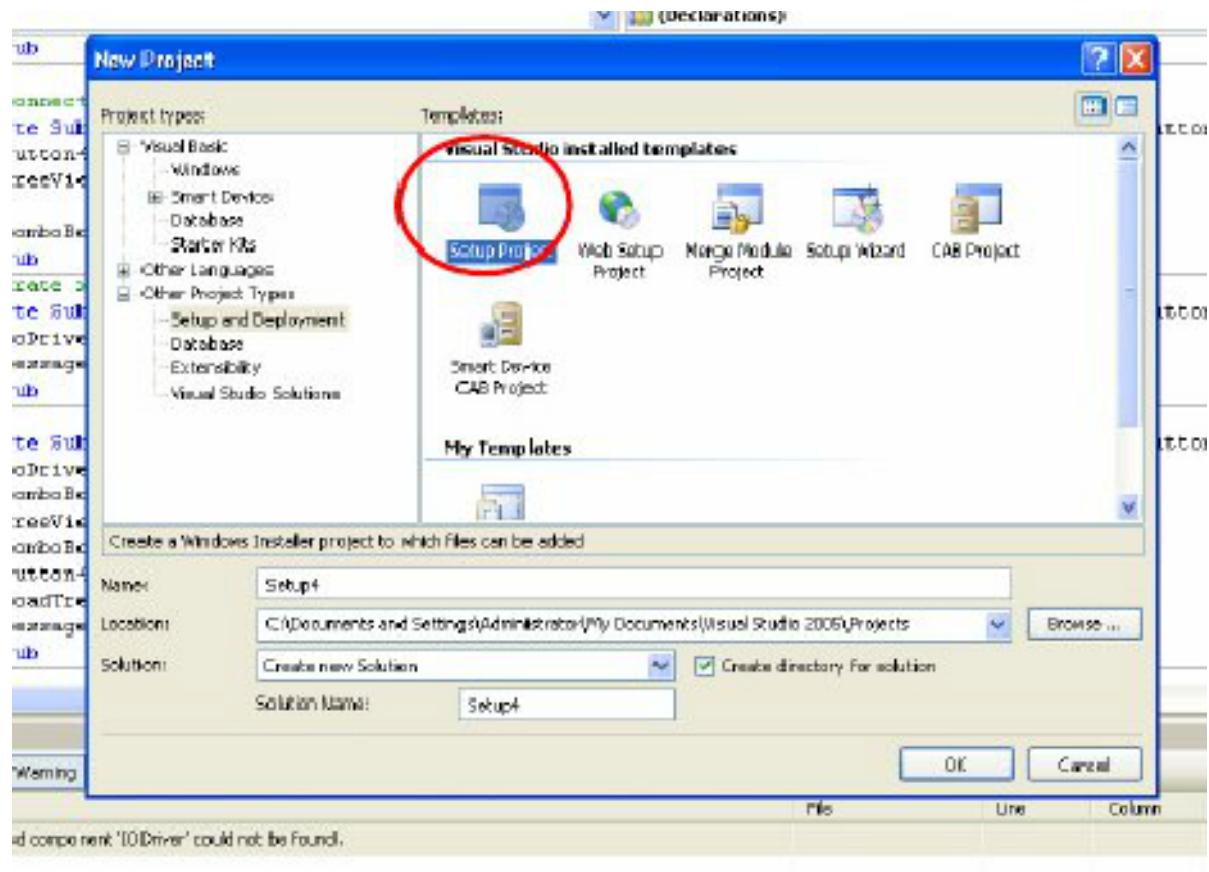
Để tạo ứng dụng phần mềm desktop, người dùng cần tạo một “Windows Application” Project như hình 4.8. Tại project này, người dùng có thể tham chiếu đến các component để sử dụng dịch vụ của component đó nhằm tạo ra một ứng dụng desktop. Trên nền tảng lập trình hướng component, các ứng dụng được tạo ra từ tham chiếu đến các components sẽ không lệ thuộc vào vị trí của components này trong mạng máy tính.



Hình 4.8: Tạo mới Windows Application Project để xây dựng các ứng dụng desktop

Sau khi xây dựng xong một ứng dụng, người dùng có thể đóng gói sản phẩm của mình bằng cách tạo một project đóng gói như hình 4.9. Việc đóng gói tạo

điều kiện thuận lợi cho việc thống nhất và phân phối các component phần mềm tạo nên sản phẩm.



Hình 4.9: Tạo mới một dự án đóng gói sản phẩm

CHƯƠNG 5: XÂY DỰNG PHẦN MỀM SCADA TRONG MÔI TRƯỜNG VISUAL STUDIO.NET

Trước khi bắt đầu xây dựng phần mềm SCADA trong môi trường Visual Studio, tác giả xin được trình bày tóm tắt lại các đối tượng chính trong một phần mềm SCADA. Theo phần 2, một phần mềm SCADA bao gồm: *IO-Driver* là đối tượng dùng để thu thập dữ liệu thời gian thực từ các thiết bị điều khiển, *Designer* (đối tượng dùng để thiết kế ứng dụng), *Runtime* (đối tượng dùng để thực thi ứng dụng), *Data Center* dùng để lưu trữ và chế biến dữ liệu, *Network Connector* dùng để truyền thông qua lại giữa các ứng dụng SCADA đang cùng hoạt động, các đối tượng hiển thị và điều khiển: *trend*, *alarm*, *report*, *textbox*, *label*, ...

5.1. SO SÁNH PHẦN MỀM VISUAL STUDIO VÀ PHẦN MỀM SCADA

So sánh các thành phần mà phần mềm SCADA cần có với các thành phần chức năng của phần mềm Visual Studio, ta có được bảng so sánh như bảng 5.1.

Theo bảng 5.1. ta rút ra được VSE chỉ khác phần mềm SCADA hiện tại ở chỗ VSE chưa hỗ trợ IO Driver chuyên dụng để kết nối thu thập dữ liệu từ thiết bị điều khiển công nghiệp. VSE chưa hỗ trợ một số controls như vẽ đồ thị thời gian thực (Trend), Cảnh báo thời gian thực (Alarm) và báo cáo dữ liệu số (Report), để có được các controls này, ta hoàn toàn có thể chế biến từ các

No.	SCADA Software	Visual Studio	Note
1	Designer	Designer	Phần này giống nhau hoàn toàn giữa VSE và SCADA software. Dùng để thiết kế giao diện đồ họa, cấu hình và lập trình cho ứng dụng. Hơn nữa, việc thao tác thiết kế trên VSE được đánh giá là nhuần nhuyễn hơn.
2	Runtime	Runtime	Cả VSE và phần mềm SCADA đều sử dụng cơ chế Runtime để chạy ứng dụng được xây dựng ở Designer.
3	Data Center	Data Base	VSE hỗ trợ kết nối dễ dàng nhiều loại cơ sở dữ liệu, việc chế biến dữ liệu đơn giản. Do vậy có thể xem VSE hỗ trợ tốt về mặt Data Center cũng giống như SCADA software.
4	Network Connector	Remoting Technology	Được xây dựng dựa trên components, kỹ thuật remoting trong VSE bao gồm cả dịch vụ web - một cơ chế truyền thông diện rộng tốt nhất hiện nay. Do vậy VSE tốt hơn bất cứ phần mềm SCADA hiện tại.
5	IO Driver		VSE không tích hợp cơ chế truyền dữ liệu từ thiết bị điều khiển công nghiệp lên máy tính.
6	Controls ToolBox: textbox, label, button, trend, alarm, report, ...	Controls ToolBox: textbox, label, button, ...	VSE có tập hợp các controls cơ bản phong phú đa dạng hơn nhiều so với phần mềm SCADA. Tuy nhiên, để phù hợp với phần mềm SCADA, VS cần chế biến thêm các control components dựa trên các controls sẵn có .

Bảng 5.1. So sánh tương quan tính năng giữa phần mềm SCADA và Visual Studio

controls sẵn có của VSE. Còn lại các tính năng khác, xem như VSE có thể tốt hơn hoặc bằng so với các phần mềm SCADA hiện tại.

Tuy nhiên, khi xem xét về kiến trúc phần mềm và kỹ thuật lập trình ứng dụng, như đã trình bày trong phần 2, các phần mềm SCADA lớn hiện tại đều được phát triển từ kỹ thuật lập trình cũ (lập trình modular) chạy trên nền tảng Windows của Microsoft và do Microsoft đưa ra. Trong khi VSE là sản phẩm trực tiếp của Microsoft, được Microsoft phát triển liên tục (hiện nay đã có phiên bản 2010), nền tảng của VSE được hiện đại hóa liên tục theo các công nghệ phần mềm mới:

- Lập trình hướng component theo kiến trúc hướng dịch vụ (SOA) cho VSE.
- Về truyền thông, VSE tích hợp công nghệ Web, làm cho các ứng dụng xây dựng từ VSE càng ngày càng trở nên “trong suốt” về khoảng cách vật lý. Việc truyền thông không còn phân biệt giữa tương tác trên cùng một máy tính, trong mạng LAN hay mạng Internet.
- Bởi vì VSE tuân theo kiến trúc hướng dịch vụ SOA của các components nên việc hỗ trợ người dùng tự tạo các tính năng theo ý riêng rất mạnh mẽ. Người lập trình có thể cập nhật thêm tính năng cho các controls hiện có, hoặc xây dựng thêm các controls mới phù hợp với công việc riêng.

Dựa trên các phân tích trên, VSE hoàn toàn có thể trở thành một phần mềm SCADA hiện đại nếu ta xây dựng thêm IO Driver, một số control mới như Trend, Alarm và Report và cập nhật thêm tính năng cho các controls sẵn có. Phần mềm SCADA mới này mang trong nó đầy đủ các công nghệ hiện đại,

phù hợp với các hệ thống điều khiển phân tán (Distributed Control Systems), phù hợp với việc thay đổi công thức sản xuất liên tục, dễ dàng cập nhật hệ thống sản xuất, làm nâng cao năng suất sản xuất công nghiệp.

Xem xét sự tương quan về mặt giá thành của các gói phần mềm, ta có một vài so sánh ở bảng 5.2.

Với chi tiết giá thành phần mềm như ở bảng 5.2. Việc xây dựng một phần mềm SCADA hiện đại trong môi trường VS là khả thi và cần thiết để phục vụ cho quá trình tự động hóa, hiện đại hóa sản xuất.

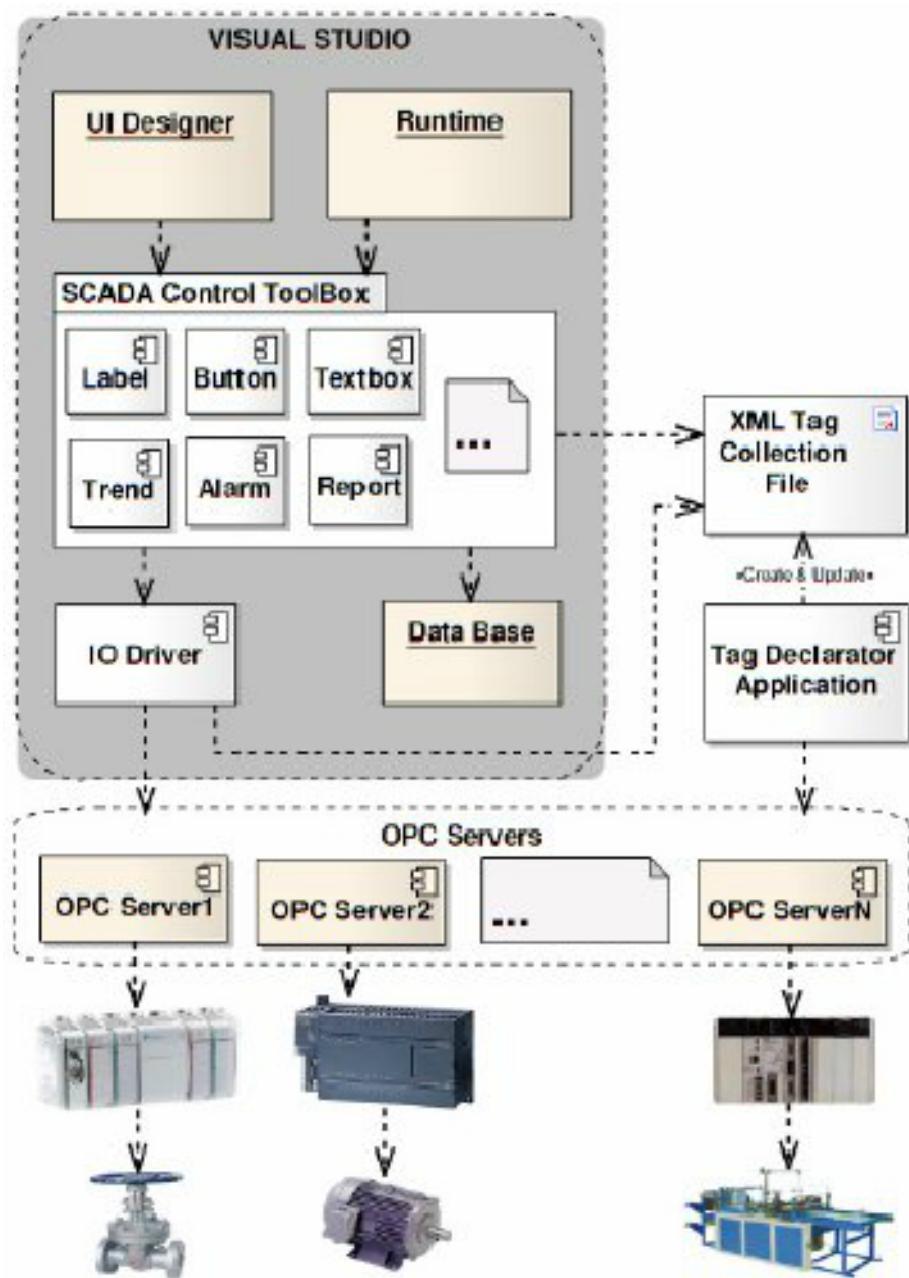
No.	Software Name	Software Price
1	WinCC 8,000 tags	8,531 USD
2	Intouch 16,000 tags	6,750 USD
3	Visual Studio 2010 Professional	799 USD

Bảng 5.2. Giá thành chi tiết của một số phần mềm

5.2. PHÂN TÍCH VÀ THIẾT KẾ PHẦN MỀM SCADA TRONG VISUAL STUDIO ENVIRONMENT

Đi vào phân tích và thiết kế phần mềm SCADA trong môi trường VSE, tác giả đề xuất mô hình tính năng phần mềm như hình 5.1. Trong mô hình này, data từ các thiết bị điều khiển công nghiệp được đưa lên phần mềm SCADA để giám sát điều khiển theo giao thức OPC. Trong giao thức OPC, các OPC-DA servers do (OPC Foundation sản xuất) được sử dụng để kết nối và thu thập dữ liệu từ các PLCs điều khiển các thiết bị. Phần mềm SCADA muốn kết nối được với các OPC servers này thì nó phải chứa đựng một OPC Client

Component bên trong nó. Trong mô hình này, Tag Declarator và IO Driver mang bên trong nó một OPC Client có tên *OPC.NET Wrapper* được cung cấp bởi hãng Kepware Inc. Sự tương tác giữa OPC Clients và OPC servers tạo ra giao thức truyền thông công nghiệp OPC, đảm bảo độ tin cậy trong việc điều khiển giám sát các thiết bị công nghiệp (chương 3).



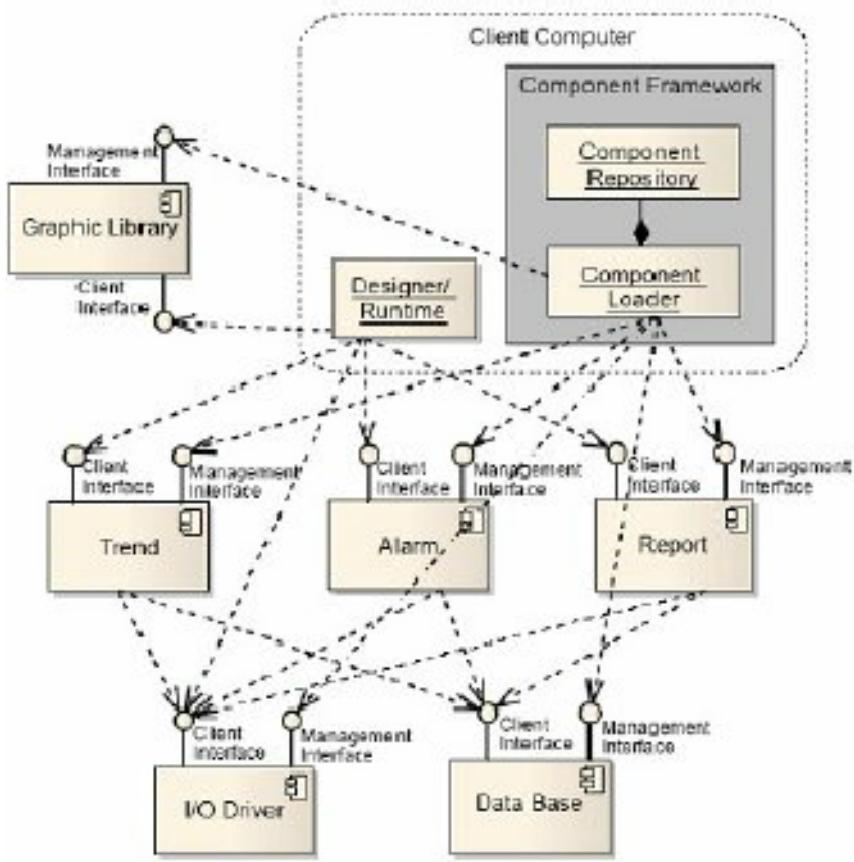
Hình 5.1. Mô hình tính năng phần mềm SCADA xây dựng trong môi trường Visual Studio

Trong mô hình được mô tả bởi hình 5.1, IODriver là một component có các giao diện với các OPC Servers và các SCADA components khác. Các tính năng SCADA như trend, alarm, report cũng là các component có các giao diện để cung cấp dịch vụ ra bên ngoài.

Hình 5.2 mô tả mô hình tương tác giữa các SCADA components trong môi trường hướng dịch vụ. Ở đây, các server components có các *managements interfaces* dùng để cung cấp địa chỉ của mình. Clients ở đây là các chương trình designer và runtime, khi muốn sử dụng dịch vụ từ các server components thì chúng gửi yêu cầu tìm kiếm đến component framework của .NET framework, nhờ vào các *managements interfaces* cung cấp địa chỉ cho component framework và địa chỉ này được trả cho clients để clients tương tác và sử dụng dịch vụ thông qua các client interface của các *server component* này.

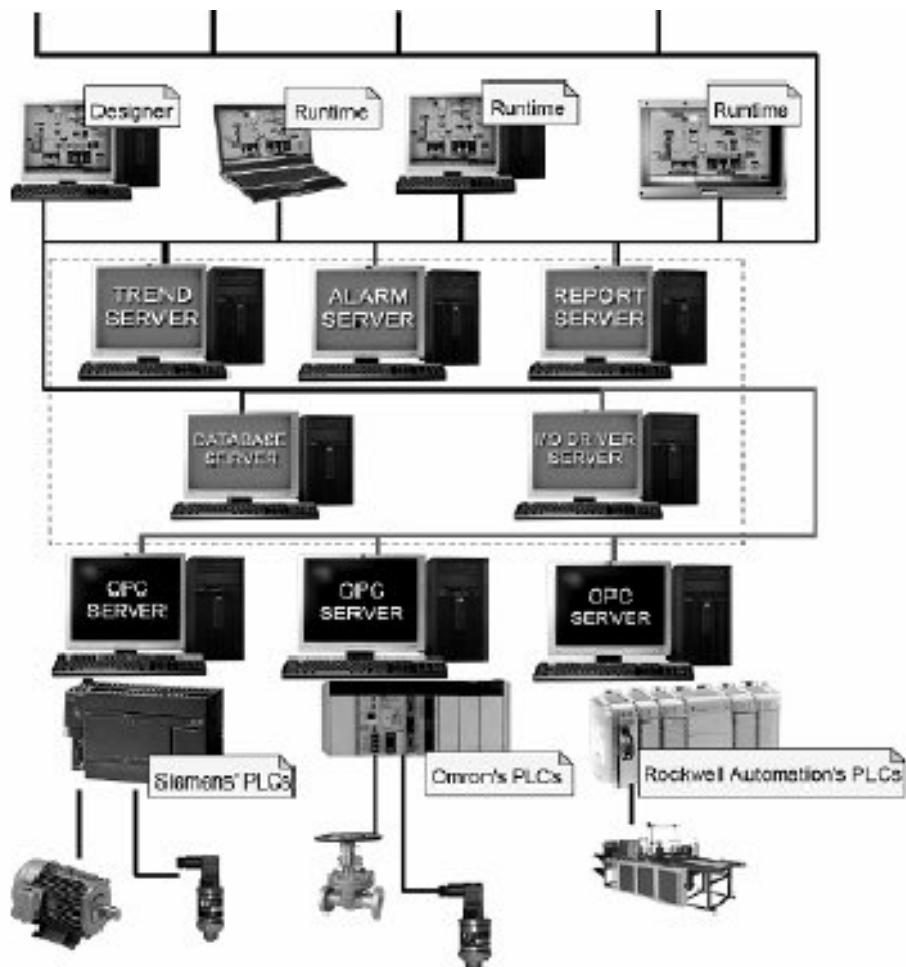
Trong môi trường .NET, việc tương tác giữa các components không bị bó gọn trong một máy tính, các components có thể tương tác với nhau trong một hệ thống mạng ethernet dễ dàng hệt như tương tác trong cùng một máy tính. Do vậy, đối với hệ thống SCADA lớn và phân tán, mô hình SCADA nền tảng .NET có thể được triển khai như hình 5.3.

Trong hệ thống SCADA hình 5.3, mỗi SCADA component như Alarm component, trend component, report component, ... được cài đặt trên một máy tính server riêng biệt để tận dụng tốc độ xử lý mạnh của máy tính này, nhằm chuyên biệt hóa từng dịch vụ cho hệ thống SCADA, nâng cao độ mạnh xử lý và tính cơ động cho toàn bộ hệ thống SCADA.



Hình 5.2. Mô hình tương tác trong phần mềm SCADA nền tảng .NET

Cấu trúc phần mềm SCADA lập trình theo hướng component và tương tác trong môi trường SOA được đánh giá là hệ thống mở và uyển chuyển bởi khả năng dễ dàng cập nhật phiên bản, vá lỗi, Việc cập nhật chỉ diễn ra trong một hoặc một vài component, không ảnh hưởng đến hoạt động của toàn bộ hệ thống SCADA, tận dụng được tài nguyên mạng LAN trong công ty, nhà máy, không phải cài đặt tất cả hệ thống phần mềm vào cùng một máy tính SCADA. Với nền tảng .NET, hệ thống phần mềm SCADA không những có thể triển khai trong mạng nội bộ mà còn có khả năng mở rộng tương tác trên hệ thống mạng internet nếu vấn đề bảo mật được đảm bảo an toàn.



Hình 5.3. Hệ thống mạng SCADA cỡ lớn nền tảng .NET

Hình 5.4. là mô hình thiết kế của IODriver Component. Trong Interface của Component này, mỗi property và method có vai trò như sau đây.

IODriver

PROPERTIES:

Server_Name_Object là thuộc tính lấy tập hợp tên các OPC Servers đang hiện hữu trên máy tính chứa IODriver Component.

Server_Name là thuộc tính gán và lấy tên của OPC Server được chọn

Group_Name dùng để gán tên Group

Update_Rate để gán giá trị tốc độ cập nhật của group tương ứng ở trên

Tag_Name dùng để gán tên Tag

Tag_Description dùng để gán mô tả cho tag tương ứng

Tag_Address dùng để gán địa chỉ cho tag tương ứng ở trên

METHODS:

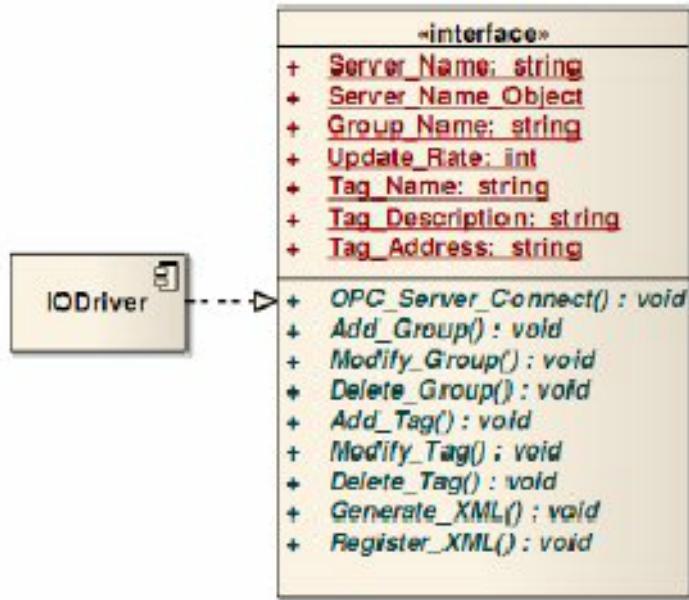
OPC_Server_Connect() tạo kết nối từ OPC Client tới OPC Server được chọn

Add_Group(), Modify_Group(), Delete_Group() tương ứng dùng để thêm mới nhóm, cập nhật nhóm và xóa nhóm

Add_Tag(), Modify_Tag(), Delete_Tag() tương ứng dùng để thêm mới tag, cập nhật tag và xóa tag

Generate_XML() dùng để tạo file XML chứa cấu trúc group và tag đã được tạo ra.

Register_XML() được dùng để thực thi file cấu trúc XML để đăng ký hệ thống group và tag để thu thập, giám sát và điều khiển



Hình 5.4: Mô hình cấu trúc và giao diện tính năng của Component IO Driver

Hình 5.5 mô tả mô hình của ứng dụng khai báo tag (Tag Declarator). Ứng dụng này gói bên trong nó là IODriver Component. Nó sử dụng hầu hết các properties và methods của IODriver để tạo nên cấu trúc XML file lưu trữ thông tin OPC Server, Group và tag. Cấu trúc cú pháp của XML file này có dạng như sau:

```

<Root>
  <Server Name= “...” />
  <Group>
    <GroupName> ...</GroupName>
    <UpdateRate>...</UpdateRate>
    <Tag TagName="..." Address="..." />
    ...
  </Group>

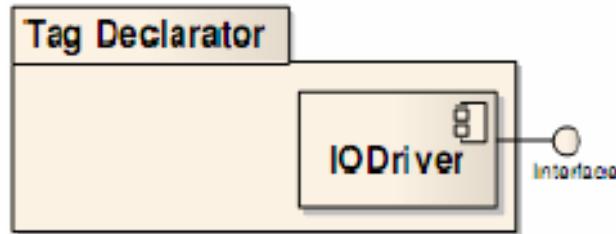
```

```

<Tag TagName="..." Address="..."/>
...
</Group>
<Group>
...
</Group>
...
</Root>

```

Trong đó *Server* để chỉ OPCServer được dùng có tên là *Name*. *Group* là nhóm các tags có cùng đặc điểm tốc độ cập nhật *UpdateRate*. *GroupName* là tên của Group tương ứng. Mỗi tag đều có hai đặc trưng là tên tag *TagName* và địa chỉ *Address*.



Hình 5.5. Mô hình cấu tạo và tính năng của ứng dụng khai báo tag Tag Declarator

Ứng dụng Tag Declarator có tác dụng khai báo tag đến các OPC servers cụ thể và lưu quá trình này thành file XML cấu hình để các OPC Servers thực thi và kết nối đến OPC Servers. Kết thúc quá trình này, ứng dụng Tag Declarator được trả về trạng thái rỗ.