Please cite as follows: J. Yan, V. Vyatkin, "Distributed Software Architecture Enabling Peer-to-Peer Communicating Controllers", *IEEE Transactions on Industrial Informatics*, 9(4), 2013, pp. 2200-2209

Distributed Software Architecture Enabling Peer-to-Peer Communicating Controllers

Jeffrey Yan and Valeriy Vyatkin

Abstract— **This paper presents a novel model-driven software architecture for systems with high degree of redundancy and modularity of the equipment. The architecture is based on totally decentralized control. It combines adaptability and robustness of multi-agent control architectures with portability and interoperability benefits of IEC 61499 function block architecture. The architecture has been successfully proven feasible on a number of field trials, including modeling and implementation of a medium scale airport baggage handling control. Deployment was done on distributed networks consisting of configurations ranging from a few, to dozens of communicating control nodes. The work confirmed the ability to deliver similar functional characteristics as centralized systems but in a distributed implementation. Performance testing and development verified sufficient performance and software life cycle benefits.**

Index Terms—**Distributed Systems, Intelligent Control, IEC 61499**

I. Introduction

Modern automation systems become increasingly complex and the automation industry has rapidly progressed from focusing on mass production to mass customization. Many industrial systems are distributed over large physical areas. Applying traditional centralized development paradigms is becoming increasingly difficult. The programming paradigm of Programmable Logic Controllers (PLC), which are the industry standard for automation hardware, is of a very low level of abstraction. PLCs attempt to solve automation problems with a monolithic centralized control code which is not sufficiently scalable. For example, design failures that have been recently reported for several large airport projects, show the limits of the incremental PLC technology improvement and ask for a qualitative change.

Maintaining flexibility, adaptability and robustness using traditional control approaches becomes difficult as systems reach monolithic scales. Imbuing artificial intelligence into industrial control immediately creates a system that requires less user intervention but at the same time is highly scalable. Integrating intelligent agents into manufacturing [1] allows system components to renegotiate a production schedule based on current load, faults or external intervention. Logistics [2] and transport systems [3] benefit by dynamic re-routing and accurate tracking of product or vehicle conditions.

The solution proposed in this paper combines model-driven software engineering for distributed systems with elements of artificial intelligence, such as multi-agent control. Towards this end, this work presents a design methodology for integrating intelligence into highly modular industrial automation systems. The proposed design flow covers code architecture and design, deployment and some experimental testing. The fundamental architectural concept in the implementation relied on every mechatronic component in the system having its own software module. Functionality was handled using distributed algorithms and the result was simulated on a distributed, networked hardware test-bed of about 50 controllers. This paper extends that work by generalizing the design pattern and proposing an automated approach to the control design for highly modular distributed systems.

This paper is structured as follows. Section II gives a summary of the state of the art with some extensions from the previous work. Section III discusses the design guidelines and architecture for the development of distributed control of highly modular industrial systems. An example baggage handling system is used however the design methodology is kept as generic as possible. Section IV describes an initial experiment done using this architecture and the deployment hardware used. Some performance metrics are used to demonstrate the feasibility of the architecture.

II. Related Works

Distributed hardware and software architectures have the potential to bring about many benefits to industrial automation. Compared to centralized hardware configurations, distributed systems have the advantage of having inherent redundancy to failure.

New manufacturing paradigms such as mass customisation were targeted towards the last decade to respond to globalization and rapid changes in consumer demand. Surveys into the needs of future manufacturing systems show a definite demand for flexibility in order to handle mass customization efforts [4]. Flexible Manufacturing Systems (FMS) [5] which are often targeted at mid-volume and mid-variety production requirements, attempt to anticipate any required system changes in response to potential new products. However, developing contingencies into the application led to some undesirable resource under-utilization. Reconfigurable Manufacturing Systems (RMS) [6] attempt to improve upon

FMS by enforcing design principles including modularity, flexibility, scalability and interoperability. The modularity is the clue for RMS to scale rapidly in any desired increments, but it imposes new requirements to modular design of software.

The design of a fully reconfigurable system is still the area of intense research. Minimizing the effect of system reconfiguration at runtime is hugely sought after, as any level of downtime is always undesirable. Works such as implementing zero-downtime reconfiguration [7], sharing available resource capacity [8] and automating system recovery [9] provide differing design patterns for how downtime minimization can be realized using RMS concepts. However, integrating such patterns into the software life-cycle may be difficult for existing applications. Decision making for RMS can be orders of magnitude more difficult to develop compared to non-RMS. Thus, imbuing intelligence into the decision making process may reduce initial design difficulties.

A. Multi-Agent Systems

To facilitate design of control application for RMS, taking cues from object-oriented software and designing the application in a modular nature already speeds up system development and maintenance significantly. However, sometimes the production management, scheduling, maintenance and fault-handling grow so complex that even distributing functionality begins to become inadequate [1].

Multi-Agent Systems (MAS) [10] bring concepts inspired from the domain of artificial intelligence and apply them to automation. Rather than centralized control, or even distributed control with fixed logic, MAS deploy a multitude of agents over a distributed system. Each agent has only a local view and collaborates with other agents to achieve high level goals. Some promises of MAS include re-configurability [11], self-adaptation [12], self-organization [13] and fault-tolerance [14]. The reconfiguration process itself can be handled using agents, with the work in [15] utilizing a Reconfiguration Agent and Coordination Agent to ensure safe distributed reconfigurations. Using agents at the high-level control layer to reconfigure the low-level control has also been investigated [16].

Agent-based technologies have been investigated for reconfigurable manufacturing [1, 17, 18], process control [19], embedded systems [20] and transport systems [3, 21]. Integrating agents into a traditional control application is often highly domain specific, although some design patterns have been proposed for particular applications. The work in [22] proposed a two-level architecture separating agent based high-level behaviour from the low-level reactive behaviour in knowledge sharing applications. Another similar implementation of this executed the high-level agent functionality on a local PC [23]. When considering high performance applications, it may be useful to execute high level decisions on more powerful hardware.

Although MAS research has produced promising results, its adoption in industry is low. In the research domain, ad-hoc MAS implementations are common due to the variation of each application. However, many frameworks have been developed to ease MAS development. Current MAS frameworks such as JADE [24] and JACK [11] act as middleware. They facilitate MAS development through the provision of services such as self-organization, discovery, co-ordination, instantiation of agents, fault management and standardized communication. While works such as [2] explore handling agent co-ordination explicitly, frameworks aim to standardize these efforts. The resultant multi-agent systems then exhibit characteristics such as autonomy, local views and emergent behaviour in response to environment changes, but without any centralized control.

B. IEC 61499 and the Intelligent Mechatronic Component architecture

The IEC 61499 standard [25] provides a reference architecture for the next generation of distributed automation systems that complements the centralized programming architecture of the IEC 61131-3 standard. There are commercial IEC 61499 compliant integrated development environments (IDE) such as ISaGRAF [26] and NxtStudio [27] that have been used in some investigative works [28]. There are also mature tools developed in research and academic communities, such as 4DIAC, FBDK and BlokIDE. IEC 61499 was conceived to provide an adequate implementation platform, bridging the gap between new generation of distributed automation systems (such as RMS and MAS), and the existing automation architectures. Works such as [20] and [29] attempt to leverage the modular nature of IEC 61499 for reconfiguration purposes. However, the IEC 61499 standard can be interpreted in a number of ways resulting in differing execution semantics per-platform [30] which should be taken into account during development.

Current MAS frameworks are often insufficient for many applications due to their real-time requirements. For these, timely reaction and sensing is critical and [31] implements IEC 61499 based real-time applications in parallel with unconstrained applications. The concept of time in [32] was used as a mechanism to deliver synchronous actions over a distributed, networked system.

The concept of an Intelligent Mechatronic Component (IMC) was first proposed in [33] and thereafter a proposal for its implementation in IEC 61499 was made in [34]. The IMC concept describes a notion of intelligent physical components (machines or parts thereof) that come pre-packaged with software modules such as control programs, plant simulation modules and potentially human-machine interface (HMI) software components and applications. It allows hardware components to encapsulate functionality into re-usable, portable units that can be distributed along with the hardware. These units can then be assembled into functional systems. The modular nature of IEC 61499 was shown to support this concept well.

IMC developers can take advantage of the earlier developed PLC code using migration methods [35, 36] which enable automatic translation from an IEC 61131-3 control program into IEC 61499. This makes IMC and IEC 61499 an attractive solution for preserving existing hardware investments while trying state of the art distributed design. Furthermore formal verification of IMC based systems has also been explored [37], which can provide additional assurance of functional correctness for developers.

III. Design Architecture for Distributed Control

This work is notably different from other multi-agent control implementations in several aspects. Firstly rather than relying on existing MAS frameworks, this work will map agent-like functionality directly into the IEC 61499 architecture. The term agent is used here to represent distributed intelligence functions such as self-organization and fault management. Not all features of MAS frameworks are implemented such as assisted agent instantiation and standardized communications protocols. However, the authors believe self-organization, decentralization and autonomy and other features represent MAS adequately.

The IEC 61499 architecture improves life-cycle management of the application which is essential for industry adoption, but often not in the focus of traditional MAS approaches. Similar modularity and distribution can be implemented in IEC 61131 applications depending on the chosen design paradigm. However, integration effort may become a concern when attempting to aggregate separate applications. This is significantly mitigated by IEC 61499 due to event-driven execution semantics, standardized communications protocols, inherent modularity and obligatory elimination of global variable usage.

The usage of IEC 61499 not only synergizes well with MAS but could also integrate with other architectures including IEC 61131 and Service Oriented Architectures (SoA) [38]. IEC 61499 can provide a suitable system level architecture for the description of SoA based systems. FBs could represent services and FB compositions and connections could describe the relationships between these services. The SoA paradigm also has well standardized service discovery and description mechanisms which are lacking in MAS platforms. A mixed system of SoA, MAS alongside traditional IEC 61499 automation applications could utilize the strengths of each paradigm and also be unified under an IEC 61499 system level description.

The work in [23] describes the execution of agent functionality in a local PC and low level control in PLCs. There are increasing trends depicting intelligent components bundled with some lightweight computational units [39]. An example of these are the “intelligent” motor drives which include a separate microcontroller in addition to the one implementing motion control, such as the Eaton Distributed Electronic Drive line of products [40]. This work maps agent functionality (both high and low-level control) directly onto distributed hardware, leveraging this shift from centralized computation to distributed lightweight nodes. Since this hardware is markedly different from a simple network of distributed PLCs, each intelligent computational node will be referred to as a micro-PLC.

A. Architecture

The architecture is based on the IMC concept, structured internally following the Model-View-Controller (MVC) design pattern [41]. The building blocks of the architecture are IMCs integrating controllers and simulation models. Software designed according to this architecture is highly portable, thus can be deployed to a variety of hardware topologies, with one extreme being a central controller and the other extreme being micro-PLCs embedded into every component. Baggage handling systems (BHS) will be used as an example application for this work and previous investigations have yielded promising results in the domain of distributed BHS.
An example is the BHS shown in Figure 1.

The plant is composed of a set of mechatronic components (conveyor sections, x-ray machines, diverters, tag readers, etc.), denoted by M of which the granularity is a reasonable assumption of modularity.

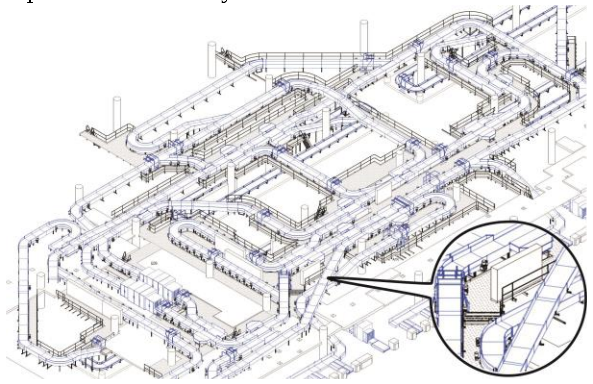

Figure 1 - CAD drawing of an example baggage handling system composed of a multitude of conveyor sections.

Each component $\gamma \epsilon M$ has some corresponding software sub-components $S(\gamma) = \{sim, hmi, view, ctl\}$ following the assumptions of the IMC architecture [33][1]:

1. $\gamma(sim)$: This is an accurate simulation model of the dynamics of the component and is assumed to be provided by the equipment vendor. For testing purposes this could be used in closed-loop simulation with the control. For relatively lightweight simulations, deployment to micro-PLC hardware may be possible. However, if performance is an issue then PC based simulation with I/O cards could be used. [42] proposes a framework for a systematic approach for the composition of simulation and control in a distributed system of intelligent mechatronic objects. Custom simulations could be developed and deployed to real-time simulator hardware; however closed-loop testing will be limited by the complexity of the simulation.
2. $\gamma(hmi)$: This is the human-machine-interface (HMI) that provides interfacing to the component and is analogous to the HMI of Supervisory Control and Data Acquisition (SCADA) systems. The HMI is connected in closed-loop with $\gamma(ctl)$ and facilitates user interaction with the control application.

[1] Here a simplification of IMC architecture with only four functional domains is used.

3. $\gamma(view)$: This is a visualization of the dynamics of the simulation model and would be provided by the vendor. Since the simulation model is executing in closed-loop with the controller, a live view of the component dynamics can be viewed at runtime. Due to the lack of simulation in many SCADA based systems this live view is usually not present.
4. $\gamma(ctl)$: The controller of the component. This can be of various degrees of complexity. The component's vendor can provide at least a controller implementing certain basic operations or services, invoked by a higher level controller, or, in some cases, a more sophisticated software agent capable of self-organization with other such components. The controller code can eventually be executed on the embedded micro-PLC or on other PLC having information access to the component.

The ability to deliver a higher level controller as a part of the $S(\gamma)$ certainly adds value to the whole component. In this work, it is investigated how the controller can implement the agent functionality that collaborates with other components to achieve certain goals. Holonic Manufacturing Systems research often describes the decomposition of control into low-level and high-level [43]. Similarly in this work, the controller $\gamma(ctl)$ is further decomposed into these two distinct elements:

1. $\gamma(ctl - LLC)$: The low-level control (LLC) which governs the reading of sensors and the triggering of actuators. The LLC should abstract out details of I/O and other low-level behaviour such as initialization routines, fault detection and PID functions. LLC should also be self-contained and only communicate with the component it is assigned to.
2. $\gamma(ctl - HLC)$: The high-level control (HLC) accesses the abstract interface provided by the LLC and provides agent-like behaviour and thus the HLC will be referred to as the agent for a particular component. The agent should communicate with other agents to fulfil intended goals.

LLC is executed on local hardware. HLC implementations can vary although standards such as FIPA [44] exist to encourage interoperable communication. However, HLC is usually executed on the local PC with an interface to the controller hardware. This work instead deploys agent functionality directly onto controller hardware alongside LLC modules aiming at more integral and cost-effective solution. Furthermore, agent behaviours may be better specified in distinct sub-modules. These behaviours could be routing, fault management and isolation, tracking and order handling.

While the plant contains a group of components, there are also many relationships between these components. Details such as physical topology, electrical wiring, material flow, dependencies and redundancies are typically captured using some form of design document. This methodology will mainly focus on the physical relationships between components. Topology can be expressed in many forms, however for simplicity the plant configuration will be described as an attributed directed graph $G = \{M, E, A, F_M, F_E\}$, where the set of components M encompasses the nodes of the graph and $E \subseteq M \times M$ denote the edges of the graph, where these edges describe the topological relationships between components, A is a set of attributes, $F_M: M \rightarrow A$ is an assignment of attributes to nodes and $F_E: E \rightarrow A$ is assignment of attributes to the edges. The edges of the graph could represent such relations as upstream, downstream, merging into, diverting from and other details. Each arc $\delta \in E$ can be described by two elements $\delta = \{pred, succ\}$:

1. $pred(\delta)$: The tail of the arc, referred to here as the predecessor. This could be an upstream conveyor, or an upstream processing station or any component that precedes the connection.
2. $succ(\delta)$: The head of the arc, referred to here as the successor. This could be a downstream conveyor, or a component that the current component belongs to, such as a sensor belonging to a conveyor.

Representing this information as a graph opens up the possibility of using a variety of graph transformation techniques [45, 46] to assist in automatic software generation.

B. *Standardized agent composition and connection*

Specific inter-component and intra-component communication is defined by the tuple $C = \langle R, Q \rangle$. $R \subseteq S(\gamma) \times S(\gamma)$ that specifies the intra-component communication between software sub-components $S(\gamma) = \{sim, hmi, view, ctl\}$ within a single component γ. $Q \subseteq S(pred(\delta)) \times S(succ(\delta))$ specifies inter-component communication between sub-components between the predecessor and successor components defined by each arc δ. Intra-component communication for every component $\gamma \in M$ is shown in Figure 2 and follows the IMC pattern. The set of relations defining communication is expressed by the set R:

1. $R(\gamma(sim), \gamma(ctl))$ and $R(\gamma(ctl), \gamma(sim))$: Bidirectional communication between the simulation and control.
2. $R(\gamma(sim), \gamma(view))$: Unidirectional communication between simulation and view representing state data from the simulation passed to the view for rendering.
3. $R(\gamma(ctl), \gamma(hmi))$ and $R(\gamma(hmi), \gamma(ctl))$: Bidirectional communication between control and HMI representing control panel interfacing with the controller application.

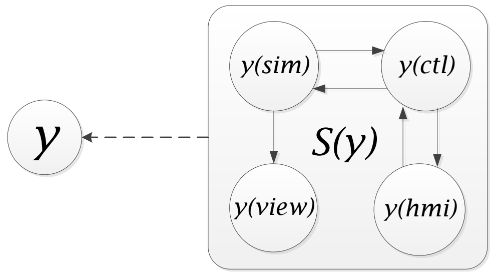

Figure 2 - Communication between software elements within a single component.

Inter-component communication between individual software sub-components is defined by each topological arc $\delta \in E$ and can be described by the set Q:

1. $Q(pred(\delta)(ctl), succ(\delta)(ctl))$ and $Q(succ(\delta)(ctl), pred(\delta)(ctl))$: Controller to controller

communication between two components, usually accompanied by communication in the opposite direction resulting in bidirectional communication between controllers.

2. $Q(pred(\delta)(sim), succ(\delta)(sim))$ and $Q(succ(\delta)(sim), pred(\delta)(sim))$: Simulation model communication used to pass model based data between components.

Communication between two components c_i and c_{i+1} is shown in Figure 3. An arc connects these two components and is attributed with the *downstream* relationship. Bidirectional communication would then be initiated with a corresponding *upstream* relationship. Additionally, the controller element of each component is then subdivided further into the LLC and agent (HLC) sub-modules described earlier.

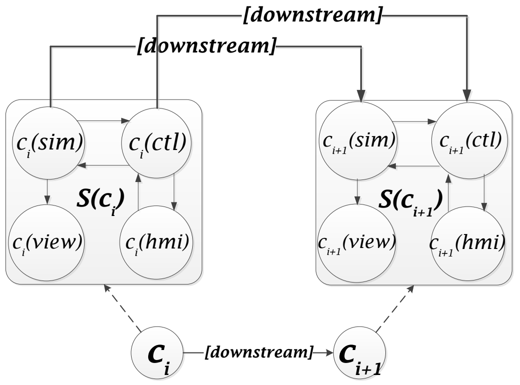

Figure 3 - Communication between two components with a downstream topological relationship.

System composition in this manner ensures reachability and correctness of construction. Since each communication link is an abstraction for data flow, distributed algorithms become ideal for intelligent behaviour implementation such as fault isolation, routing and discovery. To achieve equivalent functionality compared to centralized implementations, the agent module should implement a variety of distributed algorithms rather than rely on centralized data.

One of the important functions for a conveyor agent is handling transport; transportation decision making is in addition highly related to routing. Figure 4 depicts a conveyor module c_i with upstream, downstream and merging relations to adjacent conveyors. This is then decomposed into sub-modules with the control $c(ctl)$ decomposed further into LLC and HLC (agent). Routing is integrated into the HLC as a separate module, receiving data from the downstream HLC while passing data to upstream HLCs. Routing can be handled using any of a number of distributed routing algorithms such as Bellman-Ford [47]. Routing tables would be stored as local data in the routing sub-module and distance vectors can be passed to and from adjacent HLCs.

Attached components also affect the composition of the agent sub-modules. These components could be diverters, encoders or x-rays with distinct functions. Modules connected directly to I/O can be instantiated in the LLC depending on the number of physical modules, such as the diverters in Figure 4. Routing decisions are passed to LLC from HLC to divert bags on the correct route.

Another function useful for plug-and-play composition of software components is the use of distributed mutual exclusion algorithms (e.g. Ricart-Agrawala [48]) to avoid bags collision in the point of baggage flow merge. These mutual exclusion algorithms could be implemented in the same HLC module along with routing by separating the request and release behaviours into two communicating sub-modules with local data as shown in Figure 4. Merging priorities can then be negotiated with adjacent conveyors as per Ricart-Agrawala to determine the current merging situation.

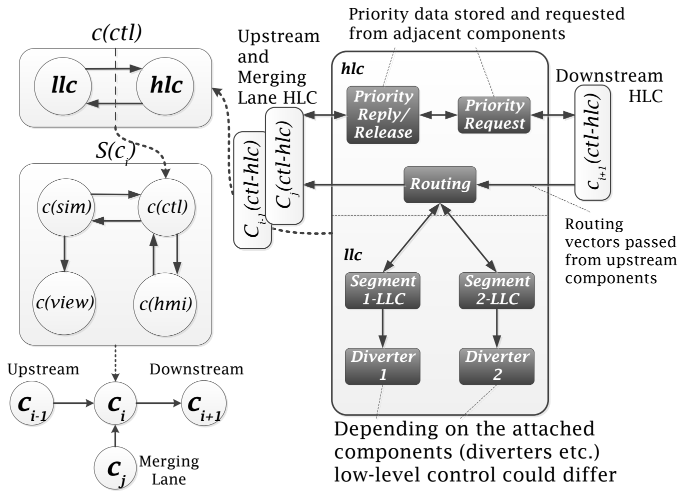

Figure 4 - Architecture of the controller FB for a single conveyor segment.

C. *Implementation with IEC 61499 reference architecture*

The IEC 61499 architecture suits this methodology for a number of reasons compared to IEC 61131-3. The FB concept of IEC 61499 encourages encapsulation of functions into standalone processes or composite assemblies thereof. Software tools supporting IEC 61499 can facilitate distribution of such software components to communicating micro-PLCs in a seamless way. Application development can be done from a high-level perspective rather than having to handle the low-level configuration details of a network of controllers. The implementation of both high-level and low-level functions using IEC 61499 is more hardware compliant compared to executing high-level functionality on proprietary platforms. In future distributed systems, there will be a great diversity in the types of distributed intelligent applications and the unification of both HLC and LLC into a single reference architecture such as IEC 61499 may assist in standardization efforts [49]. The event-driven nature of IEC 61499 suits the message passing communication paradigm that is implied by a network of distributed agents. Since IEC 61499 provides an executable block-diagram language, it allows for execution of the agents directly on micro-PLCs. Finally, an additional benefit in using IEC 61499 is that formal verification techniques can be used [50, 51]. These increase dependability and may reduce testing effort, however, as with simulation the usefulness of formal verification will also depend on the complexity of the test cases.

The IEC 61499 platform used in this work was ISaGRAF by ICS Triplex. This environment was one of the two commercial tools supporting IEC 61499 at the moment when this research started. As compared to another environment, NxtStudio by NxtControl, ISaGRAF had a better combination of features for this project. One such feature is support of user defined data types (UDTs) that is particularly useful for agent-based applications, as they can be data intensive. ISaGRAF fully supports UDT definitions not only in the IDE, but also on both the software and hardware runtimes. ISaGRAF automatically inserts communication between devices if an application is distributed over a network and their implementation has been shown to scale well [52]. In addition to network scale, the process of wiring large distributed applications with I/O is highly streamlined.

Backwards compatibility with PLC style of programming is another benefit of ISaGRAF. Adoption of IEC 61499 is partly hindered by reluctance for investment in new hardware; the ability to execute state of the art IEC 61499 distributed applications on existing hardware can be appealing, easing migration concerns.

The ISaGRAF IEC 61499 runtime is cyclic and is built on top of their existing IEC 61131-3 framework. Within a device, events are recorded as integer values and FBs trigger based on changes to these values during each cyclic scan. Between devices, events are implemented as UDP messages, minimising network utilization. Although some events are implemented in a cyclic approach, it is still fully in line with IEC 61499.

A comparison between ISaGRAF and FBDK implementations of IEC 61499 revealed some semantic and syntactic differences but did not conclude that either implementation was superior [53]. The constant CPU load under the ISaGRAF cyclic paradigm can even yield relatively predictable performance characteristics compared to more sporadic event-driven implementations.

D. Distributed functions using ISaGRAF IEC 61499 mechanisms

Internally, the controller FB is further separated into LLC and agent based control. Since this is a conveyor controller, a significant amount of the agent functionality will be for handling transport of baggage. For the routing sub-module in Figure 4, the internal functionality was implemented using a modified Distributed Bellman-Ford (DBF) algorithm. While other routing algorithms can be used such as link-state based Dijkstra algorithm [47], DBF only maintains routing information about its neighbours to be able to route baggage using shortest path. In a typical DBF implementation all nodes are of interest however for a BHS application, a limited number of nodes in the system are of interest. These are the final bag destinations known as bag-sinks, although other intermediate destinations such as x-rays, tag readers and manual security checks should also have topological information stored. DBF will have a shorter convergence time for smaller applications compared to Dijkstra algorithm and is useful due to the significant reduction in the number of nodes.

Networked communication can be significantly reduced to account for this change; rather than every node in the system passing data about every other node, only points of interest within the BHS need to be included. Each agent must store a local routing table with distance metrics. The routing tables consist of a 2-dimensional array C with dimensions m x n. Array size is selected based on a predicted number of destinations (n) and maximum number of exit interfaces per conveyor (m).

- i is the index of the destination and these are indexed from 0,1,2,3...,n-1 ;
- j is the divert exit interface. These are indexed from 0,1,2,3...,m-1;
- C[i, j] is the distance metric to destination(i) when diverting to exit-interface(j).

The support for nested arrays in ISaGRAF is ideal for modeling this data, compared to other IEC 61499 solutions which only support one-dimensional arrays. Figure 5a) depicts a scenario where two adjacent conveyors are preparing to share routing data. The upstream conveyor has an empty routing table. The downstream conveyor routing module iterates through its routing table and finds the fastest routing path denoted by the cheapest metric for each destination index. This information is stored in a one-dimensional vector. Figure 5b depicts the upstream conveyor receiving the routing vector and appending it to its table, adding its own distance data to each metric.

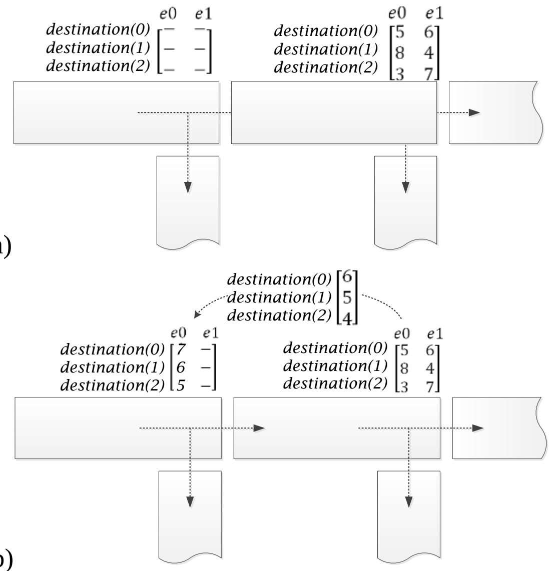

Figure 5 - Modified DBF algorithm transmitting routing vectors based on destination bag-sinks only.

For every bag in the BHS, a bag-record object can be used to represent data such as destination, priority and security level clearance. The User Defined Types (UDT) mechanism in ISaGRAF is ideal for these complex data structures. A table storing bag-record objects for all bags on the current conveyor should be stored within a tracking FB which can be part of a composite agent FB. This FB can query the routing table to determine bag diversion eligibility. Intermediate destinations such as x-rays can then modify this bag-record based on a scan result, possibly incrementing the security clearance and re-determining the next destination.

Fault handling was based on a fault-model with two types of faults:

- Mechanical faults were detected by the local controller and fault data propagated to upstream components. Upstream components update routing tables and further propagate these changes.
- Electrical faults such as the failure of a controller were detected by adjacent upstream controllers. Upstream controllers poll downstream controllers regularly to determine status. Any faults are propagated upstream.

This fault information can be inserted into routing tables as non-desirable values and using DBF, the system can then route baggage around faults occurring during runtime. Figure 7 depicts a scenario during experimentation when a fault occurred in a BHS layout. In this BHS configuration, conveyor c115 has failed due to a simulated electrical fault. Fault data is then propagated upstream to reroute baggage during system operation using alternative path (c020, c209).

ISaGRAF applications can be accessed via OLE for Process Control (OPC) which is commonly used in industrial automation to read and write data within hardware. Socket communication FBs utilizing UDP or TCP protocols could be used as an alternative to OPC. Additionally, another alternative is the built in HMI functionality of NxtStudio. This allows data from the FB application to connect to HMI FBs, which can be consumed in NxtStudio developed HMI. Both these alternatives require data to exist explicitly on FB interfaces and FBs to be instantiated whenever new data is required. As systems scale in size, OPC allows the visualization to simply connect to new data without any modification to the control application.

E. Verification by Simulation

Along with facilitation of the system design, the IMC architecture in combination with IEC 61499 dramatically reduces time for development of a "Software in the Loop" (SiL) simulation model. Simulation models can be developed in IEC 61499 or an external environment such as MATLAB as discussed in detail in [54], where migration of models is also discussed. If some of the simulation models are vendor provided, then model development time is further reduced. These closed-loop simulations could be either executed on high-speed hardware, or directly deployed to control hardware. Combined with PC based I/O solutions such as EtherCAT cards, closed-loop simulation using IEC 61499 models becomes attractive due to its low cost of entry.

Due to the entire application being IEC 61499 based, formal verification techniques can further extend the scope of testing beyond closed-loop simulations. Another benefit of having the simulation model based on IEC 61499 is discussed in Hirsch *et al*. [55] which describes that the usage of a simulation model for predictive control may increase reliability and ease of diagnostics.

Initially SiL testing was used with both the simulation model and control executing on a local PC. Due to the executable nature of the IEC 61499 applications, once the developer is satisfied with performance the same control application can be directly deployed to hardware. Hardware in the Loop (HiL) testing then deploys the control to a single hardware controller such as a PLC and runs in closed loop with the simulation model on a local PC. Finally, for a distributed system, distributed HiL testing deploys the control application across a network of multiple controllers. Benefits of each phase of testing are as follows:

1. *SiL*: Allows the developer to test the control application against a simulation of the plant rather than predefined test scripts. Simulation provides accurate dynamics and can test more scenarios compared to predefined scripts.
2. *HiL*: Once SiL testing is complete, HiL testing allows testing the execution performance of the control application on the hardware controller. I/O performance can then be tested between control and simulation.
3. *Distributed HiL*: This gives an indication of the network performance for the control application. Metrics such as reaction times, delays and network utilization can be tested.

IV. Experiments

With IEC 61499 and ISaGRAF, a highly distributed application can be designed from a high level, the corresponding FBs can be mapped onto devices and communication will be inserted automatically. For these experiments, usually a single composite FB containing both HLC and LLC would be mapped per device. If distribution of these functions were desired, IEC 61499 provides a sub-application mechanism as an additional feature to manage granular deployment details. depicts a network of embedded controllers for which the configuration was deployed to.

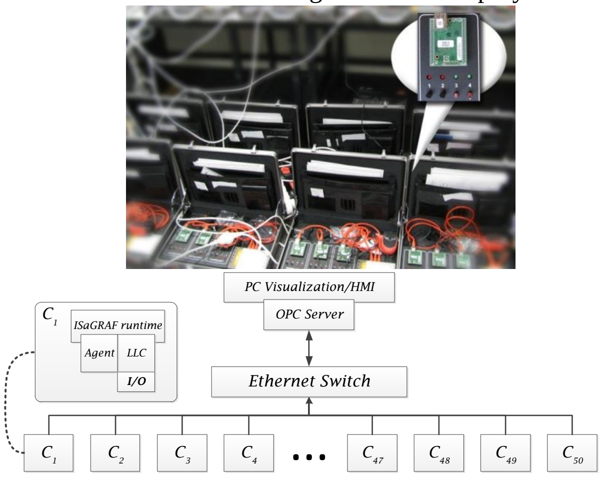

Figure 6 – Architecture and image of 50 embedded control nodes running the ISaGRAF IEC 61499 runtime used for deployment of BHS control application.

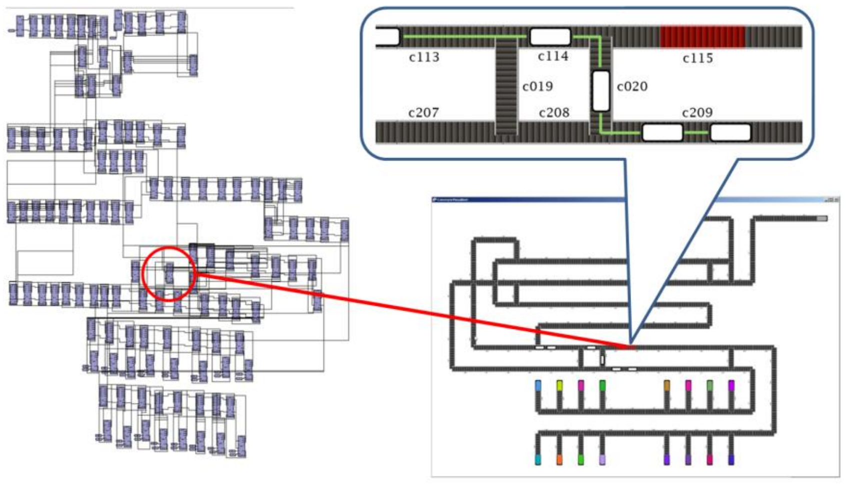

Figure 7 - Experiments conducted on a BHS configuration depicting routing bags around a fault.

BHS system with 53 conveyor FBs distributed over 27 NetBurner controllers (approximately two conveyor FBs per controller) was used for network utilisation measurements. A sample of three controllers in the system showed that utilization was less than 0.1% and was far from saturating the bandwidth of 100Mbit/s Ethernet.

Similar to PLC based architectures, the ISaGRAF IEC 61499 runtime is cyclic and cycle times can be a good indicator of performance. Table 1 shows cycle times in a small scale system with only three conveyor nodes distributed over three controllers and the 53 conveyor node system distributed over 27 controllers. Results were obtained using the cycle time measurement functionality of the ISaGRAF runtime.

Table 1 - Individual Controller PLC Cycle Times when running only 3 controllers compared to a system running 27 controllers.

	Controller 1	Controller 2	Controller 3
3 Controllers	5.36ms	5.71ms	5.54ms
27 Controllers	5.51ms	5.64ms	5.72ms

Results show not adverse impact on cycle times when scaling the system up. The maximum dimension of BHS system tested in our design framework so far includes approximately 100 conveyors and their control distributed across 48 hardware control nodes. ISaGRAF provides an option for fixed cycle times with a flag indicting cycle time overflow which can be useful for testing some time critical applications. Since agent functionality is developed in IEC 61499, if hard real-time performance is desired over ISaGRAF features, options such as the synchronous compiler for IEC 61499 and static timing analysis [56] can be used.

Another metric of the viability was deemed to be the re-usability of software components. Developing a small system was a matter of connecting IEC 61499 FBs in a manner that mirrored the system topology. To extend this to a larger system only involved adding more FBs depending on corresponding new components in the system. Due to the use of ISaGRAF as the IDE, communication between hardware controllers was generated automatically at the deployment stage and an OPC server configured for the new system without any user intervention.

Performance metrics such as cycle times and network utilization were satisfactory and resulted in correctly routed baggage and execution of the BHS. System characteristics such as autonomous local fault handling, minimal setup time per topology and distributed reproduction of centralized functions represented agent behaviour adequately.

V. Conclusions

This work proposed an architecture to facilitate the implementation of the smart machine which implies distributed automation hardware. The solution can be regarded as agent-based due to its use of distributed intelligence, self-organization and absence of centralized control. Agent behaviour and peer-to-peer communication pattern was custom developed for the application based on the physical topology of the system using IEC 61499 rather than relying on existing MAS frameworks. This had the benefit of being lightweight and directly executable on the micro-PLC hardware. Testing was done to confirm the feasibility of the approach and the resultant network utilization levels and scan-cycle times measured. The usage of the IEC 61499 standard addresses the automation software lifecycle concerns and provides a supporting architecture from software development to commercial hardware deployment.

This architecture is aimed at highly modular logistics systems. However sectors such as energy which typically include high levels of physical dispersion, show promising results when applying intelligent, distributed control [57]. Applications whose functionality is less determined by the physical topology of the plant, such as process control or motion control may be less appropriate for this architecture. Future extensions could integrate formal verification techniques as well as simulation for testing. Works already exist on formal verification IEC 61499 applications and it would be ideal to integrate this architecture with the formal verification solutions. Formal verification could then be used to test for scenarios where simulation would not uncover. Interoperability with MAS frameworks could also be investigated such as implementation of the FIPA protocol. This could be integrated as a set of FIPA communication function blocks based on the IEC 61499 standard.

VI. References

[1] P. Leitão, "Agent-based distributed manufacturing control: A state-of-the-art survey," *Engineering Applications of Artificial Intelligence,* vol. 22, pp. 979-991, 2009.

[2] C. Gang, Y. Zhonghua, and L. Chor Ping, "Coordinating Agents in Shop Floor Environments From a Dynamic Systems Perspective," *Industrial Informatics, IEEE Transactions on,* vol. 2, pp. 269-280, 2006.

[3] D. Herrero-Perez and H. Martinez-Barbera, "Modeling Distributed Transportation Systems Composed of Flexible Automated Guided Vehicles in Flexible Manufacturing Systems," *Industrial Informatics, IEEE Transactions on,* vol. 6, pp. 166-180, 2010.

[4] M. G. Mehrabi, A. G. Ulsoy, Y. Koren, and P. Heytler, "Trends and perspectives in flexible and reconfigurable manufacturing systems," *Journal of Intelligent Manufacturing,* vol. 13, pp. 135-146, 2002.

[5] H. ElMaraghy, "Flexible and reconfigurable manufacturing systems paradigms," *International Journal of Flexible Manufacturing Systems,* vol. 17, pp. 261-276, 2005.

[6] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy, and H. Van Brussel, "Reconfigurable Manufacturing Systems," *CIRP Annals - Manufacturing Technology,* vol. 48, pp. 527-540, 1999.

[7] 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," presented at the IPROMS NoE Virtual International Conference, 2006.

[8] W. Shengyong, C. Song Foh, and M. A. Lawley, "Using Shared-Resource Capacity for Robust Control of Failure-Prone Manufacturing Systems," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on,* vol. 38, pp. 605-627, 2008.

[9] T. Strasser and R. Froschauer, "Autonomous Application Recovery in Distributed Intelligent Automation and Control Systems," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* vol. PP, pp. 1-17, 2012.

[10] M. N. Huhns and L. M. Stephens, *Multiagent systems: a modern approach to distributed artificial intelligence*. Cambridge, Massachusetts: MIT Press, 2000.

[11] V. Marik and D. McFarlane, "Industrial adoption of agent-based technologies," *Intelligent Systems, IEEE,* vol. 20, pp. 27-35, 2005.

[12] D. Weyns and M. Georgeff, "Self-Adaptation Using Multiagent Systems," *Software, IEEE,* vol. 27, pp. 86-91, 2010.

[13] R. Frei and G. Di Marzo Serugendo, "Self-Organizing Assembly Systems," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* vol. 41, pp. 885-897, 2011.

[14] P. Tichý and R. Staron, "Multi-Agent Technology for Fault Tolerant and Flexible Control." vol. 310, D. Srinivasan and L. Jain, Eds., Innovations in Multi-Agent Systems and Applications - 1 ed: Springer Berlin / Heidelberg, 2010, pp. 223-246.

[15] M. Khalgui and H. M. Hanisch, "Reconfiguration Protocol for Multi-Agent Control Software Architectures," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* vol. 41, pp. 70-80, 2011.

[16] W. Lepuschitz, A. Zoitl, Valle, x, M. e, and M. Merdan, "Toward Self-Reconfiguration of Manufacturing Systems Using Automation Agents," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* vol. 41, pp. 52-69, 2011.

[17] O. J. L. Orozco and J. L. M. Lastra, "Agent-based control model for reconfigurable manufacturing systems," in *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on*, 2007, pp. 1233-1238.

[18] M. F. Zaeh and M. Ostgathe, "A multi-agent-supported, product-based production control," in *Control and Automation, 2009. ICCA 2009. IEEE International Conference on*, 2009, pp. 2376-2383.

[19] M. Metzger and G. Polakow, "A Survey on Applications of Agent Technology in Industrial Process Control," *Industrial Informatics, IEEE Transactions on,* vol. 7, pp. 570-581, 2011.

[20] M. Khalgui, O. Mosbahi, and H. M. Hanisch, "Implementation of agent-based reconfigurable embedded control systems," in *Industrial Informatics, 2009. INDIN 2009. 7th IEEE International Conference on*, 2009, pp. 428-433.

[21] C. Bo and H. H. Cheng, "A Review of the Applications of Agent Technology in Traffic and Transportation Systems," *Intelligent Transportation Systems, IEEE Transactions on,* vol. 11, pp. 485-497, 2010.

[22] J. P. Soto, A. Vizcaíno, J. Portillo-Rodríguez, M. Piattini, and O. Kusche, "A Two-Layer Multi-agent Architecture to Facilitate Knowledge Sharing within Communities of Practice," *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial,* pp. 46-54, 2009.

[23] I. Hegny, O. Hummer, A. Zoitl, G. Koppensteiner, and M. Merdan, "Integrating software agents and IEC 61499 realtime control for reconfigurable distributed manufacturing systems," in *Industrial Embedded Systems, 2008. SIES 2008. International Symposium on*, 2008, pp. 249-252.

[24] (2011). *Java Agent DEvelopment Framework*. Available: http://jade.tilab.com/

[25] J. H. Christensen, "Design patterns for systems engineering in IEC 61499," presented at the Verteilte Automatisierung - Modelle und Methoden für Entwurf, Verifikation, Engineering und Instrumentierung, Otto-von-Guericke-Universität Magdeburg, Germany, 2000.

[26] (2011). *ICS Triplex ISaGRAF (6.0 ed.)*. Available: http://www.isagraf.com

[27] (2011). *NXTControl NXTStudio (1.4 ed.)*. Available: www.nxtcontrol.com

[28] V. Vyatkin, "IEC 61499 as Enabler of Distributed and Intelligent Automation: State-of-the-Art Review," *Industrial Informatics, IEEE Transactions on,* vol. 7, pp. 768-781, 2011.

[29] T. Strasser, I. Muller, C. Sunder, O. Hummer, and H. Uhrmann, "Modeling of Reconfiguration Control Applications based on the IEC 61499 Reference Model for Industrial Process Measurement and Control Systems," in *Distributed Intelligent Systems: Collective Intelligence and Its Applications, 2006. DIS 2006. IEEE Workshop on*, 2006, pp. 127-132.

[30] T. Strasser, A. Zoitl, J. H. Christensen, Su, x, and C. nder, "Design and Execution Issues in IEC 61499 Distributed Automation and Control Systems," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* vol. 41, pp. 41-51, 2011.

[31] A. Zoitl, G. Grabmair, F. Auinger, and C. Sunder, "Executing real-time constrained control applications modelled in IEC 61499 with respect to dynamic reconfiguration," in *Industrial Informatics, 2005. INDIN '05. 2005 3rd IEEE International Conference on*, 2005, pp. 62-67.

[32] C. Pang, J. Yan, V. Vyatkin, and S. Jennings, "Distributed IEC 61499 material handling control based on time synchronization with IEEE 1588," in *Precision Clock Synchronization for Measurement Control and Communication (ISPCS), 2011 International IEEE Symposium on*, 2011, pp. 126-131.

[33] V. Vyatkin, "Intelligent mechatronic components: control system engineering using an open distributed architecture," in *Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference*, 2003, pp. 277-284 vol.2.

[34] C. Pang and V. Vyatkin, "IEC 61499 function block implementation of Intelligent Mechatronic Component," in *Industrial Informatics (INDIN), 2010 8th IEEE International Conference on*, 2010, pp. 1124-1129.

[35] T. Hussain and G. Frey, "Migration of a PLC Controller to an IEC 61499 Compliant Distributed Control System: Hands-on Experiences," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, 2005, pp. 3984-3989.

[36] W. Dai and V. Vyatkin, "Ontology Model for Migration from IEC 61131-3 PLC to IEC 61499 Function Block," in *Electronic Design, Test and Application (DELTA), 2011 Sixth IEEE International Symposium on*, 2011, pp. 172-175.

[37] S. Patil, V. Vyatkin, and M. Sorouri, "Formal verification of Intelligent Mechatronic Systems with decentralized control logic," in *Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference on*, 2012, pp. 1-7.

[38] W. Dai and V. Vyatkin, "On migration from PLCs to IEC 61499: Addressing the data handling issues," in *Industrial Informatics (INDIN), 2010 8th IEEE International Conference on*, 2010, pp. 1142-1147.

[39] V. Vyatkin, Z. Salcic, P. S. Roop, and J. Fitzgerald, "Information Infrastructure of Intelligent Machines based on IEC61499 Architecture," *Industrial Electronics Magazine, IEEE,* vol. 1, pp. 17-29, 2007.

[40] C. Batini, E. Nardelli, and R. Tamassia, "A layout algorithm for data flow diagrams," *Journal Name: IEEE Trans. Software Eng.; (United States); Journal Volume: 12:4,* pp. Medium: X; Size: Pages: 538-546, 1986.

[41] E. Curry and P. Grace, "Flexible Self-Management Using the Model-View-Controller Pattern," *Software, IEEE,* vol. 25, pp. 84-90, 2008.

[42] V. Vyatkin, H. M. Hanisch, P. Cheng, and Y. Chia-Han, "Closed-Loop Modeling in Future Automation System Engineering and Validation," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* vol. 39, pp. 17-28, 2009.

[43] J. H. Christensen, "HMS/FB Architecture and its Implementation," *Agent-Based Manufacturing: Advances in the Holonic Approach,* pp. 53–87, 2003.

[44] V. Marik, P. M., P. Vrba, and V. Hrdonka, "FIPA standards and Holonic Manufacturing," *Agent-Based Manufacturing: Advances in the Holonic Approach,* 2003.

[45] U. Ryssel, J. Ploennigs, and K. Kabitzsch, "Generative function block design and composition," in *Factory Communication Systems, 2006 IEEE International Workshop on*, 2006, p. 253.

[46] C. Huijs, "A graph rewriting approach for transformational design of digital systems," in *EUROMICRO 96. 'Beyond 2000: Hardware and Software Design Strategies'., Proceedings of the 22nd EUROMICRO Conference*, 1996, pp. 177-184.

[47] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, Third ed.: MIT Press and McGraw-Hill, 2009.

[48] M. Maekawa, A. Oldehoeft, and R. Oldehoeft, *Operating Systems: Advanced Concepts*: Benjamin/Cummings Publishing Company, Inc., 1987.

[49] J. L. M. Lastra and M. Delamer, "Semantic web services in factory automation: fundamental insights and research roadmap," *Industrial Informatics, IEEE Transactions on,* vol. 2, pp. 1-11, 2006.

[50] V. Vyatkin and H. M. Hanisch, "Formal modeling and verification in the software engineering framework of IEC 61499: a way to self-verifying systems," in *Emerging Technologies and Factory Automation, 2001. Proceedings. 2001 8th IEEE International Conference on*, 2001, pp. 113-118 vol.2.

[51] Z. E. Bhatti, R. Sinha, and P. S. Roop, "Observer based verification of IEC 61499 function blocks," in *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on*, 2011, pp. 609-614.

[52] J. Chouinard, J. Lavallée, J.-F. Laliberté, N. Landreaud, K. Thramboulidis, P. Bettez-Poirier, F. Desy, F. Darveau, N. Gendron, and C.-D. Trang, "An IEC 61499 configuration with 70 controllers; challenges, benefits and a discussion on technical decisions," presented at the IEEE Conference on Emerging Technologies and Factory Automation, Patras, Greece, 2007.

[53] V. Vyatkin and J. Chouinard, "On comparisons of the ISaGRAF implementation of IEC 61499 with FBDK and other implementations," in *Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on*, 2008, pp. 289-294.

[54] C.-h. Yang and V. Vyatkin, "Transformation of Simulink models to IEC 61499 Function Blocks for verification of distributed control systems," *Control Engineering Practice,* vol. 20, pp. 1259-1269, 2012.

[55] M. Hirsch, V. Vyatkin, and H. M. Hanisch, "IEC 61499 Function Blocks for Distributed Networked Embedded Applications," in *Industrial Informatics, 2006 IEEE International Conference on*, 2006, pp. 670-675.

[56] Y. Li Hsien, P. S. Roop, V. Vyatkin, and Z. Salcic, "A Synchronous Approach for IEC 61499 Function Block Implementation," *Computers, IEEE Transactions on,* vol. 58, pp. 1599-1614, 2009.

[57] G. Zhabelova and V. Vyatkin, "Multiagent Smart Grid Automation Architecture Based on IEC 61850/61499 Intelligent Logical Nodes," *Industrial Electronics, IEEE Transactions on,* vol. 59, pp. 2351-2362, 2012.