AN OBJECT-ORIENTED APPROACH TO THE CO-DESIGN OF INDUSTRIAL CONTROL-BASED INFORMATION SYSTEMS*

Ricardo J. Machado[❖], João M. Fernandes[*], Henrique D. Santos[❖]

[❖]Departamento de Sistemas de Informação
[*]Departamento de Informática
Escola de Engenharia – Universidade do Minho
PORTUGAL

Abstract: This paper defines a system-level hardware/software co-design approach to the development of real-time applications, which allows the implementation of industrial control-based information systems. The relevance of the co-design-based development presented resides in the fact that with the three-level project decoupling proposed it is easier to benefit from the speed-up that results from the use of reconfigurable processing architectures in the implementation of critical and real-time requirements. This methodology defends the use of homogeneous, neutral-object-oriented and executable specifications, the adoption of the operational approach and the spiral process model to support an architectural-pattern-based inward-level microprocess design flow. *Copyright © 2000 IFAC*

Keywords: Systems engineering, System-level co-design, Object modelling techniques, Reconfigurable Processing Architectures.

1. INTRODUCTION

Control, monitoring and supervision of industrial processes are increasingly demanding a great investment in technological solutions each time more embedded and with real-time capabilities, especially devoted to the interconnect, in an intelligent way, of shop-floor equipment with operational information systems (for production, quality and maintenance management). The main goal of *industrial control-based information systems* (ICISs) is the management of the information that flows in the factory plants between the lower and the upper CIM (*computer integrated manufacturing*) [Waldner 1992] levels (fig. 1).

In the context of fig. 1, it seems evident the need for technological solutions for the easy interconnect of upper (0, 1 and 2) and lower (3 and 4) CIM levels [Ranky 1990]. These solutions must, certainly, use embedded, real-time, and eventually distributed, *computer-based systems* (CBSs) to computationally support the implementation of ICISs, truly complementary, within the industrial organisations, to the well known *management information systems* (MISs) [Scholz-Reiter 1992]. The resulting set MIS + ICIS is the answer to the promising ERP (*enterprise resource planning*) approaches to accomplish the definition of an applicational platform that can integrate and unify the management and control of all the organisational information [Lipro 1999].

The design and open implementation of this new kind of information systems demand some methodological and architectural issues to be carefully treated, which are discussed in this communication.

* This work has been partially funded by the Portuguese Science & Technology Foundation project PRAXIS/P/EEI/10155/1998, *Reconfigurable Embedded Systems: Development Methodologies for Real-Time Applications.*

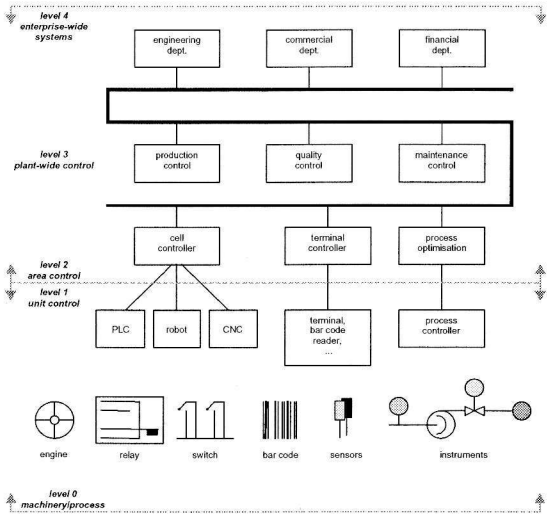

Fig. 1: CIM levels in an industrial organisation.

2. THE CO-DESIGN APPROACH

The hardware/software co-design promotes the cross fertilisation between the hardware and the software domains, allowing the semantical unification of the relevant concepts for system-level modelling, the application of data abstraction (object-orientation) to digital hardware design and the use of executable specifications to evaluate the system's requirements in its initial developments steps [Machado *et al.* 2000]. The research in cross fertilisation between both domains gave 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 consensuously solved up to now, since they impose the complex conciliation of the synchronisation of pseudo-concurrent software with inherently parallel hardware, together with the minimisation of communications cost between technological diverse partitions [Yen *et al* 1996]. 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. RECONFIGURABLE ARCHITECTURES

With the advent of hardware structures capable of dynamic reconfiguration (during execution time), those co-design problems became extremely complex, due to the fact that this new hardware dimension eliminates almost completely the traditional functional differences between hardware and software. With ISP (***in-system programming***) technology the FPGA (***field-programmable gate arrays***) components gained competitive advantages in the implementation of reconfigurable computing systems, instead of their typical role in the temporary and precarious replacement of the expensive MPGAs (***mask-programmable gate arrays***) [Hutchings *et al.* 1995].

The list of reconfigurable systems based on ISP FPGAs (*FCCMs – **F**PGA based **c**ustom **c**omputing **m**achines*) is increasingly growing [Guccione_list], although some architectural design trade-offs are still under study [Esteves *et al.* 1997], namely, the granularity of the reconfigurable hardware structures and the topological proximity between the CPU and the reconfigurable hardware resources. These two problems are the origin of the actual difficulty to define an overall reconfigurable computing paradigm capable of supporting the design of CBSs using FCCM-based target architectures, without the need to skillfully optimise the hardware synthesis [Buell *et al.* 1996].

In this context, the success of the reconfigurable computing paradigm demands the execution of some methodological innovations for designing real-time embedded CBSs, assuring the models' continuity throughout the reification phases and addressing the problem of controlling the complexity in the design [Fernandes *et al.* 1999]. Since real-time embedded CBSs have progressively been developed within the hardware/software co-design approach, those innovations can be reached by integrating ECAD (***e**lectronic **c**omputer **a**ided **d**esign*) and CASE (***c**omputer **a**ided **s**oftware **e**ngineering*) tools so that an effective EDA (***e**lectronic **d**esign **a**utomation*) methodology, capable of supporting the virtual prototyping and the system-level design [Mangione-Smith *et al.* 1997], can be obtained.

4. A THREE-LEVEL CO-DESIGN APPROACH

Taking into account the previous considerations about hardware/software co-design and about reconfigurable processing target architectures, the key point that this paper wants to discuss consists on how to obtain a design environment for system-level co-design of real-time embedded CBSs capable of implementing ICISs. Within that environment, models should be iteratively reified untill system is implemented, without the need for manual macro-refinements, with the transparent reuse of hardware and software modules, and supporting, throughout the design process, the activities of the three typical professionals involved (hardware engineers, software engineers and systems engineers).

In the presence of this problem, the authors defend the co-design "democratisation", by guaranteeing that all three engineering professionals have effective

access to it and by refuting to relegate the co-design only to those who possess strong motivations for hardware synthesis. To accomplish this claim, 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 organised by a common middle-out macroprocess design flow (fig. 2): (1) at level 1, hardware engineers build target architectures which can have reconfigurable components; (2) at level 2, software engineers design functional modules; (3) at level 3, systems engineers (typically, information systems engineers) construct final solutions (ICISs).

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 using the 5 T's analysis [Madisetti *et al.* 1996]:

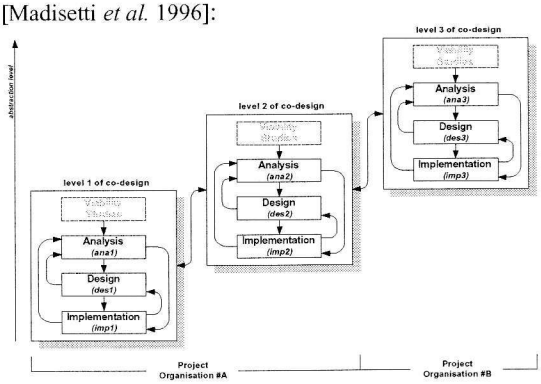

Fig. 2: Macroprocess of the three-level co-design approach.

(1) *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***) [Voas 1998]. To make this feasible in the use of reconfigurable processing target architectures, functional modules (built with target architectures loaded with configurable software) that transparently implement low-level control primitives must be defined. These modules offer, to the level 3 of co-design, the possibility to reuse reconfigurable technology. Authors call these modules FMOTSs (***functional-modules-off-the-shelf***).

(2) *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 hardware and software and does not support the effective reuse of components, available from previous projects. This reality justifies the need to carefully frame 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 (information systems engineers).

(3) *Tools*. There is an extreme necessity of promoting the integration of tools to allow information systems engineers achieve an effective (semi-)automatic design at system-level. This demand is only feasible if the other two kinds of engineering professionals have access to design tools capable of supporting the communication intra design flow levels, in what concerns the semantical manipulation of unified representations and the automatic generation of code [Machado *et al.* 1997].

(4) *Technology*. Taking into account the Moore's Law, custom solutions are only interesting in a decreasing width time-window. In this context, it is advantageous to adopt methodologies capable of supporting technologies that allow the periodic update of components (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 concern with the update within its co-design level, but contributing for the global updating of the final solution.

(5) *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 (hardware and software) must not be ignored.

Fig. 2 illustrates two kinds of projects, each one executed within a different organisation: (1) organisation #A, that supports levels 1 (hardware engineers) and 2 (software engineers) of co-design, by delivering ready-to-use FMOTSs; (2) organisation #B, that supports level 3 (information systems engineers) of co-design, by configuring FMOTSs to deliver an ICIS final solution. There is a third class of organisations: organisation #C, that corresponds to the industrial organisation that receives the designed ICIS to install in its shop-floor.

5. THE THREE-LEVEL CO-DESIGN PROCESS MODEL

The execution of a project process-level (note that it is not product-level) requirements capturing task of the three-level co-design approach results in a UML use case diagram (not shown in this paper) that represents the duties and responsibilities of each engineering professional. This diagram allows the obtention of the final solutions life-cycle diagram, shown in fig. 3.

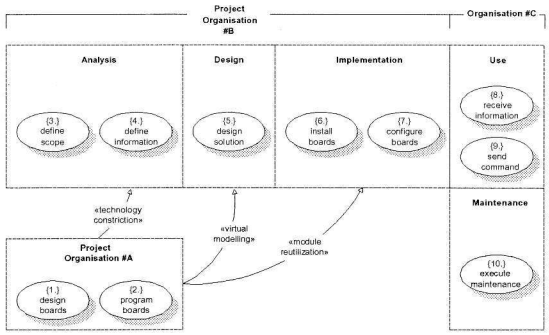

Fig. 3: Final solutions life-cycle diagram.

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

(1) *«technology constriction»*. This stereotype restricts the acting domain of both the *3. define scope* and the *4. define information* use cases within one organisation #B to the economical scope of its supplier (organisation #A). This delimitation is justified by the fact that each organisation #A possesses a limited set of target architectures and supplies only a restricted set of FMOTSs with perfectly well defined functionalities.

(2) *«virtual modelling»*. This stereotype imposes that in the execution of the *5. design solution* use case, it is necessary to manipulate virtual models of the selected FMOTSs (supplied by one organisation #A and chosen to be components of the final solution). This virtual modelling is only concerned with the required characteristics to allow the interconnection and configuration, 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 both the *6. install boards* and the *7. configure boards* use cases reuse previously designed FMOTSs (by the organisation #A). This reutilization assurance avoids the design of specific and narrow application solutions, although it guarantees one enough customisation level to satisfy the functional and non-functional demands of the final solution.

These three stereotypes have been defined to formally separate the level 2 from the level 3 of co-design and, thus, to perfectly characterise the design activities of the software engineers and the information systems engineers.

Following the 4-set rule set presented in [Fernandes *et al.* 2000], 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 three-level co-design approach. In this diagram, there are two application packages:

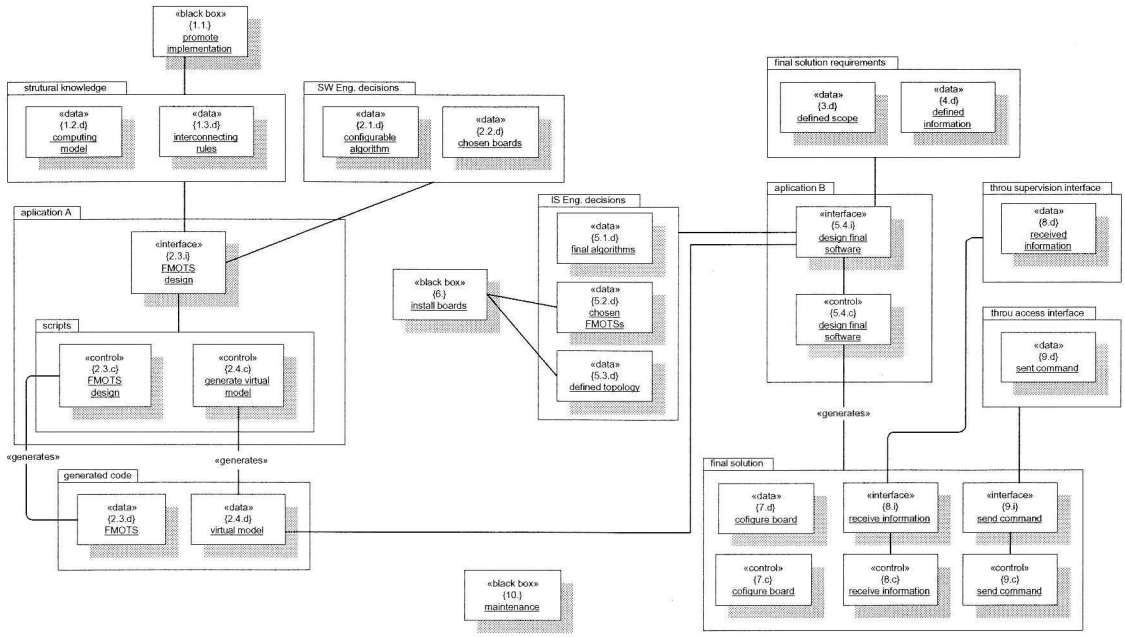

Fig. 4: Process-level object diagram of the three-level co-design approach.

(1) *Application A package*. This package supports the level 2 of co-design by making available: (i) one environment with a HMI interface (*2.3.i design FMOTS*) 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*); (ii) one engine capable of (semi-)automatize both the design of FMOTSs (*2.3.c design FMOTS*) and the generation of final code (*2.3.d FMOTS*) for the system synthesis and

ISBN 972-98603-0-0

implementation in the chosen target architecture; (iii) one engine capable of (semi-)automatize 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*). For implementing the *application A package* the authors are using the OBLOG CASE tool [Andrade *et al.* 1996] (fig. 5).

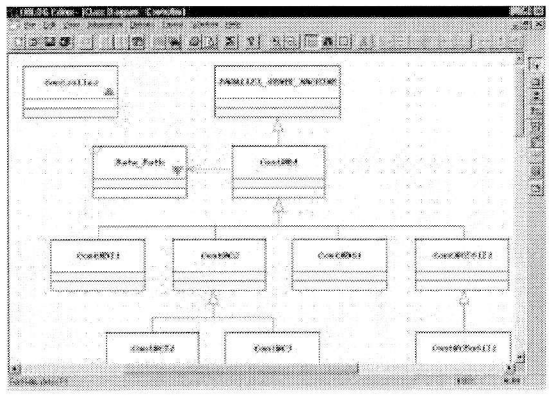

Fig. 5: OBLOG EDITOR design environment.

(2) *Application B package*. This package supports the level 3 of co-design by making available: (i) one environment with a HMI 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*); (ii) one engine capable of (semi-)automatize the generation of final solutions (*5.4.c design final software*). For implementing the *application B package* the authors are using the LabVIEW CAE (*computer aided engineering*) tool.

It is important to note that the *final solution package* corresponds to the ICIS final solution to be installed in the organisation #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*).

Fig. 6 depicts the global EDA three-level co-design environment with the cascaded ECAD, CASE and CAE tools supporting the three engineering designers in the implementation of ICIS final solutions.

6. MODELLING CHARACTERISTICS

The proposed methodology is based in a strong investment in object-oriented specification and design techniques [Machado *et al.* 1998].

In what concerns the specification aspects, it is imperious to assure the following issues: (1) at the language level, to deal with exceptions, to model data-paths/plants in a reactive way and to support multiple-view UML-based operational meta-models; (2) at the complexity control level, to support graphical and hierarchical formalisms and middle-out approaches; (3) at the continuity of models level, to integrate co-related refined representations within the successive design stages for forward and backward navigation.

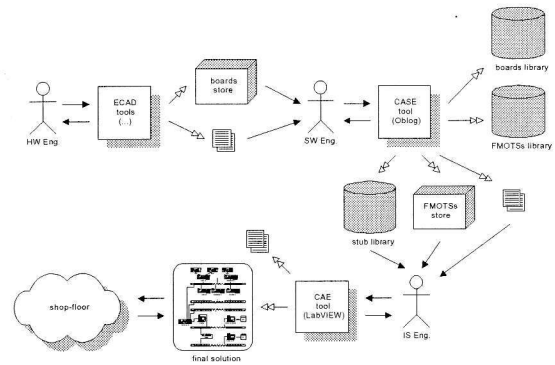

Fig. 6: Global three-level co-design environment.

In what concerns the design aspects, it is critical to adequately deal with the following problems: (1) at the technological constriction level, to decouple the traditional one-all-going project approach into three feed-forward quasi-independent project levels (target architecture design, functional module design, final solution design), each one with a different design flow; (2) at the virtual modelling level, to integrate CASE and CAE design tools by using stubbing techniques; (3) at the module reutilization level, to adopt a component-based development path throughout the three project levels.

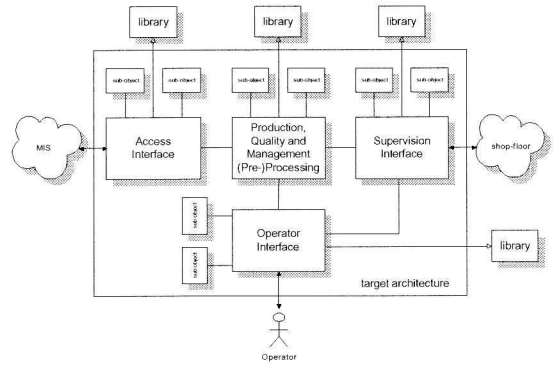

Fig. 7: ICISs architectural pattern.

This last item is based on the definition of some architectural patterns to guide both the FMOTSs and the final solutions structural design [Machado *et al.* 1999]. In fig. 7, it is possible to observe one typical architectural pattern to support ICISs design. This architectural pattern possesses different inheritance hierarchies for each component of the design ("*access interface*", "*production, quality and*

maintenance (pre-)processing", "*supervision interface*" and "*operator interface*"), which demands the construction of a FMOTSs library in the OBLOG repository. Each pattern component can be designed in a compositional approach, since it can be composed of several other sub-objects.

7. CONCLUSIONS

This paper presented a three-level co-design approach, which allows three different IT professionals (hardware engineers, software engineers, and information systems engineers) to participate in the development of a real-time embedded CBS capable of implementing ICISs.

Since the development process was divided into three levels (one per each professional category), 5 issues (timelines, tasks, tools, technology, and talent) must be considered to assure that the systems are consistently developed.

The paper also refers the existence of one UML use cases diagram that shows the responsibilities of the three professionals within the development process. This diagram was transformed into a process-level object diagram that represents the design tools supporting the co-design approach. The tools adopted were OBLOG and LabVIEW.

The proposed methodology is strongly based on the object-oriented paradigm and the systems are graphically specified with UML. The approach presented takes into consideration that a set of modelling mechanisms must be offered to deal with the systems' complexity and to assure that the several models are always inter-related.

This work is not yet completed and the approach presented here is being used to automatize the information flow of a textile industry from the shop-floor (lower CIM levels) to the management (upper CIM levels). The tools are also being adapted and developed. The authors hope that these two projects will help them to gain a better understanding of the problems that the approach introduces, in order to rectify its drawbacks.

REFERENCES

[Andrade *et al.* 1996] Andrade, Luís and Amílcar Sernadas, *Banking and Management Information System Automation*, 13th IFAC World Congress, U.S.A., 1996.

[Buell *et al.* 1996] Buell, Duncan A. and Jeffrey M. Arnold, Walter J. Kleinfelder, *SPLASH 2: FPGAs in a Custom Computing Machine*, IEEE CS Press, 1996.

[Esteves *et al.* 1997] Esteves, António and João Fernandes, Alberto Proença, *EDgAR: A Platform for Hardware/Software CoDesign*, Embedded System Applications, C. Baron, J. Geffroy (Ed.), Kluwer, 1997.

[Fernandes *et al.* 1999] Fernandes, João and Ricardo Machado, Henrique Santos, *A UML-based Approach for Modeling Industrial Control Applications*, 2nd OMG/IEEE/ACM Int. Conf. on the Unified Modeling Language - UML'99, E.U.A., October, 1999.

[Fernandes *et al.* 2000] Fernandes, João and Ricardo Machado, Henrique Santos, *Modeling Industrial Embedded Systems with UML*. 8th IEEE/IFIP/ACM Int. Workshop on Hardware/Software Co-Design - CODES'2000, E.U.A., ACM Press, May, 2000.

[Guccione_list] Guccione, Steve, *List of FPGA-based Computing Machines*, http://www.io.com/~guccione/HW-list.html.

[Hutchings *et al.* 1995] Hutchings, Brad L. and Michael J. Wirthlin, *Implementation Approaches for Reconfigurable Logic Applications*, Field- -Programmable Logic Applications, LNCS 975, Springer, 1995.

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

[Machado *et al.* 1997] Machado, Ricardo and João Fernandes, Alberto Proença, *SOFHIA: A CAD Environment to Design Digital Control Systems*, Hardware Description Languages and Their Applications: Specification, Modelling, Verification and Synthesis of Microelectronic Systems, C. Kloos, E. Cerny (Ed.), Chapman & Hall, 1997.

[Machado *et al.* 1998] Machado, Ricardo and João Fernandes, Alberto Proença, *An Object-Oriented Model for Rapid-Prototyping of Data Path/Control Systems - A Case Study*. 9th IFAC/IFIP Symp. on Information Control in Manufacturing – INCOM'98, France, June, 1998.

[Machado *et al.* 1999] Machado, Ricardo and João Fernandes, Henrique Santos, *Architectural and Methodological Concerns for Industrial Real-Time Applications: An Hardware/Software Co-Design Approach*, Workshop on Architectural Aspects in Specification and Design - AASD'99, Portugal, December, 1999.

[Machado *et al.* 2000] Machado, Ricardo and João Fernandes, António Esteves, Henrique Santos, *An Evolutionary Approach to the Use of Petri Net based Models: From Parallel Controllers to HW/SW Co-Design*, Hardware Design and Petri Nets, A. Yakovlev, L. Gomes e L. Lavagno (Ed.), Kluwer, 2000.

[Madisetti *et al.* 1996] Madisetti, Vijay K. and Mark A. Richards, *Advances in Rapid Prototyping of Digital Systems*, IEEE Design & Test of Computers, Fall, 1996.

[Mangione-Smith *et al.* 1997] Mangione-Smith, William and B. Hutchings, D. Andrews, A. DeHon, C. Ebeling, R. Hartenstein, O. Mencer, J. Morris, K. Palem, V. Prasanna, H. Spaanenburg, *Seeking Solutions for Configurable Computing*, IEEE Computer, December, 1997.

[Ranky 1990] Ranky, Paul G., *Computer Networks for World Class CIM Systems*, CIMware Limited, 1990.

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

[Voas 1998] Voas, Jeffrey, *The Challenges of Using COTS Software in Component-Based Development*, IEEE Computer, June, 1998.

[Waldner 1992] Waldner, Jean-Baptiste, *CIM: Principles of Computer-Integrated Manufacturing*, John Wiley & Sons, 1992.

[Yen *et al.* 1996] Yen, Ti-Yen and Wayne Wolf, *Hardware-Software Co-Synthesis of Distributed Embedded Systems*, Kluwer, 1996.

ISBN 972-98603-0-0