Integration of Embedded Software with Corporate Information Systems *

Ricardo J. Machado, João M. Fernandes
Dept. Sistemas de Informação & Dept. Informática
Universidade do Minho, PORTUGAL

Abstract: This paper describes a methodology and corresponding tools to support the development of information systems, by integrating and interconnecting a network of embedded devices, that supervise processes in industrial environments, with the corporate information system of an organization. We discuss in detail how the LabVIEW environment was customized, so that it effectively supports a component-based and data-flow approach in the development of the gateway responsible for the integration.

Key words: integration of embedded software, design methodology, network and communication systems, system architectures

1. INTRODUCTION

Nowadays, internet makes electronic commerce and electronic business a reality. Therefore, organizations are starting to base their logistic processes in the internet, in an attempt to guarantee a continuous satisfaction to their clients. However, the industrial organizations still feel that the organizational supporting processes do not include their true business, which are the production of goods and services [1].

This new reality forces an important re-structuring on the organizations and gives rise to the need of supporting and controlling the information of their production processes, especially in real-time, anywhere, and at any level of management [2].

* This work has been supported by projects STACOS (POSI/CHS/48875/2002) and METHODES (POSI/CHS/37334/2001)

The integration of real-time shop-floor applications, which we designate industrial control-based information systems (ICIS), with the corporate, or management, information system (MIS) is currently a question of survival for any industrial organization that is dependent on the information and communicant technologies. The typical network topology of the final MIS+ICIS solution is presented in fig. 1, where two distinct "zones" can be identified: (1) the first one corresponds to the shop-floor network supporting the ICIS implementation (in our approach, a CAN network of several embedded devices and one gateway executing LabVIEW software); (2) the second one corresponds to the Ethernet network supporting the MIS implementation by using typical ERP (enterprise resource planning) and POS (plant operations system) software.

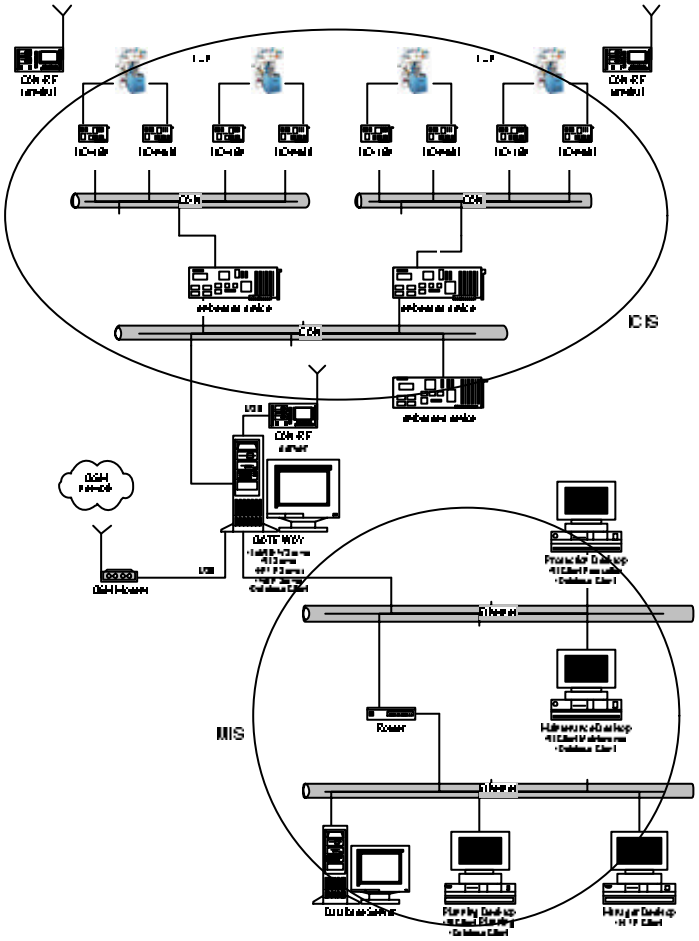

Figure 1 - Network topology for typical MIS+ICIS solutions.

Monitoring and supervision of industrial processes require huge investments in technical solutions based on real-time embedded technologies, especially developed to interconnect the production equipments with the MIS applications. To make CIM (computer integrated manufacturing) [3] an effective reality for an organization, without dependencies from proprietary solutions, we need to fill the gap that exists between the MIS and the ICIS technologies [4].

CIM levels integration have already been handled by the definition of specific industrial communications protocols, namely MAP (manufacturing automation protocol), TOP (technical and office protocol) and MMS (manufacturing message specification). However, the same degree of sophistication was not yet achieved for the applications, which are crucial to feed MIS with truly relevant information flows.

In this context, we have witnessed the relatively low success of the SCADA (supervision, control, automation and data acquisition) applications, due mainly to their low degree of flexibility, whose main examples are WINCC, INTOUCH, and BridgeVIEW. It is important to notice that the CIM levels 0 and 1 are well supported in what concerns the physical control of industrial processes, since industrial equipments usually embed a control system to make their operation more automatic. Thus, the "big" problems are on the systems that monitor and supervise processes and equipments.

Even recent efforts on the definition of standards [5], technologies [6] and methodological frameworks [7] to support the enterprise application integration [8] have failed to support the specific characteristics of industrial information systems. These approaches have not been able to support the design of middleware capable of filling the semantic gap amongst MIS and ICIS.

Under these circumstances, it seems clear that we need technical solutions to easily interconnect the lower CIM levels (0 to 2) with the higher ones (3 and 4). These solutions must necessarily use real-time embedded (and eventually distributed) systems to manage the information flow among the lower CIM levels, and also between these ones and the higher levels [9].

The strategies for the technological integration of MIS and ICIS must also take into consideration that the differences between those two types of information systems can be classified in four groups [10]:

(1) temporal, since a MIS handles time scales that typically are within the days or weeks (in some cases, months or even years are also common), while a ICIS must work with time frames that are measured in seconds or even in milliseconds;
(2) informational, since for a MIS the information is handled under the transactional semantics, while a ICIS is typically described under a event-driven model of computation;

(3) operational, because a MIS directly supports operations of planning and scheduling of the production process and a ICIS supports the control and the supervision of the industrial equipments;
(4) cultural, since a MIS is oriented towards the organizational business, whilst a ICIS focuses on the industrial processes.

2. SOFTWARE ENGINEERING WITH LABVIEW

Allowing non-technical persons to be able to program is becoming not only advantageous but also needed, taking into account the constant changes in requirements and the scarceness of programmers to handle all applications [11]. Therefore, anyone could program a small application according to his own needs and requirements, without the need to follow a software engineering process. This is already a reality for spreadsheets, for example, since anyone (namely those without a computer engineering background) can create his tables and introduce his formulas without knowing the details of the computational engine that is hidden by the graphical interface. This does not happen, in any sense, in the case of embedded systems, with respect to the development activities. However, the use of the LabVIEW tool allows the integrations of physical modules (hardware) and algorithmic modules (VIs - LabVIEW virtual instruments) and increases the abstraction level [12]. This permit the development of components (VIs and virtual models of hardware) with a clear separation between their interface and implementation [13], to directly support the construction of solutions at the system-level and following a component-based design approach.

An important aspect that must be taken into account is to study how adequate is the LabVIEW specification language to the specific working area where the environment is being used. Thus, we need to reinterpret the way applications are to be developed through the definition of the language subset to be used and to produce of a set of guidelines that constitute an architectural reference for developing applications.

One of the main differences that LabVIEW presents with respect to the conventional textual languages is that it follows a data-oriented paradigm (data-flow and data-driven), in the sense that the execution of the programs is controlled by the availability of data and their flow. Thus, the results of the computations (data tokens) are directly transported between instructions, and the data items produced by an instruction are replicated to feed all the instructions that further need them to continue the computational flow. This approach is completely the opposite of the traditional model of computation, known as von Neumann, where the execution of a program is controlled by the sequence of instructions written by the programmer (control-driven).

In LabVIEW, the data-flow approach is mainly reflected on the specification of:

(1) parallelism (concurrency), since it is inherent to the data-flow and is frequently independent of the structural replication of the control units;
(2) reactions of the asynchronous events, since it is often needed to take into consideration the computational context where the event occurs;
(3) hierarchy, since the violation of the hierarchical levels and the termination of activities on computational sub-levels put some difficulties in the gateway design.

We must reinforce the idea that in data-driven programming the data semantics is stronger than in control-driven programming, since there is a direct relation between its availability and an implicit control over the flow of computational execution. In data-driven approach, this execution is typically concurrent and asynchronous in contrast with sequential and synchronous execution of the traditional control-driven programming.

3. GATEWAY DESIGN

To adapt the LabVIEW platform as the development environment for the gateway, responsible for the integration of the MIS and the embedded devices, the tasks next described have been undertaken by the authors.

3.1 Virtual Modeling

The virtual models of the embedded devices must be created in LabVIEW through the implementation of VIs that possess: (1) an interface that contains all the attributes previously formalized through an interface schematics; (2) an implementation that ensures the electronic and run-time access to the corresponding device.

Fig. 2 illustrates an example of the LabVIEW virtual model of an embedded device, as well as the internal attributes that belong to the device access interface. The access to the attributes of an embedded device is accomplished through *unbundle* constructors.

3.2 Access to the Embedded Devices

To assure the electronic and run-time access to the embedded devices in LabVIEW, it is necessary to build a thread in Windows OS, independent of the thread executing the algorithms of the final solution. This new thread aims to multiplex the single physical access to the industrial network (CAN server). The implementation of this CAN server is based on the construction

of a bi-directional queuing system that guarantees the storage and forwarding of the data packets sent from a virtual model to its corresponding embedded device and vice-versa.

This thread is needed, since several embedded devices may be connected to the network, and consequently the corresponding virtual models will be used, potentially in parallel, by the application in LabVIEW.

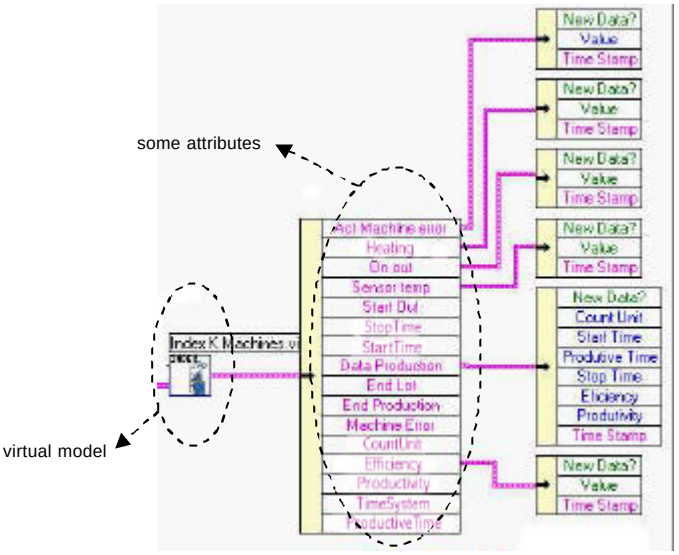

Figure 2 - Virtual model of an embedded device and the "unbundle" of some attributes.

The communication between the LabVIEW application and the embedded devices requires the definition of a communication protocol that implements a layer of services on top of the CAN protocol (VAP) to ease the end-to-end transport of information. This protocol also provides a set of communication services, such as send attributes (*putData( )*), send immediately attributes (*putDataNow( )*), ask attributtes (*getData( )*), ask recent attributes (*getFreshData( )*), parameterize embedded devices, and announce the arrival of new embedded devices (see [14] for more details).

The definition of the VAP protocol followed some of the techniques typically adopted in group communications [15], by using a network of embedded devices that support:

(1) redundancy of embedded devices to permit fault-tolerance;
(2) usage of time stamps in all attributes of the embedded devices;
(3) uniform treatment of a set of embedded devices that handle similar messages (broadcast and multicast communication), both as consumers and producers;

(4) dynamic management of computational resources connected at any time to the network.

In this last topic, it is important to highlight the fact that the connection (or disconnection) of a "new" embedded device is automatically detected by the gateway, behaving as the group server. This implements a truly *plug & play* technology for embedded devices, seen as dynamic components of the distributed system.

3.3 Library of components

To ease on the development of the gateway, a library of components (VIs) was implemented. This library includes several VIs, which are grouped in the following categories:

(1) communication with the CAN network to send and receive data packets;
(2) interaction with structured documents, typically stored in databases, using SQL (structured query language) commands;
(3) sending and reception of SMTP mail messages;
(4) sending and reception of SMS messages to and from entities connected to GSM (global system for mobile communications) networks;
(5) virtual modeling of embedded devices to use in the construction of the gateway software.

3.4 Architecture of the gateway software

The software of the gateway must be developed according to well-defined architectural patterns; such as the *multi-level ICIS* pattern [14]. Apart from that, the gateway software must be structured into the following units:

(1) declaration, where all the embedded and I/O devices that integrate the final solution must be declared (fig. 3);
(2) initialization, where some communication services to be used in the final solution are initialized (fig. 4);
(3) parameterization, where the VIs corresponding to the embedded devices of the final solution are parameterized (fig. 5);
(4) interconnection, where the final solution is developed at the algorithmic point of view (fig. 6).

Some of the interconnection mechanisms between the four aforementioned structural units of the gateway software may be automatically inserted at the very beginning of the development, if the designers use explicitly a LabVIEW wizard developed by the authors to supports this task.

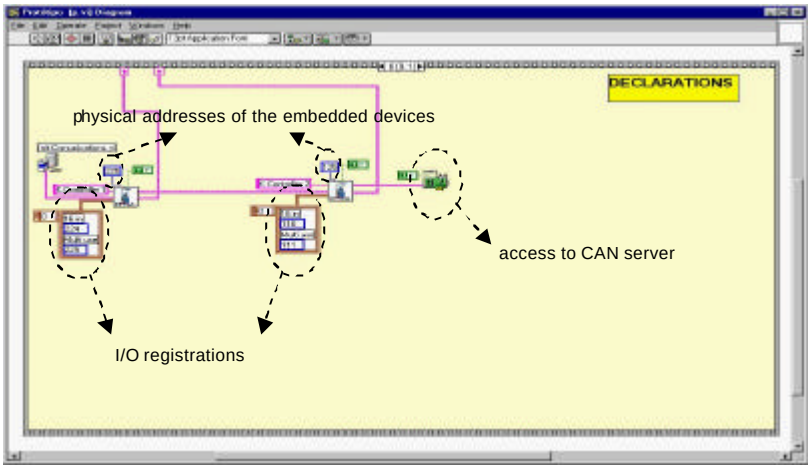

Figure 3 - Declaration of the embedded and I/O devices.

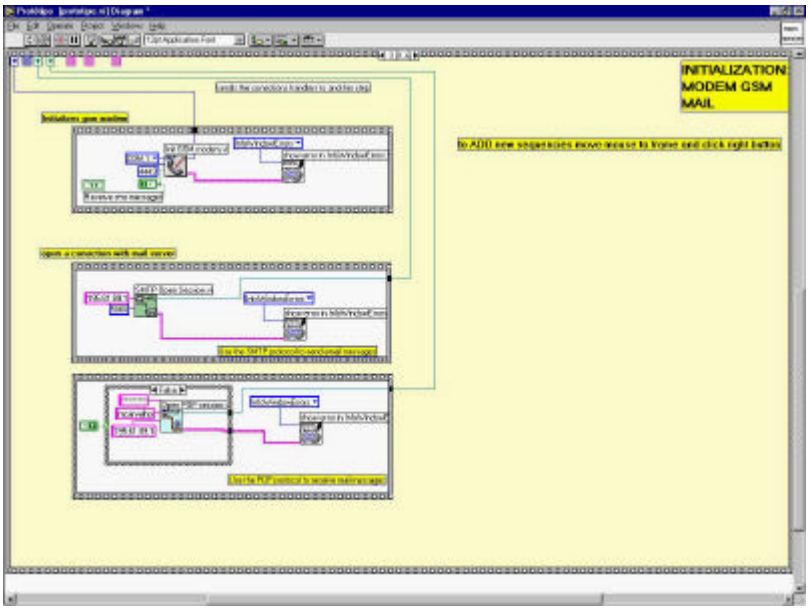

Figure 4 - Initialization of the communication services.

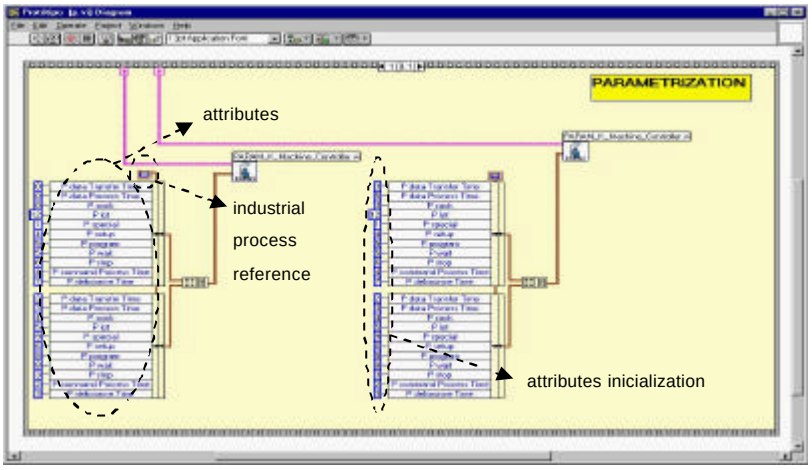

Figure 5 - Parameterization of the virtual models of the embedded devices.

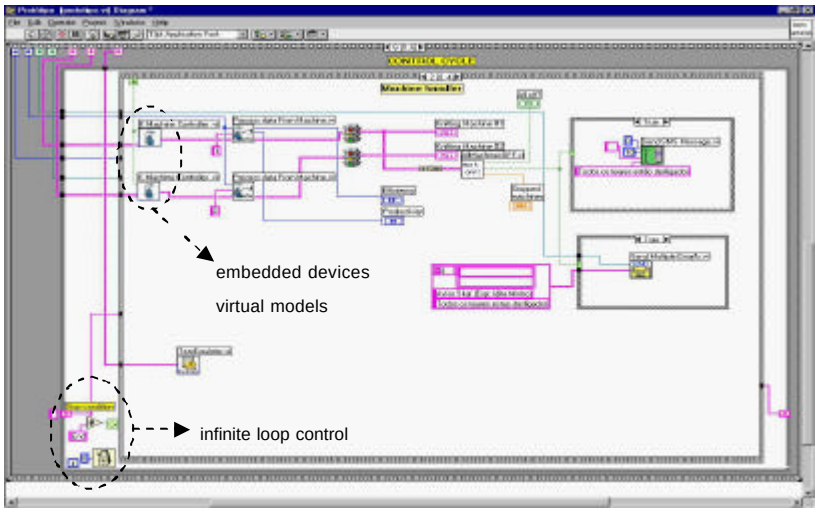

Figure 6 - Interconnection of the MIS with the embedded devices.

4. CONCLUSIONS

The implementation of industrial information systems demands the integration of the industrial networks of embedded devices that supervise the industrial equipments and processes, with the corporate information systems. This integration can be executed at the application level, by developing a

semantic gateway capable of dealing with the temporal, informational, operational and cultural gaps between the ICIS and the MIS "zones" of the industrial information systems.

It is possible to implement the semantic gateway in the LabVIEW environment by extending its platform to support the virtual modeling of the embedded devices, as well as their run-time access. Additionally, the gateway design flow must support the declaration of the embedded and I/O devices, the initialization of the communication services, the parameterization of the virtual models of the embedded devices and the logically interconnection of the MIS with the embedded devices.

5. REFERENCES

[1] A. W. Sheer, Business Process Engineering: Reference Models for Industrial Enterprises, Springer-Verlag, 2nd edition, 1994.

[2] The New Productivity Factor, LIPRO Holding AG, 1999.

[3] J. B. Waldner, CIM: Principles of Computer-Integrated Manufacturing, John Wiley & Sons, 1992.

[4] B. Scholz-Reiter, CIM Interfaces: Concepts, Standards and Problems of Interfaces in Computer Integrated Manufacturing, Chapman & Hall, 1992.

[5] EIA-836 standard for CM Data Exchange and Interoperability, EIA, June, 2002.

[6] R. Zahavi, Enterprise Application Integration with CORBA, John Wiley & Sons, 1999.

[7] UML Profile and Interchange Models for Enterprise Application Integration, OMG, 2001.

[8] D. Linthicum, Next Generation Application Integration – From Simple Information to Web Services, Addison-Wesley, 2002.

[9] R. J. Machado, J. M. Fernandes, Heterogeneous Information Systems Integration: Organizations and Methodologies, PROFES'02, pp. 629-643, M. Oivo, S. K. Sirviö (editors), LNCS 2559, Springer-Verlag, 2002.

[10] R. R. Derynck, T. Hutchinson, Integrating Real-Time Systems with Corporate Information Systems, The Hewlett Packard Journal, vol. 50, no. 1, pp. 26-28, November, 1998.

[11] T. Williams, Object-Oriented Methods Transform Real-Time Programming, Computer Design, pp. 101-118, September, 1992.

[12] J. Jehander, Graphical Object-Oriented Programming in LabVIEW, Application Number no. 143, National Instruments, October, 1999.

[13] R. J. Machado, J. M. Fernandes, A Petri Net Meta-Model to Develop Software Components for Embedded Systems, ACSD'01, pp. 113-122, IEEE CS Press, 2001.

[14] R. J. Machado, J. M. Fernandes, A Multi-level Design Pattern for Embedded Software, DIPES 2004, pp. 247-256, B. Kleinjohann, G. R. Gao, H. Kopetz, L. Kleinjohann, A. Rettberg (editors), Kluwer AP, 2004.

[15] K. P. Birman, The Process Group Approach to Reliable Distributed Computing, Communications of the ACM, vol. 36, pp. 36-53, December, 1993.