Downtimeless System Evolution: Current State and Future Trends

Oliver Hummer, Christoph Sünder, Thomas Strasser, Martijn N. Rooker and Gerold Kerbleder

Abstract— **Recent studies report a growing demand for downtimeless evolution of distributed automation systems. This paper therefore first analyzes evolution approaches from the industry as well as from academic research pointing out benefits and drawbacks. The standard IEC 61499 is identified as promising basis for dynamic reconfiguration of control applications, but it lacks for an adequate engineering methodology. Finally the εCEDAC approach for structured evolution modeling is presented. Therein a distributed evolution control application modeled in terms of IEC 61499 is used for transforming the control application from current to desired state without downtime. Further important issues are correctness of the evolution process and runtime error handling mechanisms.**

I. Introduction

THE future of manufacturing industries in Europe is currently a topic for frequent discussions. Competitors from Far East dominate the international markets with low-price products, particularly in the domain of mass production. In opposite to former times, these products tend to approach European quality levels. This development is a major threat for European production industries. Therefore such industries in traditional high-wage countries need unique selling propositions more than ever.

In recent studies, such as Favre-Bulle and Zeichen [1], international domain experts identified flexible and fast reaction on changing customer demands as key distinguishing factor in this competition. Resulting policies, such as "built to order" or "lotsize one", strongly require flexible and adaptive production facilities, which is also confirmed by investigations of the European Commission [2]. At the manufacturing system level these requirements create an urge not only for enhanced reconfigurability, but even more for dynamic reconfigurability , since downtimes are very costly. Such dynamically reconfigurable systems can be modified at runtime, without recompiling or relinking the control application and without shutdown or reboot of the system.

A prerequisite for fast reconfiguration is the change from single monolithic system architectures to structured, component based design concepts with defined interfaces. Such components can be combined in order to jointly realize the desired application [3]. In the mechanical domain (automation hardware) the modularization issue can be considered solved to a sufficient degree, whereas the software part of automation systems falls behind: present production systems predominantly rely on programmable logic controllers (PLCs) programmed according to the standard IEC 61131-3 [4] for lower level control. Although this standard provides architectural elements for encapsulation of software in an object based paradigm (mainly the function blocks), it is not well suited for engineering distributed systems (due to modularization intelligence becomes distributed) and dynamic reconfiguration [5], to be explained in more detail in section III-B.1.

For that reason the International Electrotechnical Commission (IEC) finally released IEC 61499 [6] as successor standard in 2005. It specifies an architectural model for distributed applications in industrial process measurement and control systems (IPMCS) serving as reference architecture and extends the IEC 61131-3 function block concept by additional event-handling mechanisms. The design paradigm of this new standard is *application centered*, i.e. control applications are designed as a whole, assigning execution of application parts to concrete devices is done afterwards via *mapping*. Hardware access is abstracted in so-called *service interface function blocks (SIFBs)* with a defined interface for use in applications, any hardware specifics are left to the implementation of the underlying runtime system. IEC 61499 further introduces a *device management* functionality included in each device in form of a management application, which is accessible at runtime via special device management SIFBs.

The intention of this paper is to provide an overview on the vision of downtimeless system evolution demanded above. Thereby the current state of practise is analyzed with respect to its abilities and limitations. Incorporating the improvements brought by IEC 61499, what is still missing to fully realize this vision? Therefore the rest of the paper is organized as follows: Section II defines the most important terms used in this domain, whereas section III gives an outline on evolution in present industrial automation systems. Subsequently, the aptitude of IEC 61499 for dynamic reconfiguration is discussed briefly. Sections IV and V have a look at evolution approaches from academic research first before deriving a sophisticated evolution engineering method based on means provided by IEC 61499. It will be shown that, if used properly, the ideas and concepts of IEC 61499 allow for performing complex but clearly structured and downtimeless system evolution. Finally

Manuscript received January 15, 2007; revised March 29, 2007. This work is supported by the FIT–IT: Embedded Systems program, an initiative of the Austrian federal ministry of transport, innovation, and technology (bm:vit) within the εCEDAC project under contract FFG 809447. The εCEDAC consortium consists of Bachmann Electronic GmbH, kirchner SOFT GmbH, SIEMENS VAI, LOYTEC electronics GmbH and the research institutes PROFACTOR and ACIN. Further information about the project is available at: www.ecedac.org

O. Hummer and C. Sünder are with the Automation and Control Institute (ACIN), Faculty of Electrical Engineering and Information Technology of Vienna University of Technology. Gusshausstrasse 27-29/376, 1040 Vienna, Austria. (phone: +43(1)58801-37682, fax: +43(1)58801- 37698, e-mail: {hummer,suender}@acin.tuwien.ac.at)

T. Strasser and M.N. Rooker are with the Robotics and Adaptive Systems Department of PROFACTOR Produktionsforschungs GmbH. Im Stadtgut A2, 4400 Steyr-Gleink, Austria (phone: +43 7252 885-309, fax: +43 7252 885-101, email: {thomas.strasser, martijn.rooker}@profactor.at)

G. Kerbleder is with Bachmann Electronic GmbH. Kreuzäckerweg 33, 6800 Feldkirch, Austria (phone: +43 5522 3497-438, fax: +43 5522 3497-102, email: g.kerbleder@bachmann.info)

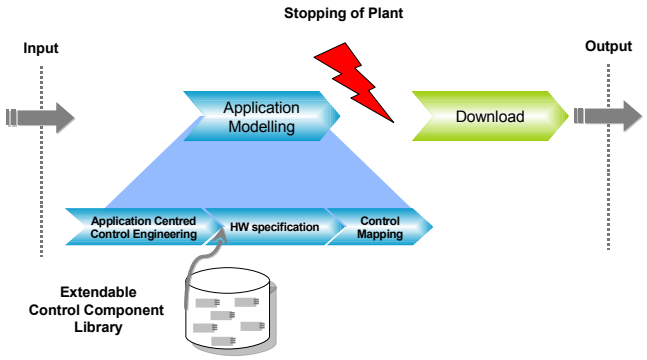


Fig. 1. Simplest evolution case in an IEC 61499 system

the conclusion summarizes the findings of this paper.

II. Terms and Definitions

a) Reconfiguration: Within the context of this paper "reconfiguration" is generally understood as modifying the current configuration of a given system, may it be in hardware or software.

b) Online change: Generally speaking, online change means changing the application(s) of a controller while it is in RUN mode, i.e. without stopping the automation system or controlled process.

c) Evolution: Evolution terms a special case of reconfiguration, whereby available information from the current system configuration is taken into account in order to minimize reconfiguration effort.

d) Evolution Engineering: Performing precise and safe evolution successfully under real-time conditions requires clearly structured and controlled procedures. Within the scope of this paper "evolution engineering" terms the design of the whole evolution workflow consisting of such procedures.

III. Evolution in present Systems

A. Simplest Evolution Case

The simplest case that might still be called some sort of "evolution" is characterized by pure manual work. The control engineer thereby starts with the current software revision of the controllers in question, either available in the according engineering tools or gathered by simple upload. Next the desired changes in the control application(s) are made. Finally the whole automation system of the underlying plant (not only the controllers subject to modifications) has to be stopped before downloading the new program(s) to the hardware and restarted afterwards. This is still a quite common way to apply changes to IEC 61131-3 automation systems today, where even small changes in a single controller can affect the whole system due to tight coupling of different controllers via global variables. Nevertheless, it is also possible to perform this basic kind of evolution on IEC 61499 systems as depicted in Fig. 1.

Drawbacks:

- The major drawback is certainly the stopping of the plant, since it causes enormous loss of both time and revenue.
- A second important aspect is that this simple type of evolution cannot be utilized for runtime adaptation.

B. Evolution via Online Change

1) Cyclic approach (IEC 61131-3): In order to reduce maintenance costs and effort quite a number of automation system vendors, such as [7]–[9], provide an online change mechanism in current versions of their IEC 61131-3 engineering tools, often termed "instant reload" or similar. Typically such systems exchange (i.e. switch from existing to new version) IEC 61131-3 Program Organization Units (POUs) between two execution cycles of the controller, some tools also feature transfer of variable values (i.e. preservation of state information).

What all these approaches have in common is exchange of target code: modified applications have to be recompiled/relinked partially or completely before being downloaded to a different memory segment of the controller. Switching between old and new version is done by adjusting the program start address. It is evident that this technique requires free memory for the new version of the changed POU (old *and* new version are kept in memory at the same time).

Improvements:

- There is no need to stop the plant while modifying the control application (within boundaries of given system architecture and the underlying process).
- In principle this mechanism can be used for (manual) runtime adaptation.

Sünder et al. have analyzed the shortcomings of this approach [10], reporting the following

Drawbacks:

- The switching point in time can not be determined because of the cyclic way of execution and the lack of information about the state of the system or application. Therefore such evolution via online change is limited to a single controller at a given time, since there is no possibility to synchronize switching points.
- The reconfiguration of one task of an application interferes with all tasks of this application since all tasks have to be stopped because of the asynchronous cyclic execution of tasks. This leads to jittering effects.
- The lack of fine granularity (task level) introduces high complexity according to communication, memory management and re-initialization.
- The reconfiguration of elements may lead to inconsistent states, e.g. deadlocks or token-proliferation in Sequential Function Charts (SFC).
- New elements start with their cold start initial values.

2) Event-driven approach (IEC 61499): At the moment many IEC 61499 compliant runtime environments and engineering tools are still in development and therefore only available as evaluation versions. To the author's best knowledge the only commercially available IEC 61499 solution featuring online change is ISaGRAF by ICS Triplex [11]. According to Čengić et al. [12] even this solution is not a pure implementation of the event-driven execution paradigm of IEC 61499 concerning the runtime environment, but instead

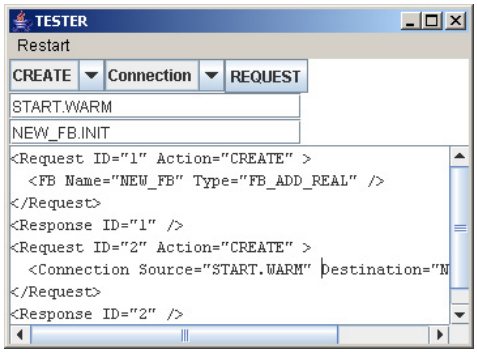


Fig. 2. FBDK remote test application for device management

IEC 61499 has been implemented on top of an existing scan-cycle based IEC 61131 runtime environment. That way IEC 61499 online change suffers from the same drawbacks as in the IEC 61131 approach.

Considering pure IEC 61499 compliant runtime implementations, the configuration of an IEC 61499 device is handled by the device management application. An example for such an implementation is the Function Block Development Kit (FBDK) by Holobloc Inc., free for download at [13]. Within FBDK it is possible to instantiate function block instances as well as event and data connections at runtime, as shown in Fig. 2. Note that the same method is used for initial application download: there is no need to compile whole applications in IEC 61499 (which would not be feasible anyway, since an application may be distributed to different targets), *application download* in the sense of control engineering is done by instantiating components provided by the type library of the runtime system. The device management therefore provides required functions, which can be requested by XML management commands, covering all basic functionality needed to control the lifecycle of ordinary applications and their building parts [10]:

- CREATE: FB instance, data or event connection
- DELETE: FB instance, data or event connection
- START: FB instance, application
- STOP: FB instance, application
- READ: FB instance data outputs
- WRITE: FB instance data inputs
- KILL: FB instance
- QUERY: FB types, FB instance, data or event connection

For illustration, examples of possible actions have been added. Special IEC 61499 SIFBs (the management function blocks) grant access to the device management from IEC 61499 applications, a concrete interface for those FBs is described within the *IEC 61499 Compliance Profile for Feasibility Demonstration* [14].

These few above mentioned commands are sufficient for performing simple dynamic reconfiguration in a straightforward manner. Uhrmann [15], for example, successfully exchanged velocity and position controllers in a gantry system with autonomously controlled axes while the system was moving according to a given motion profile using this approach.

Improvements:

- In IEC 61499 function blocks are fully decoupled, since their internals are only allowed to use data provided by their interfaces. The standard does not allow for global variables or access paths into devices, resources or FBs [6].
- Access to the device management is done via special function blocks with normal event/data interface, thus evolution can be triggered by events making the time of modifications deterministic.
- Since IEC 61499 specifically targets distributed automation systems by abstracting applications from the underlying hardware, evolution is not limited to single devices anymore.

Although more powerful than ordinary online change, this approach also features some

Drawbacks:

- It lacks for a suitable engineering methodology, making evolution hard to understand and therefore fault-prone for the control engineers.
- Even for little changes this procedure is confusing the user.
- There is no possibility to check whether the system is really able to perform the desired changes.
- In case of failure during the reconfiguration there is no strategy to recover.
- On the whole, this method is not acceptable for critical or safety-related processes or equipment.

IV. Enhanced Evolution Method

Various researchers have recognized the suitability of IEC 61499 for online reconfiguration of control systems as well as the weaknesses pointed out above and therefore conducted further research on this topic. For instance, Brennan et al. [16] developed an orthogonal adaptation framework adding additional data and event flows to the IEC 61499 function block model for controlling the control application configuration. Since this caused too much memory and execution overhead for small control devices they developed the contingencies approach [17] in a more recent work, which switches between preprogrammed application scenarios according to the current system state. Drawbacks are that the first approach is not IEC 61499 compliant anymore, whereas the second lacks flexibility and brings about problems with consistency.

Thramboulidis et al. [18] have developed an execution environment supporting runtime-reconfiguration using the Realtime Specification for Java. Therein event propagation is controlled by event handlers, function blocks subject to reconfiguration can be unsubscribed in order to remain idle during the critical phase. This approach is highly dependent on on runtime implementation details, it neither covers consistency issues so far, nor intuitive engineering methods suitable for control engineers at field level.

To overcome the limitations of evolution approaches mentioned before, a sophisticated evolution engineering methodology has been developed in the ongoing research project εCEDAC [19] (Evolution Control Environment for Distributed Automation Components). This method specifically

targets evolution of distributed automation systems based on IEC 61499, taking into account real-time requirements, reliability and safety issues.

As a first main concept the *evolution step* is introduced. The idea is to break down a complex evolution task into a number of small steps that can be executed under real-time constraints. The whole engineering cycle for such a single step is depicted in Fig. 3: the current system state (consisting of applications, automation hardware, the mapping between them and system parameter values) serves as input. If not already available in the engineering tool, the control application to be subject to evolution has to be acquired, where three cases can occur:

1) The latest revision of the application is still available in the engineering tool, no further steps required.
2) The hardware (device) configuration is known, the current application can be gathered by upload from the hardware.
3) The structure of the automation system is unknown, automation devices and their configuration have to be determined by scanning the automation network. Such device identification protocols are outside the scope of IEC 61499 or εCEDAC, nevertheless many system vendors have already implemented this functionality in their (proprietary) products. When finished, the application can again be uploaded.

The next phase is application modeling: the desired changes are engineered within the tool in an application centered manner, modifications to the underlying automation hardware are specified, if any, and finally the control mapping is updated. These tasks have to be done manually by the control engineer, aided by component libraries and hardware description files. Optionally, the modified system configuration can be checked for correctness offline by applying common verification and validation (V&V) or simulation techniques. Further information on verification of downtimeless system evolution can be found at [20].

A second important concept is that, in order to fulfill real-time requirements, evolution shall be executed locally whenever possible, thus eliminating all uncertainties one might encounter when transmitting management commands via network. In order to let control engineers design evolution tasks with familiar tools, this is implemented as follows: reconfiguration services (which consist of sending a certain sequence of management commands) are encapsulated in so-called *reconfiguration service interface function blocks (RSIFBs)*, hence providing a component library for evolution modeling. Detailed research on IEC 61499 runtime reconfiguration services has been conducted in the µCrons project [21]. That way evolution engineering becomes as straightforward as control engineering. At first, the modifications that have to be made are given by the difference ("delta") of current and modified application. A *reconfiguration application*, responsible for performing those modifications, is modeled out of RSIFBs as well as ordinary function blocks like any other IEC 61499 application. In this *evolution control engineering* phase, events and data from the control application may be used for controlling the whole evolution process, e.g. they might serve

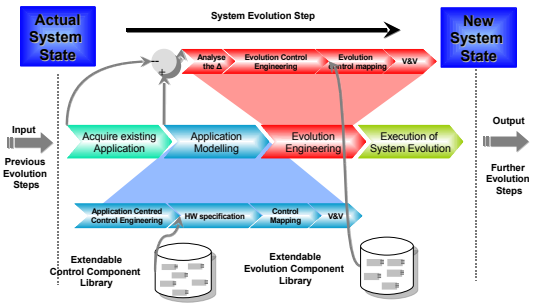


Fig. 3. Enhanced evolution method for a single step

as a trigger signal. *Evolution control mapping* ensures that evolution control takes place locally at the devices where changes have to be made. When the evolution application is finished it can again be checked for correctness via simulation or V&V. Taking into account information from the hardware (e.g. available memory or supported function block types) it can be checked whether the system is able to perform the desired operations.

Each evolution step is concluded by downloading the reconfiguration application onto the hardware and executing it.

In order to improve real-time performance of a reconfiguration application while avoiding unnecessary disturbance of normal system operation, its execution can be divided in 3 phases [10]:

- *RINIT sequence* (time–uncritical): contains all preparation work that is needed to enable a controlled change of applications (for instance new FB types will be instantiated and interconnected).
- *RECONF sequence* (time–critical): encapsulates all time-critical operations necessary to change from the old state to the new one. Considering the exchange of a FB as example this phase includes reading of internal states, calculation of the new internal states, writing the internal state to the new FB instance and finally the switch to the new FB.
- *RDINIT sequence* (time–uncritical): responsible for cleaning up after reconfiguration (e.g. deleting connections and function blocks not needed anymore).

The main advantage of this split is that each evolution step is further subdivided into a time–critical part and two parts being time–uncritical. So only a fraction of each evolution step (the reconfiguration sequence) has to be executed with "run-to-termination" semantics.

Improvements:

- Assuming that all manual tasks will be heavily supported by commercial engineering tools this new methodology guides the engineer through the whole evolution process.
- Both, simulation and V&V, yield a sound basis for deciding if an evolution is feasible or which risk management measures have to be taken.
- Such a standardized procedure allows for a steady quality of evolution tasks.
- Evolution control logic together with event and data flow of the control application enables detection of errors

and illegal system states at execution time. Likewise it is possible to integrate error handling mechanisms, such as rollback or alternative strategies, directly into the reconfiguration application.
- UNDO: Since every IEC 61499 management command changing an application has its exact opposite (e.g. CREATE and DELETE) changes can be reversed easily by keeping a log of all executed commands: undo functionality is achieved by processing this log in reverse order using the complementary commands. If function blocks of the original application are deleted during reconfiguration their internal states must be saved during the RECONF phase to enable a correct UNDO.
- Finally, reconfiguration applications can be reused, e.g. equal changes can be performed on different controllers just by remapping the reconfiguration application.

Drawbacks:

- This evolution engineering method itself does not consider interdependencies between evolution steps.
- Further it does not consider issues such as complexity and presentation to the user, which are very important especially in case of a huge application with numerous evolution steps to be performed.
- Concurrency of execution is not addressed explicitly.

V. εCEDAC Approach: Structured Evolution

Refining the εCEDAC evolution method to finally eliminate the remaining drawbacks of the evolution method yields the εCEDAC *structured modeling* paradigm. *Evolution regions of interest (EROIs)* serve as common structural element for visualization and evolution control engineering. An EROI is defined as "...specific part of an IEC 61499 application that will be target for a specific reconfiguration" [22].

Assuming EROIs independent from each other, reconfiguration on each can be realized in separate reconfiguration applications that run independently. That way reconfiguring an EROI corresponds to one evolution step as depicted in Fig. 3.

Considering dependencies between reconfiguration of different EROIs it is necessary to schedule the execution order of the according reconfiguration applications. IEC 61499 does not provide means for scheduling applications, instead the execution of function blocks is scheduled with respect to occurence of events. Within a function block network possible execution orders are *parallel* or *sequential*, defined by the event flow. These mechanisms are also well suited for modeling reconfiguration of several EROIs that depend on each other. By encapsulating the contents of the according reconfiguration applications into composite function blocks the execution order (which is equivalent to dependency) is determined by event connections. Correspondingly, these special function blocks encapsulating all reconfiguration logic for an EROI (equivalent to an evolution step), are termed *evolution execution control function blocks (Evo-FBs)*. The IEC 61499 application controlling the *whole* evolution of a system, consisting of Evo-FBs along with others, is consequentially termed *evolution control application*. As aid for the user, dependencies can easiliy be visualized as *evolution graph*.

Fig. 4(a) shows such a graph for an evolution consisting of 4 EROIs, where EROI 2 and EROI 3 have to be reconfigured simultaneously while both are depending on successful reconfiguration of EROI 1. Reconfiguration of EROI 4 again depends on successful completion of all previous steps.

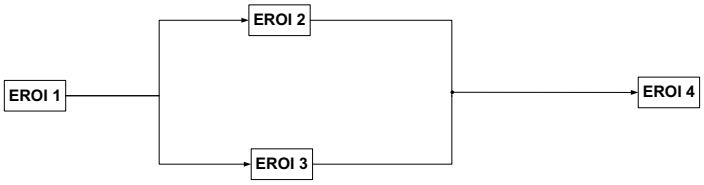
The according evolution control application is depicted in Fig. 4(b). Note that each evolution execution control function block (Evo_EROI_x) has 4 event inputs (INIT for global initialization of the block itself and RINIT, RECONF, RDINIT for starting each of the sequences described in section IV) along with boolean input qualifiers for receiving notification whether the previous step executed successfully or not. Likewise there are as many events with qualifiers on the output side for notifying subsequent blocks. E_REND and FB_AND are standard IEC 61499 library components, in this case used for synchronizing two qualified events.

Improvements:

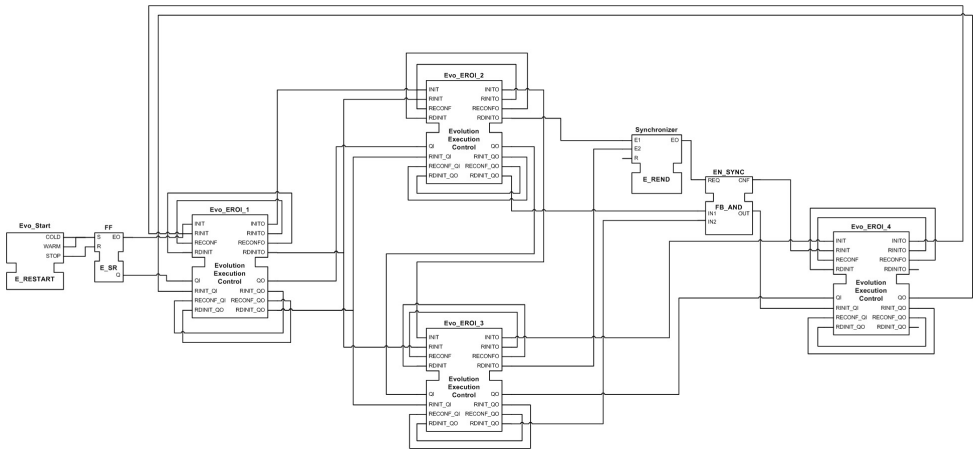
- The whole evolution process is now clearly structured:
  - Each evolution step has an according EROI.
  - Reconfiguration of an EROI is encapsulated in an Evo-FB.
  - The structure of the evolution graph visualizing interdependencies between evolution steps is directly mapped to the structure of the function block network of the evolution control application.
- Complexity of presentation is reduced considerably in the engineering tool.
- The current state/progress of an evolution is given at any time by the qualified event flow between Evo-FBs. This is in particular useful for controlling the progress of the evolution, for instance adding additional FBs in between allows for synchronizing evolution steps with the underlying hardware and/or process (e.g. waiting for a specific system state or user inputs, such as hardware change confirmations).
- Synchronizing evolution across multiple devices is reduced to synchronisation during normal operation (both cases execute a distributed IEC 61499 application).

VI. Conclusion

This paper analyzed the state-of-the-art of system evolution in present IEC 61131-3 and IEC 61499 systems along with all the shortcomings. A sophisticated evolution control modeling technique was then developed step by step to overcome these shortcomings one after another. Complex evolution tasks were clustered in atomic steps (EROIs) that again were subdivided in time-critical and time-uncritical sections. Encapsulation of IEC 61499 (re-)configuration services into function blocks allows the control engineer to model reconfiguration with familiar tools. The work of Uhrmann [15] provides the proof of concept for system evolution via IEC 61499, εCEDACadds an appropriate engineering methodology. Evolution execution control function blocks were introduced, providing means of modeling dependencies between evolution steps within the evolution control application. The final result is a complete engineering cycle for complex but clearly structured and downtimeless system evolution.



(a) Evolution graph



(b) Evolution control application

Fig. 4. The εCEDAC structured evolution approach

ACKNOWLEDGMENT

Special thanks go to Mario Semo for providing detailed information on the state-of-the-art of online change mechanisms.

REFERENCES

[1] B. Favre-Bulle and G. Zeichen, "Zukunft der Forschung in den Produktionswissenschaften," Verein zur Förderung der Modernisierung der Produktionstechnologien in Österreich - VPTÖ, Steyr – Gleink, Tech. Rep., 2006.

[2] European Commission, "MANUFUTURE A Vision for 2020," Report of the High-Level Group, Brussels, Tech. Rep., Nov. 2004.

[3] C. Szyperski, *Component Software, Beyond Object-Oriented Programming*, second edition ed. Great Britain: Addison-Wesley, 2002.

[4] IEC TC65/WG6, *IEC 61131-3: Programmable controllers - Part 3: Programming languages*, International Electrotechnical Commission IEC Std., Rev. 2.0, Jan. 2003.

[5] C. Sünder, A. Zoitl, and C. Dutzler, "Functional structure-based modelling of automation systems," accepted for publication in *International Journal of Manufacturing Research*, 2007.

[6] IEC TC65/WG6, *IEC 61499: Function Blocks for Industrial Process Measurement and Control Systems, Parts 1 – 4*, International Electrotechnical Commission IEC Std., Rev. 1.0, 2004/2005.

[7] logiCAD, "The IEC 61131 Technology Platform," kirchner SOFT GmbH. [Online]. Available: www.kirchnersoft.com

[8] MULTIPROG, "MULTIPROG®: Modern and Powerful IEC 61131 Programming," KW-Software GmbH. [Online]. Available: www.kw-software.com

[9] CoDeSys, "CoDeSys – General Overview," 3S - Smart Software Solutions GmbH. [Online]. Available: www.3s-software.com

[10] C. Sünder, A. Zoitl, B. Favre-Bulle, T. Strasser, H. Steininger, and S. Thomas, "Towards reconfiguration applications as basis for control system evolution in zero-downtime automation systems," in *Intelligent Production Machines and Systems, IPROMS 2006. IPROMS NoE Virtual International Conference on*, Jun. 3–14 2006.

[11] ISaGRAF, "ISaGRAF 5.0 Workbench Manual," ICS Triplex ISaGRAF Inc. [Online]. Available: www.isagraf.com

[12] G. Čengić, O. Ljungkrantz, and K. Åkesson, "Formal modeling of function block applications running in IEC 61499 execution runtime," in *Emerging Technologies and Factory Automation, 2006. ETFA 2006. Proceedings. 11th IEEE Conference on*, Prague, Czech Republic, Sep. 20–22 2006, pp. 1269–1276.

[13] J. H. Christensen, "Resources for the new generation of automation and control." [Online]. Available: www.holobloc.com, access date 21-12-2006

[14] ——, "IEC 61499 Compliance Profile for Feasibility Demonstration," Holobloc Inc. [Online]. Available: www.holobloc.com/doc/ita/index.htm

[15] H. Uhrmann, "Unterbrechungsfreie Rekonfiguration von IEC 61499-basierten Abtastregelkreisen im laufenden Betrieb," 2005.

[16] Z. Xiaokun, R. W. Brennan, X. Yuefei, and D. H. Norrie, "Runtime adaptability of a concurrent function block model for a real-time holonic controller," in *Systems, Man, and Cybernetics, 2001. Proceedings. IEEE International Conference on.*, Tucson, AZ, USA, Oct. 7–10 2001, pp. 164–168.

[17] S. Olsen, J. Wang, A. Ramirez-Serrano, and R. W. Brennan, "Contingencies-based reconfiguration of distributed factory automation," *Robotics and Computer-integrated Manufacturing*, vol. 21, no. 4–5, pp. 379–390, Aug.–Oct. 2005.

[18] K. Thramboulidis and A. Zoupas, "Real–time java in control and automation: A model driven development approach," in *Emerging Technologies and Factory Automation, 2005. ETFA 2005. Proceedings. 10th IEEE Conference on*, Catania, Italy, Sep. 19–22 2005, pp. 39–46.

[19] The εCEDAC Project, "Evolution control environment for distributed automation components." [Online]. Available: www.ecedac.org

[20] C. Sünder, B. Favre-Bulle, and V. Vyatkin, "Towards an approach for the verification of downtimeless system evolution," in *Emerging Technologies and Factory Automation, ETFA 2006. 11th IEEE International Conference on*, Prague, Czech Republic, Sep. 20–22 2006, pp. 1133–1136.

[21] The μCrons Project, "Micro-holons for next generation distributed automation and control." [Online]. Available: www.microns.org

[22] O. Hummer, T. Strasser, C. Sünder, A. Zoitl, M. N. Rooker, and G. Ebenhofer, "Towards zero-downtime evolution of distributed control applications via evolution control based on IEC 61499," in *Emerging Technologies and Factory Automation, 2006. ETFA 2006. Proceedings. 11th IEEE Conference on*, Prague, Czech Republic, Sep. 20–22 2006, pp. 1285–1292.