L'AUTOMATISATION INDUSTRIELLE
7-9 juin 1999
3rd International Conference
INDUSTRIAL AUTOMATION
June 7-9 1999
École de technologie supérieure
(ETS), Montréal

ACTES PROCEEDINGS

Édité par / Editor
Guy Bogdadi

Publication de / of
Association Internationale
pour l'Automatisation Industrielle (AIAI)
International Association
for Industrial Automation (IAIA)
AUTOMATISATION INDUSTRIELLE
INDUSTRIAL AUTOMATION

École de technologie supérieure
Montréal
7-9 juin 1999 / June 7-9 1999

ACTES DE LA CONFÉRENCE
CONFERENCE PROCEEDINGS

ÉDITÉ PAR / EDITOR
Guy Bogdadi

PUBLICATION DE / OF
Association Internationale de l'Automatisation Industrielle (AIAI)
International Association for Industrial Automation (IAIA)

ISBN : 2-9802946-2-4
F. Jurado, J. Carpio, J. Rodriguez
Simulation of Turbine-Governing Systems

F. Jurado, J. Carpio, J. Rodriguez
Optimal Location of Facts Based on Contingency Analysis

J. Ragot, L. Fontaine, G. Mourot
Segmentation d'electrocardiogrammes par réseau de modèles locaux

Systèmes distribués

C.-A. Brunet, R. Gonzalez-Rubio, M. Tétreault
Vers un modèle complet de pilote pour véhicules autonomes: une approche basée sur les systèmes multi-agents hétérogènes

R. W. Brennan
Metrics for Evaluating Alternative Manufacturing Control Architectures

J. P. G. Bouchard, C. Monette
Schémas de Commandes de Moteurs Sécuritaires par Automates Programmables Industriels

L. Cauffriez, J. Defrenne
Influence sur la cohérence temporelle des informations des méthodes d'accès au médium, des modes d'adressage dans les réseaux de terrain

L. Cauffriez, B. Philippe
Classification des méthodes d'accès au médium de communication, des modes d'adressage et types de trafic dans les réseaux de terrain

J. Bonnet, E. Gnaedinger, F. Lepage
Proposition d'un modèle d'architecture de communication pour des applications industrielles distribuées et multimédia

B. Conrad, F. Hermann, J.-M. Thiriet, M. Robert
Vers une boîte à outils pour le conception de systèmes intégrés distribués

J. A. Fntos-Redondo, J. M. Giron-Sierra
Automatic Code Generator for a Distributed Control System

Détection de défauts et diagnostic

M. Small, M. Thomas, A. A. Lakis
Utilisation de la Modélisation ARMA pour la Détection des Défauts dans les Poutres

A. E. Ouaif
Approche Multisensorielle pour la Détection des Défaillances dans les Procédés de Fabrication

A. Dia, C. Thibeault, C. S. Gargour, J. L. Houle
On the Use of Wavelet Analysis for IC Testing

O. Seidel, F. Kratz, H. Dusser, A. Mouchette
Diagnostic d'un actionneur électrique

G. Gelle, F. Delue, M. Colas, G. Delaunay
Les statistiques d'ordre supérieur appliquées à la caractérisation de non-linéarités. Application au diagnostic vibratoire de machines tournantes

M. Colas, G. Gelle, G. Delaunay
Application des méthodes temps-fréquence au contrôle non destructif d'un système d'entraînement par courroie. Analyse et classification de signatures acoustiques

O. Adrot, D. Maquin, J. Ragot
Estimation d'état généralisé sur horizon glissant. Application au diagnostic

D. Maquin, O. Adrot, J. Ragot
Estimation d'état de systèmes incertains. Application à la détection de défauts de capteurs

L. D. Seneviratne, P. Visw Works
Fault Detection for Automated Self-Tapping Screw Insertions

A. Hanomolo
A Neural Classifier for Fault Diagnosis: An Entropy Approach

CAO / FAO

CAD / CAM

C. Mascle, C. Fortin, D. Dal le, S. Morissette
Évaluation des gammes d'assemblage par l'analyse économique

S. Shu, D. H. Norrie, B. Zhang
Concurrent Design Oriented Assembly Process Analysis and Evaluation System

J.-F. Chatelain, C. Fortin
A CAD/CAM Integrated Balancing System for Blank Part Machining

V. François, J. C. Cuillière, M. Guerry
Remaillage automatique tridimensionnel appliqué aux modifications de modèles

L. Baron
An Optimal Surfacing Post-Processor Module for 5-Axes CNC Milling Machines

P. Chedmail, C. Le Roy
Manipulation of Mechanisms and Manipulators: A Distributed Approach

Système expert et intelligence artificielle

M. L. Doumbia, G. Roy, V. Rajagopalan
Méthodologie de développement d'un système expert d'aide à la conception des entraînements électriques

C. Mascle, M. Balazinski
Réseaux neuronaux et logique floue en conception d'assemblage

J. Sun, D. Xue, D. H. Norrie
An Intelligent Production Scheduling Mechanism Considering Design and Manufacturing Constraints

S. Zhang, Q. Sun
Distributed Control of Manipulators Using a Multi-agent Approach

E. C. Martinez
Agent-Based Process Automation using Reinforcement Learning- Applied to a Buffer Tank
AUTOMATIC CODE GENERATOR FOR A DISTRIBUTED CONTROL SYSTEM

Jose A. Frutos-Redondo¹, Jose M. Giron-Sierra²

¹Dept. Automatica, Escuela Politecnica
Universidad de Alcala, 28871 Alcala de Henares, Madrid, Spain
e-mail: atrfrutos@alcala.es, Tf: (34) 91 885 66 14

²Dept. Arquitectura de Computadores y Automatica, Fac. Fisicas
Universidad Complutense de Madrid, Ciudad Universitaria
28040 Madrid, Spain
e-mail: gironsi@dia.ucm.es, Tf: (34) 91 394 46 87

Abstract:

This paper presents an automatic code generator, designed for the fast implementation of monitoring and control applications, on the basis of a distributed control system. The code generation comprises both the operator interface and the specification of the system behavior. With the code generator it is easy to develop applications, in an interactive way. The code generated is Borland C++, open to user modifications.

1. Introduction

The development of the software in computer-based systems for supervision and control, is usually a time-consuming task. In particular, the operator interface takes an important part of the work. Tools that make easy the creation of the code, including the man-machine interface, are welcome. This need has been emphasized by influential experts, as in [1].

Helped by modern methods and technologies, several contributions have been done in this context. Following icon-based approaches, [2] describes a programming tool named OSCAR for distributed control, and [3] another tool for systems using the MAP protocol. More recent works embrace the object-oriented paradigm for computer control systems [4], also considering fieldbus-based systems [5] and devices [6]. Agents too have been applied, in connection with Petri-net models [7].

The commercial side of the field - companies working on data acquisition and control systems - reacted in a similar way. A variety of software tools are now offered, focusing mainly on a friendly operator interface, with graphics, windows and mouse. One of the most significant acronyms employed here is MMI (man-machine interface), that could include compatibility solutions regarding a set of PLCs and data acquisition cards from several firms. For instance, TAYLOR Industrial Software with the package Process Windows MMI: a tool for the generation of SCADA and other control applications, on PC, compatible with Allen-Bradley, Modicon, etc. products [8]. The company LABTECH offers VISION, with an extensive compatibility list, that includes most of the data acquisition systems on the market [9]. Taking into consideration data-processing needs, the company MICROCAL has adapted his software for data analysis and visualization, ORIGIN [10], to interface with data acquisition systems.

In particular, it is worth to mention the efforts from companies traditionally related with instrumentation and data acquisition. They want to provide software to make the most of their devices. The main ways of support offered now are tools for operator interface development, and for the development of control applications based on flow-charts. In [11] there is a comparison of 58 products belonging to this category.

A further step is to create graphical tools that generate software for process control, not only to handle control units. This is the objective of our work.

There are industrial control systems based on big or medium range computers. But there is a dominant trend to use Intel-based computers (PCs). Many of the software products for data acquisition and control are primarily intended for PCs. We adhere to this trend, so the focus of our research is on systems that are centered on a PC (here resides the operator interface), and that can employ several supervision and control nodes connected to this PC.

2. Automatic generation of code

The users of monitoring and control systems are frequently not experts of electronics, computers, sensors, and control theory and practice. There are many examples of this, from the small scale of a scientist building a laboratory set-up, or a quality control system with one PC, to other scales that can exhibit long distances and functional complexities (in such cases a distributed control system comes as the proper alternative). The general orientation of our research is to provide the means, as an integrated solution of hardware and software, for an easy implementation of supervision...
and control systems by non-expert users. An important part of our effort has been the development of a software tool that, with graphics, windows and mouse, can be employed to create the operator interface, and the supervision and control functionality, in a friendly way. The tool is mainly intended for a distributed modular system that we shall describe below. But it can also be used for other architectures (for instance, the simple case of a PC interfacing with some sensors and actuators).

With the tool, the user defines the supervision and control functions required, in an interactive way. Once finished the definition, the automatic code generator (an important part of the tool) is invoked. The generator writes a source program in C language. This program is ready for compilation, to create an executable program, to be run in the operational system. As the program generated is well structured and documented, it can serve as a basis for functional enrichment, by adding user routines.

Thinking that perhaps the user may wish to modify, after some time, the operator interface, the tool has an interesting feature: to save a description of the definitions given by the user (to create the operator interface and the system functionality) in an intermediate file. Recovering the file, the tool can be used for an easy modification of the operator interface, and to generate a new application. Figure 1 shows a diagram of the code generator operation.

For the development of our tool, in Borland C++, we employed the library "Real-Time & User Interface Tools" from Quinn-Curtis.

- Numerical panels, using sets of 7-segment digits, for the instantaneous information about analog data values.

- Button panels, that can be used for two purposes: for information about events, and to transmit orders from the operator. For instance, the color of a button can tell about the normal or abnormal status of an aspect of the controlled process.

The X-T graphic panels can display simultaneously several curves, from several sensors. The buttons can work as push-button or toggle, independent or coupled. Figure 2 shows the appearance of the three types of windows that can be combined for the operator interface.

To build the operator interface, the user starts choosing the windows he considers are pertinent. There is a menu bar with the main options. A type of window is selected, our tool displays a default instance of it on screen. With the mouse, the location and size of the window is modified, to get the desired result. Dialog windows complete the task of specification: number of channels to be displayed as curves, number of buttons, range of the data values, colors, etc.

During the building of the operator interface, any window already defined, can be modified as desired (for example, resize or move it, or change colors, etc.). Once a satisfactory design is obtained, we can generate an intermediate file to save this design, so we can modify it easily in the future, and/or generate the source program in C language.

There are two versions of the source program we can generate: one for operator interface only, and the other for a complete application using our distributed control architecture. In the first case, the source program does not interact directly with the external process to be controlled. Instead, the program has open entries for user routines that implement the input of data from the process, and the control functions on the process. What the program solves in this case, is the handling of the windows for data visualization and operator orders receiving.
4. The distributed supervision and control system

As an important part of our general objective - the quick and easy implementation of control systems -, we devised a modular architecture for distributed configurations. The system has been conceived for a simple setting up, with fairly powerful nodes and a robust communication scheme.

Each node is based on a small footprint embedded MS-DOS PC, and includes all the hardware needed for the direct interface with industrial sensors and actuators. A node is able to handle up to 8 analog inputs, 2 analog outputs, 6 digital inputs and 6 digital outputs. The electronic circuits we designed to complement the embedded PC interfacing capability, have A/D and D/A 12 bit serial converters, and signal conditioning and adaptation for a set of possible analog data sources: pt100 temperature sensors, pH and other electrochemical sensors, 4-20 mA loops, 0..1 volts inputs. There is also power amplification for analog outputs, and isolation for the digital I/O.

The modular concept allows for an autonomous operation of each node as a data-logger and/or a controller. The behavior of the node is configured through the communication port of the node. For control functions, the user can specify a P.I.D. regulation, or, more freely, a Transfer Function (the user specifies the numerator and denominator) to be executed by the node. In addition to analog control, the node can exert logic control.

The node has intelligent features, by means of a set of rules (for example: if this signal breaks this limit, then take this action), specified during the configuration. The rules serve to send alarm signals to the operator, and, if permitted, for direct intervention on the controlled process.

The nodes can be connected to the operator's PC in several ways. For instance, a shared bus can be used, with the nodes sending data to the PC when required by the PC. We decided to employ communication by modem, for a robust and flexible way of data transmission. We can use standard telephone wire and connectors, easy to install, or any other means (fiber optic, wireless, etc.), as advisable by distances, interferences, and other circumstances. The nodes can be readily adapted to any communication option, because, internally, the node's hardware is also designed in modular form, with a specific module for the communication port.

For events that require an urgent information to the operator, the system has an interrupt procedure and a special channel, using signals with a frequency that differs from the modem signals (no need to duplicate wires).

Figure 3 shows a diagram of the architecture, and the main blocks of the nodes. Each node comprises an embedded PC, a data acquisition and control unit, and a communication unit.

Fig.3- Architecture of the distributed modular system

5. Generation of the application for the distributed system

The user can build the operator interface he wants, combining windows. For instance, two with analog variables, one with digits, four with buttons. Once satisfied with the display, the operator attaches the visualization elements to the variables measured by the nodes (for instance, one of the analog windows can compare the temperature measured by the node number one, with the pressure measured by the node number two). In a similar mode, buttons can be attached to actions of the nodes. There are dialog windows for these specification tasks, that also configure functions of the affected nodes.

Therefore, during the creation of the operator interface, we specify the role of each node, and how the nodes interact with the operator's PC.

The timing to renew each of the visualized data, can be assigned to the operator's PC, or to the source of information (a node). Data can be saved as files on disk.

Buttons can be attached to digital inputs or outputs, or to announce events (for instance, an alarm), or to control the application execution (start, stop, resume, etc.).

Once the behavior of the distributed system is defined, our development tool can be used to generate the code for the operator's PC.

The code generator is based on the object-oriented design of the system. Each node behaves as an intelligent object, that exchanges messages with the central computer. There is a protocol for the dialog, and a set of possible messages.

The node obeys to a stand-alone program, structured in a hierarchy of classes (object-oriented methodology) with two main levels: the lower for the interface with the process (objects to handle the analog and digital signals, with the A/D and D/A converters and interface electronics), and the higher for the data processing, control and communication
functions. It is easy to adapt the node to new electronic devices (for each of the three main blocks of the node), because the related functions are encapsulated in specific objects.

We have included a learning feature in the nodes, for training-by-doing. Suppose, for instance, there is a rule in a node to open valve A when pressure is over the limit X. In the beginning of the system operation, this rule can only be applied if permitted by the operator's PC. When confirmed by several identical decisions, the node can assume the responsibility of executing the rule in an autonomous manner (that can be overridden by the operator). During the configuration of the system, the user specifies some parameters of the learning process, to make it fast or slow.

As explained, the nodes can take a passive role, only taking data from the sensors and sending information to the central computer as required, and/or take an active role as analog regulators. There is a degree of autonomy of the node, that can be complete in some cases (we have designed a small console, for display and user interface, to be coupled to a node, for such cases). Using our development tool, we generated an application that makes easy the complete configuration of the node, from the operator's PC.

6. Conclusions

The result of our work is an integrated system consisting of a distributed modular architecture, and software tools to automate the developing of the system application, for monitoring, supervision and control purposes. The general objective is to provide a straightforward way of implementing a working system, by means of a simple to set up hardware, and the interactive automated developing of the software.

The nodes of the system are powerful enough, with a direct interface with sensors. The embedded MS-DOS PC paves the way for developing the node control program with conventional environments, such Borland C++. We took advantage of this, to create this program following the object-oriented methodology. The node is seen from the operator's PC as an intelligent object. A set of messages, and an interaction protocol between nodes and the operator's PC, has been defined. This is the key for the initial configuration of the nodes, using our tools, and for the execution of the operational application.

Concerning the software side, a tool that comprises an interactive edition of the user interface, and the automatic generation of code, has been described. Our experience is that the tool is very simple to learn and handle. The tool can be used for developing a complete application, based on our modular architecture, or simply to create an operator interface. In both cases, it generates a source program in C, open to the user, so it can be edited to include more features.

Some examples have been implemented in our laboratory, for the monitoring and control of chemical processes, with good results, both from the point of view of an easy and quick implementation, and from the operational performances obtained.

7. References


[8] TAYLOR Staff, "TAYLOR Industrial Software Reference", TAYLOR Industrial Software, 10045-111 Street, Edmonton, Canada, 1995

