Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Solving the deployment problem of IEC 61499 applications

Tanvir Hussain and Georg Frey

University of Kaiserslautern, Dept. of Electrical and Computer Engineering, 67663 Kaiserslautern, Germany
e-mail: {hussain|frey}@eit.uni-kl.de

Abstract: Distributed control systems now-a-days are consisting of more and more heterogeneous processing nodes. Standards are making it easier to use processors, networks, operating systems etc. of varying nature. This freedom of choice is quite beneficial as long as the control applications perform according to the functional and non-functional specifications. Non-functional requirements especially that related to satisfaction of certain temporal deadlines are quite important in control applications since they often consist of a number of real-time components. Therefore, when heterogeneous processing elements are used deployment of the software components appears to be a problematic task. This article presents a methodology to combat this problem using explanation based learning.

1. INTRODUCTION

Distributed Control Applications of Industrial Process Measurement and Control Systems (IPCMSs) are now containing increasing degrees of heterogeneity – from the viewpoint of hardware as well as middleware and software components. This is mainly to achieve flexibility and reconfigurability which is essential to cope with dynamic market demands and rapid technological developments.

Still control system should meet certain temporal requirements beside the essential functional requirements. This makes it difficult to decide on deployment of software components of a Distributed Control System (DCS) on heterogeneous processing nodes. The objective of this paper is to present a way to combat this difficulty.

Software in the realm of IPMCS conforms either to proprietary regulations or general standards. The latter is more effective for development of flexible and reconfigurable manufacturing systems since it enables vendor independence and thus opens up a wider range of choice for developers as well as users. One such standard for development of distributed control applications is IEC 61499 (IEC Standard, 2005). This standard allows one to model software elements in terms of reusable components which later can be deployed on heterogeneous processing nodes as long as those nodes conform to a compliance profile defined in the standard. Therefore, in the context of this paper the deployment problem is solely concerned with applications developed using IEC 61499.

The rest of this paper is structured as follows. Section 2 formulates the problem of deployment firstly in general for DCS and then for IEC 61499 compliant ones. A solution to the problem of deploying IEC 61499 compliant control application is proposed in section 3 and later compared with respect to related approaches in section 4. Section 5 briefly presents an example of application of the methodology and accompanying results of the deployments. Finally section 6 contains concluding remarks and concerned outlooks.

2. PROBLEM STATEMENT

In general, deployment of components of a distributed control application on heterogeneous processing considers three types of constraints (Cambazard *et al*., 2004):

1) **Resource Constraints:**

   a) Memory Capacity: the available physical memory space for running the control application is certainly limited as far as the usual embedded devices are concerned.

   b) Utilization Factor: during its cycle processors cannot be kept busy in executing control applications beyond a defined percentage of time.

   c) Network Usage: certain network infrastructures restrict the number of messages that can be exchanged during a periodic interval to a defined maximum.

2) **Allocation Constraints:**

   a) Residence: software components can depend upon certain hardware or middleware features or might have run-time dependencies upon certain software components which limit their allocation to certain processing nodes.

   b) Co-residence: system architecture might even force that certain components are to be placed on the same processing node (i.e., components that share common resources).

   c) Exclusion: this is the opposite of the previous inhibiting co-existence of software components.

3) **Time Constraints:** this is the most important constraint that any distributed real-time system must satisfy. They are usually stated in terms of end-to-end response times. Often correlative relations among the response times of tasks are also considered (Gerber *et al*., 1995).

In what follows a particular distributed control application is considered to show an example of the above mentioned constrains. A schematic of the workstation or process that is to be controlled is depicted in Fig. 1. During the normal work-

ing process a cylindrical object will be delivered on the conveyor from the preceding workstation and the conveyor will bring it up to the 1st slot of the rotary table. The conveyor should then stop moving and the rotary table will start rotating counterclockwise to move the object to the 2nd position. As soon as the rotary completes a 90° rotation a sensor signal is turned on and the control application should stop the rotation within a time span of 5ms to keep the object properly in place. An object at the 2nd position of rotary table can be drilled and when at the 3rd position the drilled hole of the object can be checked while a gantry crane can pick the object up from 4th position to put it on any of the repositories depending on the material it is made of or on the slider if it is not properly drilled. As the gantry crane moves in horizontal or vertical direction its position is sensed through 4 sensors installed on the vertical bar and 5 sensors on the horizontal bar. The control application should respond to any of this sensor signals in no less than 5ms otherwise the crane will not be aligned properly to its destined place.

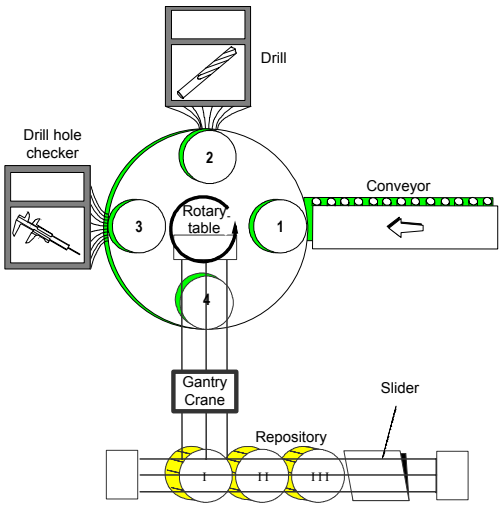

Fig. 1. Schematic of the workstation which is to be controlled

The control application is to be compliant with IEC 61499. This standard primarily defines models to describe distributed control applications. The application and the components of it should show reactivity and therefore should have a defined semantics that will imply how the modeled elements should react to external signals. Moreover, since the domain of application of the standard is principally the embedded real-time controllers of IPMCS, execution semantics should also be compliant to the underlying infrastructure, for example, the middleware, hardware and programming environments etc. Till now a number of execution semantics evolved and came into discussion among the concerned researchers. Some of the execution semantics had also been implemented in different tools or applications showing their suitability for the desired applications. In current context Non-Pre-emptive Multi-Threaded Resource (NPMTR) semantic first presented in (Sünder *et al.*, 2006) is considered. Salient features of this resource model are as following:

- Only components responsible for communicating with process or network interfaces (called Service Interface Function Block (SIFB)) within a resource can have their own threads. The threads of an SIFB can be used for receiving messages, for queuing them etc.
- In IEC 61499 a resource is a functional unit that might contain one or more SIFBs as well as other Function Blocks (FBs) that behave in response to events according to a specialized state chart called Execution Control Chart (ECC) or according to the behavior of its contained FBs for composite FBs. The FBs execute algorithmic actions and their interconnections within a resource determine in what sequence the actions will be executed. Any such event sequence is allotted its own thread for execution of constituent actions.
- Threads of the same resource should have equal priority; those of different resources may have different priorities.

This execution semantics accounts for simplicity in dealing with data to be exchanged between two FBs while a corresponding event triggers an FB. But certain events that arrive at a resource while it is busy in synchronously executing actions will be lost. To combat this deficiency the Event Dispatcher Concept (EDC) evolved (Zoitl *et al.*, 2006). It assumes that events are stored in a FIFO queue and consumed when the corresponding thread is notified about its allotted execution time. In what follows NPMTR semantics with EDC enhancement is considered. Furthermore, the priorities of the threads are assigned such that all SIFB threads are allotted the highest priority and threads for other FB chains are allotted depending on the ratio of the sum of execution requirements of the constituents with respect to the period (or minimum inter-arrival time) of their triggering event.

The available processing nodes are modern network-enabled microcontrollers (i.e., TINI, SNAP etc.) connected to a dedicated Ethernet network. The microcontrollers are equipped with peripherals to access digital sensor signals and to trigger actuators.

Considering a control application for the workstation of Fig. 1 in general the constraints are as follows:

1) Resource Constraints:

a) Memory Capacity: The code and related resources as well as the runtime memory requirement are limited by the memory capacity of the processor nodes. For example, SNAP offers 2MB flash memory for storage of code and related resources.

b) Utilization Factor: Firstly utilization factor should be below 100%. Secondly, for multithreaded applications the number of threads should be less than the maximum number of allowed threads that can be created on a processor (assumed to be 6 on SNAP and TINI microcontrollers).

c) Network Usage: As far as Ethernet is used this does not impose any restriction since the size of the messages to be exchanged can occupy only a scanty portion of the available bandwidth.

2) Allocation Constraints:

a) Residence: Current problem does not contain any such constraint.

b) Co-residence: Every FB that needs data or event triggers from FBs deployed on a different processing node should use a corresponding SIFB to accomplish the communication. This is an implicit co-residence constraint that applies to the problem.

c) Exclusion: The application needs timers to actuate cyclic reading of inputs or timed operation of drilling which are to be realized through use of hardware timers. Since only two hardware timers are available per processing node, no more than two such timer components that might operate simultaneously can be placed on the same processing node.

3) Time Constraints:

a) End-to-end response time between arrival of sensor signal that the rotary table has rotated by 90° and the actuation to stop the table motor should be within 5ms.

b) End-to-end response time between detection of arrival of the gantry crane at a stop position and actuation of corresponding signal should be within 5ms.

c) If more than one object is allowed to be placed on the conveyor then after arrival of the first one at rotary table's 1st slot the conveyor should have to be stopped within 4ms.

Finally, in the deployment problem the feasible solutions allowing maximal amount of slacks for the time constrained tasks as well as usage of minimum number of processing nodes will be considered optimal.

3. PROPOSED SOLUTION

3.1. Logic-based Benders decomposition

The problem of finding all feasible deployments satisfying a DCS specification has an attribute that, when taken into account, can simplify the problem. It can be observed that the resource and allocation constraints are rather static in nature and thus can easily be modeled in a tractable form (i.e., as an Integer Programming problem) while the time constraint is such that the dependencies among the variables cannot be expressed in a straightforward manner. But still solving the problem separately for these two types of constraints might often not be of good use. This is due to the fact that the first type of constraints usually account for a small reduction of the search space leaving the major part of the search space to be scrutinized through the more complex temporal relations.

In such a situation a variation of Benders decomposition called logic-based Benders decomposition (Hooker and Ottoson, 2003) appears to offer a better way of solving the problem. Classical Benders decomposition applies to problems of following form:

$$\begin{aligned} &\min \ cx + f(y) \\ &\text{s.t.} \ \ Ax + g(y) \geq a \\ &\qquad x \geq 0 \\ &\qquad x \in R^n, y \in D_y, \end{aligned} \tag{1}$$

where $g(y)$ is a vector of functions $g_i(y)$. The solving strategy of the classical Benders decomposition is to fix y to a trial value so that the problem reduces to the following linear subproblem.

$$\begin{aligned} &\min \ cx + f(\bar{y}) \\ &\text{s.t.} \ \ Ax \geq a - g(\bar{y}) \\ &\qquad x \geq 0. \end{aligned} \tag{2}$$

The subproblem dual can be written as

$$\begin{aligned} &\max \ u(a - g(\bar{y})) + f(\bar{y}) \\ &\text{s.t.} \ \ uA \geq c \\ &\qquad u \geq 0. \end{aligned} \tag{3}$$

If the dual has a finite solution u, it provides an inference procedure for obtaining a lower bound on the objective function of the original problem which valid for $y = \bar{y}$. The key to obtaining a bound to any y is to observe that this same u remains feasible in the dual for any y. So the same u provides a bound for any y. This bound is called Benders cut. In case of infeasible or unbounded dual one can obtain a cut

$$v(a - g(\bar{y})) \leq 0$$

where v can be obtained through solving

$$\begin{aligned} &\max \ v(a - g(\bar{y})) \\ &\text{s.t.} \ \ vA \leq 0 \\ &\qquad v \geq 0. \end{aligned}$$

In contrast to generating cuts using the linear programming dual, logic-based Benders decomposition uses a generalized notion of dual called inference dual (Hooker, 2006) which can be inferred from the constraints. In this case the problem can be represented as,

$$\begin{aligned} &\min \ f(x, y) \\ &\text{s.t.} \ \ (x, y) \in S \\ &\qquad x \in D_x, y \in D_y, \end{aligned} \tag{4}$$

Then fixing y at some trial value $\bar{y} \in D_y$ the following subproblem is obtained.

$$\begin{aligned} &\min \ f(x, \bar{y}) \\ &\text{s.t.} \ \ (x, \bar{y}) \in S \\ &\qquad x \in D_x. \end{aligned} \tag{5}$$

Following the pattern of the classical case at this point the dual of this subproblem should have to be solved. Since the subproblem is not a linear one inference duality comes into play. The inference dual of the subproblem can be defined as:

$$\begin{aligned} &\max \ \beta \\ &\text{s.t.} \ \ (x, \bar{y}) \in S \xrightarrow{D_x} f(x, \bar{y}) \geq \beta. \end{aligned} \tag{6}$$

In this abstract representation of Benders decomposition the main challenge is to define a complete inference method for the implication $\xrightarrow{D_x}$.

In case of the deployment problem the resource and allocation constraints are thus used to initially find a valid allocation and a schedulability analysis is performed on this valid allocation to infer certain additional constraints. These constraints are then added to the resource and allocation constraints to cause faster reduction of the search space. For example, when a particular allocation of tasks on a platform fails to meet one or more temporal constraints, then the inference process can find out the combination of tasks that can-

not be allocated on the same processor as far as the temporal restrictions are concerned. This process goes on until a set of mappings of software components on processing nodes is found where none of the allocations violate the temporal constraints.

In (Cambazard *et al.*, 2004) it is shown that logic-based Benders decomposition works for allocation of tasks on distributed hardware components. The context of the current problem is rather complex in comparison as the temporal behavior of the software components cannot be analyzed using simple rate monotonic analysis. Analysis of temporal behavior of the software component in this context means to investigate that the deadlines or end-to-end response times of the applications are fulfilled. This is usually done in the worst case sense. In the following subsection a description of analyzing such worst case scenarios for IEC 61499 compliant distributed control application is presented.

3.2. Worst Case Response Time (WCRT) calculation

For WCRT calculation of applications conforming to the above mentioned semantics a method based on the one presented in (Saksena and Karvelas, 2000) is taken. First the WCRTs for actions deployed on a processor node is calculated and then the overall end-to-end delays are calculated by summing the individual response times and network delays.

The method of WCRT calculation uses busy period analysis where external events instigate the beginning of a transaction (i.e., a chain of actions). An action can invoke a following one through synchronous or asynchronous message passing. This perfectly matches with the sequence of actions of IEC 61499 applications. Synchronous messages are passed within actions of own thread and asynchronous messages are sent to actions of other threads. Synchronous sequence of actions can only be preempted by another thread of higher priority.

Through an iterative calculation of worst case start time $S_i^\tau(q)$ of *q-th* instance of *i-th* action in transaction τ is found. Each algorithmic action can be blocked by another algorithmic action due to run-to-completion characteristic of the algorithmic actions. Within the priority scheme this effect can be realized through use of preemption threshold (Saksena and Karvelas, 2000). For IEC 61499 compliant applications the preemption threshold of an algorithmic action is assumed to be below the priorities of any of the SIFBs so that the algorithmic actions cannot block the tasks to be accomplished by a SIFB. Actions can experience interference from any communicating actions that have to be performed by SIFBs as well as actions of any other transaction of higher priority. Moreover, previous instances of the same transaction also account for interferences for an action. Thus $S_i^\tau(q)$ comes out to be the smallest W that satisfies:

$$\begin{aligned} W &= B(A_i) \\ &+ \sum_{k\neq\tau}[\Psi_k^+(W)\cdot\sum_l(C(A_l^k)\mid\pi(A_l^k)\geq\pi(A_i^\tau))] \\ &+ (q-1)\cdot\sum_l(C(A_l^\tau)\mid\pi(A_l^\tau)\geq\pi(A_i^\tau)) \end{aligned} \quad (7)$$

where $\Psi_i^+(t)$ denotes the maximum number of arrivals of event triggering transaction i in any right closed interval $[x, x+t]$. Correspondingly, $\Psi_i(t)$ denotes the maximum number of the same event arrival within the interval $[x, x+t)$.

WCRT of an action A_i^τ can therefore be expressed by the following equation

$$\begin{aligned} &F_i^\tau(q) = \min W \\ &\text{where } W = S_i^\tau(q) + C(\Upsilon(A_i^\tau)) \\ &\qquad + \sum_k(\Psi_k(W) - \Psi_k^+(S_i^\tau(q)))\cdot L_k(A_i^\tau) \end{aligned} \quad (8)$$

$$\text{and } L_k(A_i^\tau) = L_k(A_i^\tau, A_k^k)$$

$$L_k(A_i^\tau, A_j^k) = \begin{cases} 0 & \text{if}(\Gamma(A_i^\tau)=\Gamma(A_j^k)\vee\pi(A_j^k)<\gamma(A_i^\tau)) \\ C(\Upsilon(A_i^\tau)) + \sum\limits_{g\mid A_j^k\rightarrow A_g^k} L_k(A_i^\tau, A_g^k) & \text{otherwise} \end{cases}$$

where $\gamma(A_i^\tau)$ denotes the preemption threshold of action A_i^τ and $\Upsilon(A_i^\tau)$ denotes the chain of synchronous actions beginning at A_i^τ.

3.3. Deployment resolution algorithm

The deployment resolution algorithm is composed of three components mentioned bellow.

- WCRT calculating component that follows the above mentioned calculation scheme.
- An explanation generating component that generates explanation for any infeasibility with respect to temporal constraints. Hereby QUICKXPLAIN algorithm (Junker 2001) is used which is quite efficient in finding minimal set of variables involved in a conflict. This component iteratively uses WCRT calculating component to find out the conflicting set of software components allocated in a processing node. These explanations in form of nogoods (implying that none of the feasible solutions will contain such a set). Since in parallel optimal allocation is also sought, the lower bound on the objective value is also improved after each iteration of successful explanation. In the following searches the allocations exceeding this limit are excluded.
- The constraint solver which should solve the master problem (consisting of resource and allocation constraints) at each step to form the subproblem (containing only time constraints). Moreover, its task is to integrate the minimal set of software components involved in conflicts. This is performed through maintaining arc consistency and conflict directed backjumping (Jussien *et al.* 2000). Thus after integrating the constraints learned from explanation inefficiency of a standard backtrack is avoided through jumping to a node appearing in the explanation for raising a contradiction.

In synergy these three components solve the deployment problem. Furthermore, during each iteration partially feasible solutions are tried for overall feasibility. In case of infeasibility, explanations are used to enrich the next search. The solution process tries to achieve efficiency through use of explanations to learn about some partial solutions which could not be successfully extended.

4. RELATED WORKS

Prayati *et al.* pioneered in presenting a methodology for development of IEC 61499 compliant distributed control applications along with an algorithm for deployment of developed artifacts on heterogeneous processing nodes (Prayati *et al.*, 2004). The deployment uses Branch and Bound (B&B) where each vertex represents allocation of a closely coupled set of tasks to a physical resource and the compatibility factors other than the real-time constraints are used in pruning the branches at each level. The objective function for the optimal allocation is the feasibility ratio which refers to the probability of the set of tasks represented by a vertex to meet its deadline. It is shown that a preprocessing of the search space through priority slicing of the FB tasks and clustering of the most tightly coupled tasks shows positive effects through raising the feasibility ratio of the resulting schedules and success ratio of the B&B respectively.

Another work that emphasized the deployment aspect of IEC 61499 compliant distributed control application is that of Khalgui et al. In (Khalgui *et al.*, 2006) they presented a heuristics based approach for deployment of IEC 61499 applications where exclusion and localization constraints are considered along with the resource constraint which restrains a maximum number of schedulable tasks on a physical device. The deployment heuristic attributes static priority levels for each of the exclusion sets and then performs allocations in descending order of priority. After each deployment an integrated feasibility test of the concerned device and the communication network is performed. In case of infeasible deployments the above mentioned deployment step is performed repeatedly until feasibility is achieved. Unlike in (Prayati *et al.*, 2004) where optimization of an objective function is considered during deployment, here only a feasible deployment is looked for. Moreover, the algorithm requires formal specifications of the FB constructs and it is implicit how this specification can be achieved during a usual development lifecycle of a distributed control application.

In contrast to these works the proposed work presents a methodology where finding a feasible deployment and an optimal one can be performed in parallel. As far as satisfaction of temporal behavior is concerned only worst-case scenario is taken into consideration which is quite pessimistic even though it guarantees that the end-to-end response times are satisfied under any circumstance.

5. CASE STUDY

5.1. Description of the example system

A brief description of a workstation of a modular production system is given in section 2. Now the deployment of the control application of such a workstation is considered. The application consists of a number of FBs – both for interfacing and for performing algorithmic actions. Though the dependencies among them are quite complex it can quite easily be captured through the representation of object oriented applications presented in (Saksena and Karvelas, 2000). Each chain of action is thus represented as a transaction which has a period (for sporadic tasks this is equal to minimal inter-arrival time). Events that trigger an action contained in an FB can be either external events (i.e., interrupt, arrival of network event and signal from timers) or internal events of asynchronous or synchronous type. Asynchronous internal events are caused by another thread, while the synchronous events are caused by preceding FB in the FB network.

The overall application consists of 8 transactions – 2 of which are periodic and the others are either sporadically periodic or sporadic. The periodic transactions cyclically read the sensor inputs through SIFBs. Only one of the other transactions requires using a timer.

Initially it is assumed that two processing nodes are available for the control application. Moreover, it is assumed that each transaction can be allowed to be allotted a thread and more than one transaction can be accommodated in a thread if they share one or more FBs. Otherwise, when such transactions are to be realized on different threads the common FB instances are accessed through asynchronous event exchange.

5.2. Deployment of control application

The deployment problem has been solved using the presented algorithm. The control application for the single workstation can be deployed in two processors using 6 threads in total. The optimal solution is achieved with an average slack of 31.3% for the time constrained end-to-end response times. This allocation is summarized in Table 1.

Table 1. Deployment of functional components

Functionality	Thread index	Proc. index
Rotary control	1	1
Conveyor control	1	1
Drill module control	2	1
Drill hole check control	3	1
Vertical Crane control	4	2
Horizontal Crane control	4	2
Cyclic input reading	5 & 6	1 & 2

5.3. Extension

A more complex variation of the problem is formed through considering that an IEC 61499 control application is to be deployed which has to control four workstations of same type and the application is to be deployed on four processing nodes. For this problem it is seen that a minimum of 6 processing nodes is needed. It needs 28 threads and the average slack reduces to 1.5%. It is quite reasonable since now the

number of processing nodes in use is proportionally less, which accounted for proportionately more usage of threads and reduced average amount of slack. It is to be emphasized that the schedulability analysis is a very pessimistic one and therefore in most of the scenarios the slack time will appear to be quite large.

5.4. Random Experiments

Certain random scenarios were generated and fed to the deployment algorithm to observe its convergence for problems of different complexity.

In each of these experiments, the effect of resource and allocation constraints is assumed to result in 20% reduction of the possible allocations which are also randomly chosen. Further, in each application 4 transactions each having 10 tasks are considered. Three different structures of the transactions are considered: a) linear, b) Directed Acyclic Graph (DAG) with a linear subsequence followed by two branches and c) DAG with a linear subsequence followed by 4 branches. In each case, the difficulty of the problem is raised by increasing the ratio of the total duration of the longest algorithm subsequence in the DAG with respect to the deadline. Table 2 lists the number of feasible solutions obtained from 100 samples of each type of problem where the aforementioned ratio is denoted using *U*. Increased *U* causes more of the problems to be infeasible. Moreover, it can be seen that the branching variants are more vulnerable than the linear ones although a conclusive comment cannot be made, when the two branching variants are compared. The algorithm took on average 193.8 sec to deliver a result of a problem whereby 980ms was the minimum and 1080 sec the maximum time taken (calculations on an AMD Athlon XP 2.08 GHz PC with 1GB RAM).

Table 2. Results of random experiments

Structure	*U*=0.4	*U*=0.5	*U*=0.6	*U*=0.8
Linear	79	78	70	46
2-branch	78	68	56	44
4-branch	76	74	58	32

6. CONCLUSION AND OUTLOOK

This paper presents a methodology for finding a feasible as well as optimal deployment for IEC 61499 compliant distributed control application on heterogeneous platforms. The algorithm uses explanation based learning to effectively converge into a solution which is quite a necessity for such reconfigurable applications. Later application of the algorithm on certain problems has been shown. Still there is room for certain improvements, especially regarding the response time analysis of the distributed applications. Moreover, for certain problems it could happen that it appears to be over-constrained resulting in failure to deliver a feasible deployment at all, in that case explanation can be used to point out possible relaxations.

REFERENCES

Benders, J. F., (1962). Partitioning procedures for solving mixed-variables programming problems, In: *Numerische Mathematik*, **4**, pp. 238-252.

Cambazard, H., P.-E. Hladik, A.-M. Déplanche, N. Jussien, and Y. Trinquet (2004). Decomposition and learning for a hard real time task allocation Problem, In: *Principles and Practice of Constraint Programming (CP 2004)*, **3258,** pp. 153-167.

Gerber,R., M. Saksena and H. Hong (1995). Guaranteeing end-to-end timing constraints by calibrating intermediate processes, In: *IEEE Transaction on Software Engineering*, **21(7)**, pp. 579-592.

Hooker, J.N. and G. Ottoson (2003). Logic-based Benders decomposition, In: *Mathematical Programming*, **96**, pp. 33-60.

Hooker, J.N., (2006) Duality in optimization and constraint satisfaction, In: *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, **3990**, pp. 3-15.

IEC Standard (2005). IEC 61499-1: Function Blocks – Part 1: Architecture, 2005.

Jussien, N., R. Debruyne and P. Boizumault (2000). Maintaining arc-consistency within dynamic backtracking, In: *Constraint Programming 2000 (CP 2000)*, 1894, pp. 249-261, Springer-Verlag.

Junker, U. (2001). QUICKXPLAIN: conflict detection for arbitrary constraint propagation algorithms. In: *Proc. of International Joint Conference on Artificial Intelligence*.

Khalgui, M., X. Rebeuf and F. Simonot-Lion (2006). A heuristic based method for automatic deployment of distributed component based applications, In: *Proc. Of International Symposium on Industrial Embedded Systems (IES'06)*.

Prayati, A., C. Koulamas, S. Koubias and G. Papadopoulos (2004). A methodology for the development of real-time control applications with focus on task allocation in heterogeneous systems, In: *IEEE Transactions on Industrial Electronics*, **51(6)**, pp.1194-1207.

Prayati, A., S. Koubias and G. Papadopoulos, (2002). Real-time aspects in the development of function block oriented engineering support systems, In: *Proc .of 4th IEEE Intl. Workshop on Factory Communication Systems (WFCS'02)*, pp. 157-164.

Saksena, M. and P. Karvelas (2000). Designing for schedulability: Integrating schedulability analysis with object-oriented design, In: *Proc. of 12th EuroMicro Conference on Real-Time Systems (EMRTS'00)*, pp. 101-108.

Sünder, C., A. Zoitl, J. Christensen, V. Vyatkin, R. Brennan, A. Valentini, L. Ferrarini, T. Strasser, J. L. Martinez-Lastra, and F. Auinger (2006). Usability and interoperability of IEC 61499 based distributed automation systems, In: *Proc. of 4th IEEE Conference on Industrial Informatics (INDIN'06)*, pp. 31-37.

Zoitl. A., R. Smodic, C. Sünder and G. Grabmair (2006). Enhanced real-time execution of modular control software based on IEC 61499, In: *Proc. of IEEE Intl. Conference on Robotics and Automation (ICRA'06)*, pp. 327-332.