Heterogeneous Information Systems Integration: Organizations and Methodologies

Ricardo J. Machado[1] and João M. Fernandes[2]

[1] Dep. Sistemas de Informação, Universidade do Minho, Guimarães, Portugal
[2] Dept. Informática, Universidade do Minho, Braga, Portugal

Abstract. In this paper, a methodology for integrating heterogeneous industrial information systems is presented. The methodology is strongly based on the extensive reuse of already-made components and is conceptually divided in three levels, one for each kind of designer that is typically involved in this type of projects. To accomplish a better integration of the activities and tools necessary to develop industrial information systems with the proposed methodology, three appropriate organizational configurations are adopted.

1 Introduction

The development of industrial information systems is mainly centered in design activities where the re-usage and integration of previously implemented technologies become the main tasks. This reality is imposed by the need to extend the life-cycle of legacy systems, and also as a means to reach final solutions in fewer time and with more robust characteristics. It is important to state that the main goal of an *industrial control-based information system* (ICIS) is the management of the information that flows in the factory plants between the lower and the upper CIM (computer integrated manufacturing) levels [1]. These integration-based design activities are normally executed within an ad-hoc approach, without a strong framework to methodologically support the global process model.

In the context of industrial information systems design, this approach is motivated by the technological diversity of the components, where embedded systems, web-services, and control applications must work together to accomplish the easy interconnection between the lower (0, 1 and 2) and the upper (3 and 4) CIM levels [2]. These solutions are complementary, within the industrial organizations, to the well-known management information systems (MISs) [3]. *Industrial information systems* (IIS), which result from the integration of a MIS with an ICIS, are the answer to accomplish the definition of an applicational platform, based on ERP (enterprise resource planning) approaches, in order to integrate and unify the management and control of all organizational information.

The design and open implementation of this new kind of heterogeneous information systems demand some methodological and architectural issues to be carefully treated

M. Oivo and S. Komi-Sirviö (Eds.): PROFES 2002, LNCS 2559, pp. 629-643, 2002.

[4], which are discussed in this paper. The usage of a UML-based approach to model the organizational configurations that support the execution of IIS integration projects are also presented in this paper. The final goal is to derive an architecture of a tool-set that can aid in the design of the software components that are needed to integrate the IIS solutions. The proposed approach identifies the team structure of an IIS project and, in parallel, defines the software engineering activities that are to be executed by the multiple-organization team members.

This paper does not discuss how helpful is the methodology across organizations, neither does it explicitly report on the benefits of the method compared to others, by presenting a practical assessment of the advantages and weaknesses of the approach, compared with previous ways of working.

This paper intends to justify, in a non-formal way, the methodological steps followed during the execution of an industrial project [5] in which the authors have been involved to apply the approach. In relation to the UML notation, although it is used in some diagrams of this paper, it is not explained here (for details please refer to [6-8]). The UML diagrams that were considered vital for heterogeneous software-based systems design are: use cases, objects, classes, sequence, and statecharts [9].

2 Overview

The presented methodology, designated *virtual automation* (VA), is a complete design and run-time environment, based on software tools, on library modules and on shop-floor software components. In the VA terminology, these components are called *functional modules off-the-shelf* (FMOTSs). VA allows the rapid and easy integration of a distributed network of FMOTSs (that collectively correspond to the ICIS) with the corporate MIS. VA includes the following (in this paper, only the 2 first topics are covered):

- A CAE (Computer-Aided Engineering) software tool.
- A CASE (Computer-Aided Software Engineering) software tool.
- A library composed of several middleware implementation components.
- A FMOTS library composed of several software components.
- A RTOS (Real-Time Operating System).
- A family of embeddable target architectures.
- A run-time execution engine for real-time gateway computation.

Integrating a MIS and an ICIS is a hard task due to the semantical, cultural, temporal, and informational gaps that exist between the development process models that are typically followed for the two subsystems [10]. The main purpose of the VA methodology is to fill the gap between those two subsystems in order to easily and rapidly develop IIS solutions. Additionally, the VA approach guarantees the technological transparency in the virtual modeling of the FMOTSs and also copes with the system's design complexity, by offering a unified development environment that allows the system-level design of the IIS' integrating parts.

The VA methodology is based on co-design principles [11], by promoting the cross fertilization between the hardware and the software domains. Additionally, co-design

allows the semantical unification of the relevant concepts for system-level modeling, the application of data abstraction (object-orientation) to design target architectures and the use of executable specifications to evaluate the system's requirements in its initial developments steps [12]. The research in cross fertilization between both domains has been given excellent results and the work in this field must continue to promote the systems' virtual prototyping (totally in software) and to incorporate the operational approach and the spiral process model into design methodologies.

The hardware/software partitioning and the global scheduling of heterogeneous systems are co-design problems not consensually solved up to now, since they impose the complex conciliation of the synchronization of pseudo-concurrent software with inherently parallel hardware, together with the minimization of communication costs between several partitions [13]. These tricky problems can be even more difficult to tackle if hard real-time systems are considered, with their additional non-functional requirements that enormously constrict the allowable design space exploration.

3 Organizational Configurations

Any engineering project possesses an environment defined by: (1) the *project technical background*, which consists on a scientific and technological framework that defines methodologies, techniques and tools used by the engineer within the execution of the project activities; (2) the *nature of the economical activities*, which defines the application scope of the project deliverables.

3.1 Canonical IT Activities

Taking into account the execution context of an IIS integration project, three canonical classes of organizational IT (*information technology*) activities can be defined:

1. *#A activities (R&D)*. These activities produce generic goods and services for a set of economical activities, but without a binding with a concrete instance of those economical activities. These #A activities are typically R&D activities and are developed by universities, research centers, hi-tech enterprises, and R&D departments of important corporations. An organization that executes these activities possesses ECAD (Electronic Computer-Aided Design) and CASE development tools and produces embeddable target architectures and software elements for compositionally implementing IISs.
2. *#B activities (integration)*. These activities use the generic goods and services (furnished by #A activities) within a concrete instance of economical activities, by adapting and customizing the reusable technological components. These #B activities typically integrate technological solutions, and are developed by engineering companies and engineering departments of important corporations. An organization that executes these activities detains embeddable target architectures and software elements (produced by type #A activities) and produces IIS final solutions for supporting the supervision and monitoring of production, maintenance and quality indexes of industrial processes.

3. *#C activities (production)*. These activities correspond to the economical activities that constitute the target problem domain.

For implementing ICISs with the VA approach, there are several professional profiles that do contribute for the accomplishment of the 3 types of activities, previously indicated:

- the hardware engineer, which conceives embeddable target architectures at level 1 of the VA methodology (see sec. 4), directly contributing to execute type #A activities;
- the software engineer, which conceives FMOTSs at level 2 of the VA methodology (see sec. 4), directly contributing to execute type #A activities;
- the systems engineer, which constructs final solutions at level 3 of the VA methodology (see sec. 4), directly contributing to execute type #B activities;
- the maintenance technician, which technically supports the systems engineer with respect to the processes and equipments used for type #C activities and assures that all the procedures for installing and maintaining the final solution are done;
- the manager that is responsible for type #C activities.

3.2 Configurations

Based on the 3 types of activities and the 5 profiles involved with them, the authors obtained 5 organizational configurations that serve as a reference for the same number of environments where ICISs can be developed.

In configuration I (fig. 1a), there is a different organization for each type of activity: (i) an organization (of type #A) that just executes type #A activities, by recurring to hardware and software engineers; (ii) an organization (of type #B) that just executes type #B activities, by recurring to systems engineers; (iii) an organization (of type #C) that just executes type #C activities, by recurring to maintenance technicians and managers. This configuration requires the organizations to have a high degree of specialization, since each one only supports one type of activity.

Configuration II (fig. 1b) differs from configuration I by the fact that the maintenance technician is included in the organization of type #B instead of type #C. This configuration requires the maintenance technicians (and also the systems engineers) to be sufficiently able to adapt themselves to the specific details of all the type #C organizations where they develop ICIS projects.

In configuration III (fig. 1c), there are only 2 different organizations involved in the project: (i) an organization (of type #A) that just executes type #A activities, by recurring to hardware and software engineers; (ii) an organization (of type #B + #C) that executes types #B and #C activities, by recurring to systems engineers, maintenance technicians and managers. This configuration covers the situation where a big corporation (organization of type #B + #C) is capable of promoting the conception of final solutions.

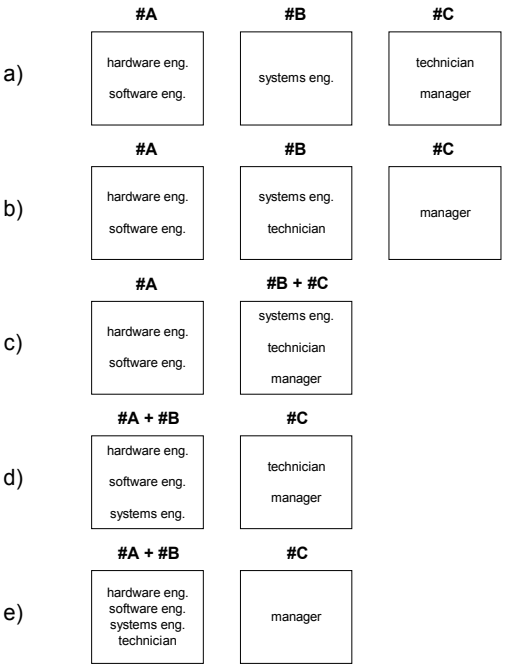

Fig. 1. Organizational configurations

In configuration IV (fig. 1d), there are only 2 different organizations involved in the project: (i) an organization (of type #A + #B) that executes types #A and #B activities, by recurring to hardware, software, and systems engineers; (ii) an organization (of type #C) that just executes types #C activities, by recurring to maintenance technicians and managers. This configuration concentrates, on a single corporation (organization of type #A + #B), all the R&D effort and the way technology is transferred to the industrial parties capable of promoting the final solutions conception.

Configuration V (fig. 1e) differs from configuration IV by the fact that the maintenance technician is included in the organization of type #A + #B instead of type #C.

It is possible to obtain more organizational configurations than those discussed above. However, there is no real interest, for example, to concentrate the 3 types of activities on a single organization or to separate in distinct organizations the hardware and the software engineers, since the VA methodology has two main goals:

- The first one is to free the organizations that execute type #A activities from: (i) having a deep knowledge of all the application areas where the technologies developed by them can be used; (ii) developing final solutions; and (iii) installing and maintaining final solutions. Thus, organizations that only execute type #A activities can have less dispersion on their work, which implies more time and attention to research and to develop industrial prototypes and integratable components. The desire of only assigning type #A activities to these organizations is related to the need of having final products with higher quality standards, i.e., more versatile, robust and effective FMOTSs.

- The second aim is to promote the appearance of organizations only responsible for type #B activities by: (i) supplying a CAE tool that transparently supports the construction of the final solutions; and (ii) making available a store of FMOTSs ready to be parameterized and installed. Thus, these organizations will not have to pursue R&D activities, which implies that they can concentrate on the integration of final solutions.

There is a third class of organizations, named #C, that corresponds to the industrial organization that receives the designed ICIS to install in its shop-floor.

With these two aims, the VA methodology addresses the first 3 configurations (I, II and III) instead of the last 2 ones (IV and V). The preferred configurations assure that the organizations have a specific interface with their counterparts: organizations of type #A only develop FMOTSs, and organizations of type #B construct final solutions that satisfy the needs of type #C activities, by using FMOTSs (they may also be responsible for installing and maintaining the final solutions).

Contrarily, configurations IV and V were considered undesirable, since they require organizations of type #A activities to additionally develop final solutions and, even, to install and maintain them.

From a methodological point of view, this analysis to the organizational configurations justifies the need to formalize the profiles of the 2 professionals (software and systems engineers) that are always present on different organizations. By working on different organizations, it is mandatory to address the issues associated to the design, the reuse and the composition of components (FMOTSs). It is also obligatory to take into account that the components must be fully documented in what concerns their functionalities and the way they can be interconnected.

Within the VA approach, the systems engineer should: (1) know the FMOTSs behavioral interface; (2) know the final solution's requirements; and (3) be able to parametrize and interconnect the selected FMOTSs to fully accomplish the requirements of the final solution. The software engineer should: (1) know the target architecture designed by the hardware engineer; (2) know the algorithmic requirements to implement over the target architecture; and (3) be able to develop generic components (FMOTSs).

4 The VA Approach

This section discusses how to obtain a design environment for system-level integration of real-time embedded FMOTSs capable of implementing ICISs. Within that environment, models should be iteratively reified until the system is implemented, without the need for manual macro-refinements, with the transparent reuse of embeddable target architectures and software modules, and supporting, throughout the design process, the activities of the three professionals typically involved (hardware, software and systems engineers).

To accomplish this objective, it is necessary to decouple the traditional top-down one-all-going project approach into three feed-forward quasi-independent project levels, each one with a different design flow, but organized by a common middle-out macro-process design flow (fig. 2):

- *Hardware level* (level 1 of the VA methodology), where the embeddable target architectures are provided to computationally support the parameterisable FMOTSs [14]. It is possible to make use of reconfigurable technologies to implement, directly in the hardware, algorithmic primitives that should be transparently used in level 2. The introduction of these reconfigurable technologies promotes the execution of typical co-design tasks, like the pre-partitioning tasks.
- *Software level* (level 2 of the VA methodology), where the parameterisable programs to run on the FMOTSs are constructed with a CASE tool [15]. At this level, the software engineer must decide which functionalities will run directly on the processor and which ones will be synthesized for the reconfigurable devices.
- *Information system level* (level 3 of the VA methodology), where the final heterogeneous IIS solution is designed with a CAE tool, by integrating the previously designed FMOTSs and the existing MIS [7]. The systems engineer can decide whether to use the algorithms embedded in the FMOTSs or to conceive new ones (or complement the existing ones) to run on the real-time gateway (the component responsible for stubbing the interconnection of the ICIS with the MIS).

This decoupling must assure that it can be possible to establish, with the three kind of engineering professionals, a co-design community within the same project, each one with the responsibility of implementing the control primitives corresponding to his capabilities and duties. This approach can be better explained by reinterpreting the 5 T's analysis [16].

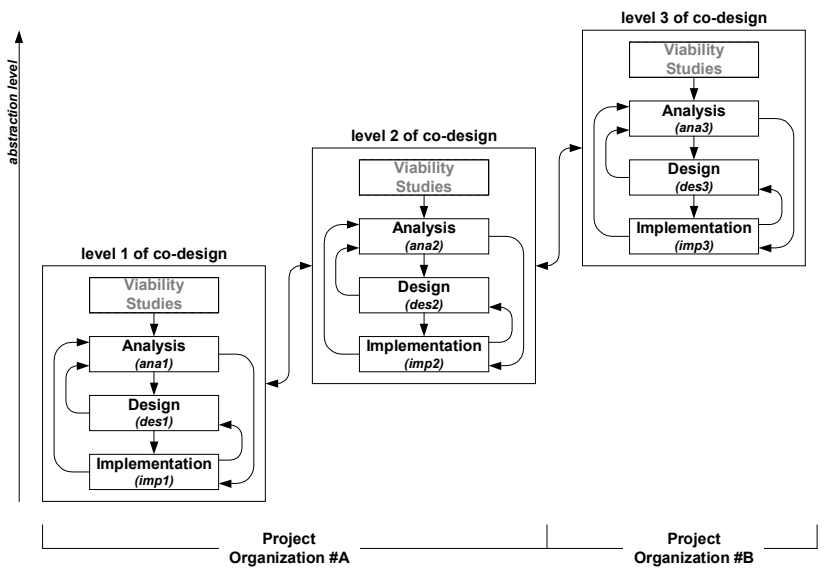

Fig. 2. Macro-process of the 3-level methodology

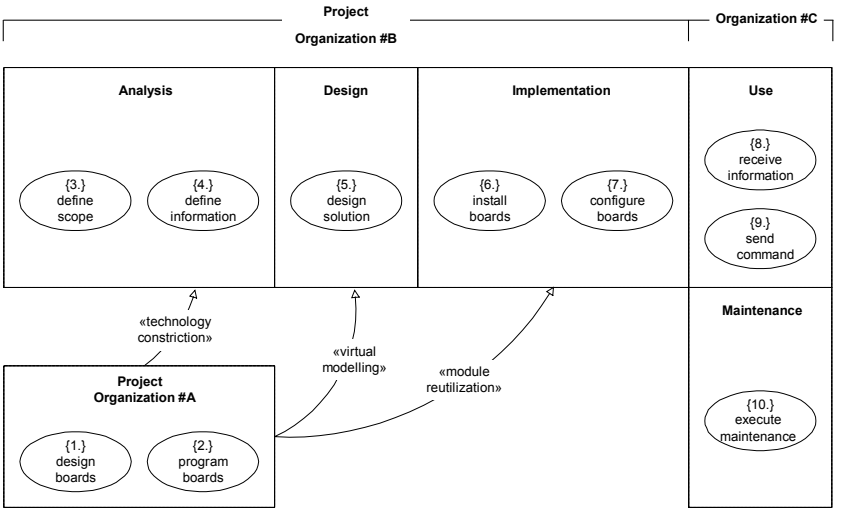

Fig. 3. Final solutions life-cycle diagram

Timelines. The time to market pressure, together with the usual requirements modifications, suggests that methodologies try to address the reduction of development times and promote the use of rapid implementation technologies, such as COTS (components off-the-shelf) [17, 18]. To make this feasible in the use of reconfigurable processing target architectures, functional modules (built with target architectures loaded with parameterisable software) that transparently implement low-level control primitives must be defined. These modules offer, to the level 3, the possibility to reuse reconfigurable technology. These modules correspond to the previously called FMOTSs.

Tasks. The methodologies in use today suffer from poor systematisation in what concerns the several design tasks that must be executed. This inconsistent design process approach does not promote a correct integrated design of the system's parts and does not support the effective reuse of components, available from previous projects. This reality justifies the need to carefully integrate co-design within its three levels, by defining differential tasks for target architectures design (hardware engineers), for FMOTSs design (software engineers) and for final solutions design (systems engineers).

Tools. There is an extreme necessity of promoting the integration of tools to permit systems engineers to achieve an effective (semi-)automatic design at system-level. This demand is only feasible if the two other kinds of engineering professionals have access to design tools capable of supporting the intra-communication design flow levels, in what concerns the semantical manipulation of unified representations and the automatic code generation.

Technology. Taking into account the Moore's Law, custom solutions are only interesting in a narrower time-window. In this context, it is advantageous to adopt methodologies capable of supporting technologies that allow the periodic update of compo-

nents (model year upgrade) for performance increase, but assuring, at least, the same functionality. This model year upgrade of components benefits from the co-design level decoupling, since each engineering professional is only concerned with the update within its design level, but contributing for the global updating of the final solution.

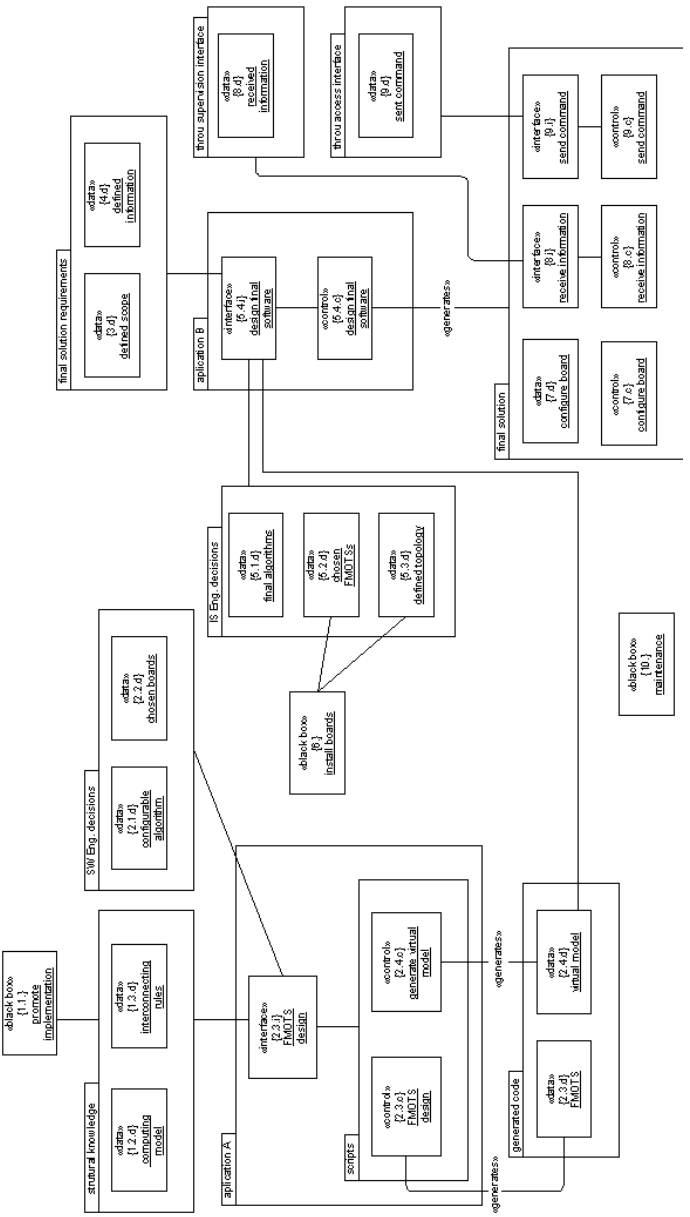

Fig. 4. Process-level object diagram of the 3-level approach

Talent. Besides all the R&D work that this co-design approach demands, the training of the three engineering professionals in this new way of executing and designing with heterogeneous implementations must not be ignored.

Fig. 2 illustrates two kinds of projects, each one executed within a different organization. At organization #A, ready-to-use FMOTSs are delivered, which support levels 1 (hardware engineers) and 2 (software engineers) of the VA methodology; and at organization #B, FMOTSs to deliver an ICIS final solution are parameterized, which support level 3 (systems engineers) of the VA methodology.

5 The 3-Level Process Model

The execution of a process-level (not product-level) requirements capturing task of the 3-level approach results in a UML use case diagram (not shown in this paper) that represents the duties and responsibilities of each engineering professional. From this diagram one can obtain the final solutions life-cycle diagram, shown in fig. 3.

In this diagram, there are three new categories for the UML «relationship» stereotype:

1. «*technology constriction*». This stereotype restricts the domain of use cases *3. define scope* and *4. define information* within organization #B to the economical scope of its supplier (organization #A). This restriction is justified by the fact that each organization #A possesses a limited set of target architectures and supplies only a restricted set of FMOTSs with well-defined functionalities.
2. «*virtual modeling*». This stereotype imposes that in the execution of use case *5. design solution*, it is necessary to manipulate virtual models of the selected FMOTSs (supplied by one organization #A and chosen to be components of the final solution). This virtual modeling is only concerned with the required characteristics to allow the interconnection and parameterization, in the scope of the final solution, of the chosen FMOTSs. This technological transparency should guarantee a good complexity control level in the design of final solutions.
3. «*module reutilization*». This stereotype indicates that use cases *6. install boards* and *7. configure boards* reuse previously designed FMOTSs (by the organization #A). This reutilization avoids the design of specific and narrow application solutions, although it guarantees a satisfactory customization level of the functional and non-functional demands of the final solution.

These three stereotypes were defined to formally separate the level 2 from the level 3 of the VA methodology and, thus, to correctly characterize the design activities of the software engineers and the systems engineers.

Following the 4-set rule set presented in [7], a process-level object diagram (fig. 4) is obtained from the process-level use case diagram. This object diagram represents the requirements of the design tools that should support the 3-level approach. In this diagram, there are two application packages (A and B).

Application A package supports the level 2 of the VA methodology by making available:

4. one environment with an user interface (*2.3.i FMOTS design*) capable of assisting the design tasks of software engineers, who possess the structural knowledge of the target architectures (*1.2.d computing model* and *1.3.d interconnecting rules*) and who take some previous decisions (*2.1.d configurable algorithm* and *2.2.d chosen boards*);
5. one engine capable of (semi-)automate both the design of FMOTSs (*2.3.c FMOTS design*) and the generation of final code (*2.3.d FMOTS*) for the system synthesis and implementation in the chosen target architecture;
6. one engine capable of (semi-)automate the generation of virtual models (*2.4.c generate virtual model*) to allow, in the application B package, the configuration, in a technological transparent way, of the previously designed FMOTSs (in application A package).

Application B package supports the level 3 of the VA methodology by making available:

1. one environment with an user interface (*5.4.i design final software*) capable of assisting the design tasks of information systems engineers, who possess the knowledge of the final solutions requirements (*3.d defined scope* and *4.d defined information*) and who take some previous decisions (*5.1.d final algorithms*, *5.2.d chosen FMOTSs* and *5.3.d defined topology*);
2. one engine capable of (semi-)automate the generation of final solutions (*5.4.c design final software*).

For implementing the application A package, a Java-based tool is being used and for the application B package, the LabVIEW CAE tool is being adopted (fig. 5) [5]. In LabVIEW, the technological transparency in the remote invocation of the shop-floor embedded components is accomplished by the use of VIs (*virtual instruments*) that represent their behavioral interface for pre-run-time parameterization and for establishing the run-time interconnection with their distributed implementations.

Note that the *final solution* package corresponds to the ICIS final solution to be installed in the organization #C shop-floor. This final solution executes the FMOTSs configuration (*7.c configure board*) and then executes the supervision and monitoring algorithms included in the FMOTSs (*8.c receive information* and *9.c send command*). LabVIEW is also the central element of the final solution that gathers and distributes the information from all the processing elements; i.e., LabVIEW performs at run-time the role of a semantical gateway between the shop-floor elements and the corporate MIS.

Fig. 6 depicts the global 3-level environment with the cascaded ECAD, CASE and CAE tools supporting the three engineering designers in the implementation of heterogeneous IIS final solutions.

6 Conclusions

This paper has presented a 3-level methodology, especially tuned to apply the basic co-design principles to the open component-based design of parameterisable modules,

capable of supporting the integration of heterogeneous industrial information systems. The methodology is intended to help various IT practitioners to design systems that can be reused and adapted by others. The paper starts by identifying three classes of activities for building an IIS:

- The R&D activities that produce generic and embeddable hardware/software elements.
- The integration activities that adapt the previous software to a specific application/system.
- The production activity that makes sure that the application/system runs in the target environment.

To achieve a comfortable integration of the activities and tools necessary to develop industrial information systems with the proposed methodology, the authors have selected three organizational configurations that are considered appropriate to support the global methodology. The activities can be split over three different organizations or be tackled within the same organization. The paper describes a methodology to support the design and development to be integrated softly among these three groups of practitioners. The systematic analysis and metric assessment of other comparable methodological approaches were not considered to be presented in this paper.

Within this context, from the process-level use case diagram (final solution life-cycle diagram), a process-level object diagram was obtained, which specifies the requirements of the design tools supporting the 3-level approach.

Based on these requirements, a set of design tools has been assembled to support the VA methodology. Both, the methodology and tools, have been already used in real projects for developing industrial information systems.

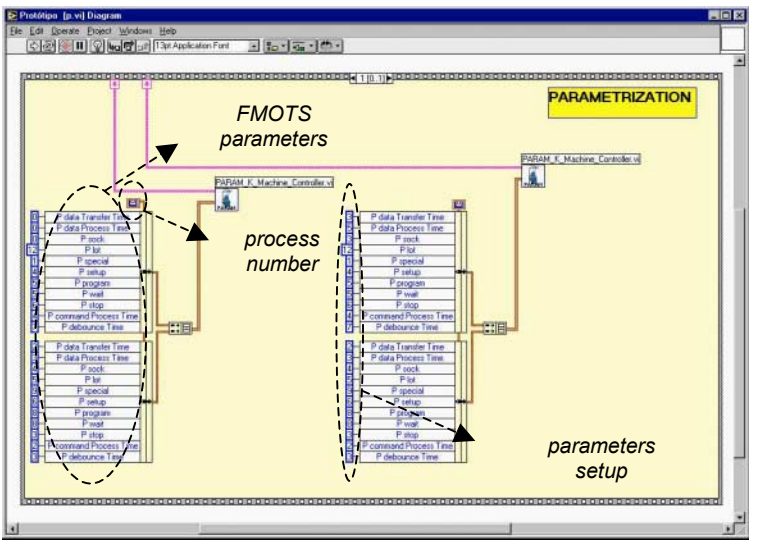

Fig. 5. The LabVIEW CAE tool

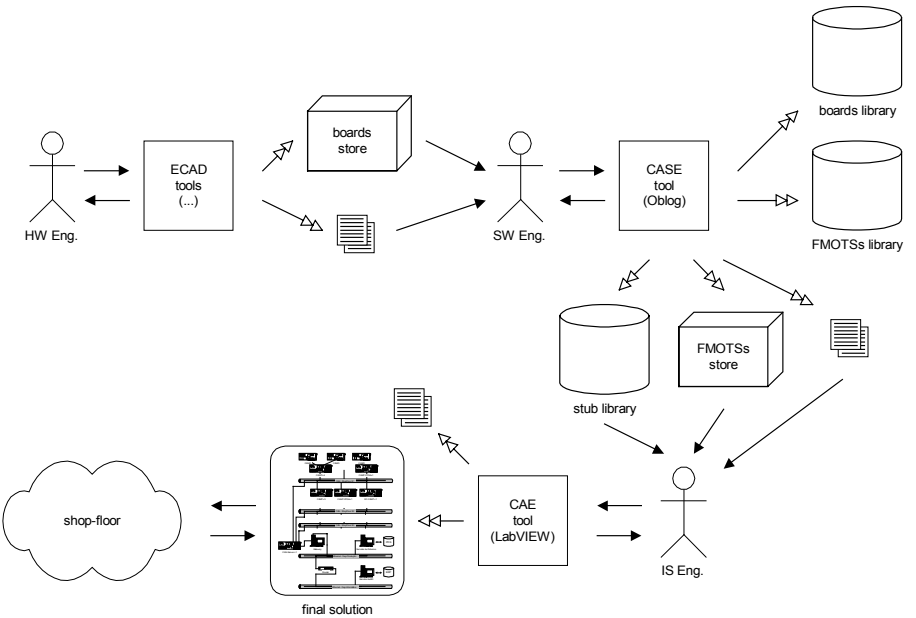

Fig. 6. Global 3-level environment

The LabVIEW environment proved to be a powerful and mature CASE tool, by providing the essential mechanisms to benefit from the component-based design approach in the reuse of components, the rapid-prototyping in the user's requirements validation, and high-quality user interfaces easily programmed for the final solutions. Additionally, the environment supports the system life-cycle evolution, requirements modification and software maintenance, which greatly contributes to deal with the development of heterogeneous information systems.

Acknowledgments

We gratefully acknowledge financial support from Fundação para a Ciência e a Tecnologia (FCT) and Fundo Europeu de Desenvolvimento Regional (FEDER) under project „METHODES: Methodologies and Tools for Developing Complex Real-Time Embedded Systems“ (POSI/37334/CHS/2001).

References

1. Waldner, J.-B. CIM: Principles of Computer-Integrated Manufacturing. John Wiley & Sons, 1992.
2. Ranky, P. G. Computer Networks for World Class CIM Systems. CIMware Limited, 1990.

3. Scholz-Reiter, B. CIM Interfaces: Concepts, Standards and Problems of Interfaces in Computer Integrated Manufacturing. Chapman & Hall, 1992.
4. Eckstein, S., P. Ahlbrecht, K. Neumann. Increasing Reusability in Information Systems Development by Applying Generic Methods. In *13th Conference on Advanced Information Systems Engineering (CAISE'01)*, pages 251-266, Interlaken, Switzerland, LNCS 2068.
5. Machado, R. J., J. M. Fernandes, A. F. Silva. LabVIEW as a CASE Environment for the Integration of Distributed Shop-Floor Embedded Components with Corporate Information Systems. In *National Instruments Conference on Measurement and Automation (NIWeek'01)*, Academic Session, Austin, TX, USA, August 2001.
6. Fernandes, J. M., R. J. Machado, H. D. Santos. Modeling Industrial Embedded Systems with UML. In *8th ACM/IEEE/IFIP Int. Workshop on Hardware/Software Codesign (CODES 2000)*, pages 18-22, San Diego, CA, USA, ACM Press, May 2000.
7. Fernandes, J. M., R. J. Machado. From Use Cases to Objects: An Industrial Information Systems Case Study Analysis. In *7th International Conference on Object-Oriented Information Systems (OOIS'01)*, pages 319-328, Calgary, Canada, Springer-Verlag, August 2001.
8. Fernandes, J. M., R. J. Machado. System-Level Object-Orientation in the Specification and Validation of Embedded Systems. In *14th Symposium on Integrated Circuits and System Design (SBCCI'01)*, pages 8-13, Pirenópolis, Brazil, IEEE Computer Society Press, September 2001.
9. Köhler, H. J., U. Nickel, J. Niere, A. Zündorf. Integrating UML Diagrams for Production Control Systems. In *22nd International Conference on Software Engineering (ICSE 2000)*, pages 241-251, Limerick, Ireland, ACM Press, June 2000.
10. Derynck, R. R., T. Hutchinson. Integrating Real-Time Systems with Corporate Information Systems, *The Hewlett-Packard Journal*, 50(1):26--28, 1998.
11. Rozenblit, J., K. Buchenrieder. Codesign: Computer-Aided Software/Hardware Engineering, IEEE Press, 1995.
12. Machado, R. J., J. M. Fernandes, H. D. Santos. A Methodology for Complex Embedded Systems Design: Petri Nets within a UML Approach. In B. Kleinjohann, editor, *Architecture and Design of Distributed Embedded Systems*, pages 1-10, Kluwer Academic Publishers, 2001.
13. Yen, T.-Y., W. Wolf. Hardware-Software Co-Synthesis of Distributed Embedded Systems. Kluwer Academic Publishers, 1996.
14. Machado, R. J., J. M. Fernandes, A. J. Esteves, H. D. Santos. An Evolutionary Approach to the Use of Petri Net Based Models: From Parallel Controllers to HW/SW Co-Design. In A. Yakovlev, L. Gomes, and L. Lavagno, editors, *Hardware Design and Petri Nets*, pages 205-222, Kluwer Academic Publishers, 2000.
15. Machado, R. J., J. M. Fernandes. A Petri Net Meta-Model to Develop Software Components for Embedded Systems. In *2nd IEEE International Conference on Application of Concurrency to System Design (ACSD'01)*, pages 113-122, Newcastle Upon Tyne, U.K., IEEE Computer Society Press, June 2001.

16. Madisetti, V. K., M. A. Richards. Advances in Rapid Prototyping of Digital Systems. *IEEE Design & Test of Computers*, 13(3):9-11, 1996.
17. Voas, J. The Challenges of Using COTS Software in Component-Based Development. *IEEE Computer*, 31(6):44-45, 1998.
18. Wang, Y., S. Patel, D. Patel. On Built-in Test Classes for Object-Oriented and Component-Based Information Systems In *7th International Conference on Object-Oriented Information Systems (OOIS'01)*, pages 307-316, Calgary, Canada, Springer-Verlag, August 2001.