Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 132 (2015) 934 – 941

Procedia Engineering

www.elsevier.com/locate/procedia

The Manufacturing Engineering Society International Conference, MESIC 2015

Design of the architecture of a flexible machining system using IEC61499 Function Blocks

E. Querol[a,*], F. Romero[a], A. M. Estruch[a], J. Serrano[a]

[a]*Department of Industrial Systems Engineering and Design, Universitat Jaume I, Avda. Vicent Sos Baynat s/n, Castellón, 12071, Spain*

Abstract

In this paper a new manufacturing control architecture for flexible manufacturing systems is proposed, with the objective of improving their reconfigurability, adaptability, scalability and performance. To do so, this research focuses on two key aspects: improving the structure of the control model and integrating the process plans into the manufacturing control itself. The reconfigurability of the system is achieved through the utilization of a holarchic architecture made up of IEC61499 function blocks, which act as autonomous and distributable entities. Adaptability and performance are improved by integrating the feature-based process plans of the parts into the control tasks. These plans are modelled using the STEP-NC standard reference models to allow the effective integration of non-linear plans into the manufacturing control. As a result, resources utilization is improved and performance is increased, thereby allowing the size of the system design to be reduced.

Peer-review under responsibility of the Scientific Committee of MESIC 2015

*Keywords:*Flexible manufacturing systems; Distributed control; IEC61499; Function Blocks; STEP-NC

1. Introduction

Nowadays manufacturing systems are strongly limited by their inability to adapt to the rapid changes taking place in the competitive global market in which we live today. In recent decades, the responsiveness of manufacturing systems has been studied from many different perspectives, in an effort to conceive a new manufacturing paradigm capable of fulfilling the current market requirements. From those efforts many kinds of manufacturing systems have

* Corresponding author. Tel.: +34-964 72 80 01.
E-mail address: equerol@uji.es

Peer-review under responsibility of the Scientific Committee of MESIC 2015
doi:10.1016/j.proeng.2015.12.580

seen the light, among which we find the so-called flexible manufacturing systems (FMS). FMS consist of a set of general purpose machines, typically CNC machines, and an automated material handling system [1], the aim of which is to produce a wide variety of products in a very automated way, in contrast to the traditional dedicated manufacturing systems (DMS), which are optimized to produce large numbers of the same product. The main objective of FMS is therefore to offer greater flexibility and allow very different kinds of products to be manufactured, although they have never been widely adopted by the industry due to the high complexity and costs involved [2]. Apart from these drawbacks, FMS are often criticized for their lack of reconfigurability, which makes them incapable of dynamically adapting their capacity, which is usually oversized, and their low performance. Midway between FMS and DMS, a third kind of manufacturing system appeared in the 90s, reconfigurable manufacturing systems (RMS), which are defined as modular systems with a structure that can be rapidly changed in order to quickly adjust its production capacity [3]. RMS attempts to offer a solution to the lack of the reconfigurability that characterizes FMS, although a price has to be paid for this increase in reconfigurability in terms of a reduction in flexibility and, consequently, RMS can produce a smaller range of products than FMS.

In this paper we propose an improvement of the manufacturing control architecture of an FMS, which allows an increase in reconfigurability to be obtained while maintaining the flexibility of the system. This new control architecture also allows for higher system performance, due to better resource utilization.

1.1. Manufacturing control architectures

The control architecture outlined here covers the ISA 2-3 [4] control levels, usually known as manufacturing control [5]. Among the main tasks assigned to this control level, we find the coordination of the physical resources, the job and operation sequencing or the routing of the parts within the system. In an FMS, because of its very nature involving distributed and heterogeneous physical devices, those control tasks are usually distributed among different devices, each one in charge of executing a specific part of the task. This leads to the need to implement a distributed control system (DCS). Traditionally two antagonistic approaches have been used to model such distributed systems: Hierarchical and Heterarchical control structures.

In hierarchical control systems complex problems are broken down into smaller and simpler problems and are distributed in layers forming a tree-like structure. Each node of this tree follows a master/slave relationship with the node above it, so ultimately the central node has full control of the system, like in a centralized system. This architecture has a high degree of efficiency in the decision-making process because it has a global view of the system, although it suffers from its low tolerance to disturbances due to the low autonomy of the lower-level nodes.

Heterarchical architectures, on the other hand, are based on a relationship between equals, in the sense that all the control elements are at the same hierarchical level. This means that all the control nodes are autonomous and cooperate to achieve the global objectives of the system. Those elements are often based on the concepts of Agent technology. An Agent is defined as an autonomous component, physical or logical, capable of achieving its goals and able to interact with other Agents when it cannot achieve them by itself [5]. Heterarchical architectures are conceived to offer a great degree of flexibility and reconfigurability, thanks to the autonomy of their components, which can easily be added and removed from the system without needing to modify the control software. Nevertheless, this kind of control is less optimized than a hierarchical one, since every decision is based on the Agents' local knowledge, without considering the global system.

Between these two architectures, a third one has become popular in recent years, namely the so-called holonic architecture, which merges some of the characteristics of the two mentioned above. In this architecture, the control tasks are distributed between Holons, which are defined as autonomous and cooperative, physical or logical components, which have information about themselves and their environment. Holons at the same time can be composed of other Holons, in a recursive way. At this point Holons look like a special type of Agent, but the biggest difference between Agents and Holons is their organization. Despite being autonomous, Holons are organized in a hierarchical manner, forming what are known as holarchies, which they can join and quit in a dynamic way, although conserving their own individuality when leaving one. This form of organization allows the system to adapt itself dynamically and automatically to the changes in its environment [6]. This architecture has the advantage of being highly configurable and flexible, since it is made up of autonomous entities, while at the same time having an overall view of the system, which allows improvements to be made in the decision-making process.

In this project, a holonic architecture is proposed as the way to resolve the reconfigurability problem of flexible manufacturing systems, from the point of view of manufacturing control. However, to develop and implement such an architecture new techniques and tools are required, since the traditional methodologies of development are considered to be unable to support this kind of implementation. In this sense, the key element that allows the development of a holonic control is the IEC61499 standard.

1.2. IEC61499 overview

The IEC61499 standard is a reference architecture created to facilitate the development of distributed control applications [7]. This standard is built around the concept of Functional Block (FB), which represents the smallest logic unit in the control system. FBs can be seen as the traditional blocks used in the Block Diagrams; they are individual units composed of two parts: their interface and their internal behaviour. The block's interface determines how an FB will communicate and interact with other FBs, while the internal behaviour determines how the block will respond to a specific input. There are three standard types of FB: Basic (BFB), Composite (CFB) and Service Interface (SIFB). These three types of blocks are combined and interconnected to form networks of function blocks, which can be used to model and implement entire control systems.

BFBs are the elemental blocks of the control applications: they are indivisible and autonomous, in the sense that they do not need anything else to carry out their internal functions. CFBs are blocks that work as containers which consist of a network of other blocks. SIFBs are a special type of block whose interface is modelled but whose internal behaviour is only defined in the implementation, therefore remaining outside the scope of the IEC61499. All these types share the same interface, but have different internal definitions. For the sake of brevity only the BFB will be detailed, since it is the most important type.

Figure 1 shows an example of an FB interface; it is composed of its input and output events and its data connections. The events are used to trigger the execution of the FBs, since the IEC61499 architecture is based on an event-driven execution model, where blocks are only executed when required, remaining idle the rest of the time, in contrast to other techniques, like IEC61131, where the control units are executed periodically. The data connections are associated with the events and are used to transmit information between blocks at execution time.

Internally, BFBs consist of a set of algorithms, a context of variables, which can only be utilized by the algorithms of the block, and an execution control chart (ECC). This last element is a kind of machine-state diagram, which determines the current state of the block and how it will react when a new event arrives. Algorithms are associated with the states of the ECC, so whenever the state of the block changes, the associated algorithms are executed, finally producing one or more output events that propagate the execution between blocks.

Once the block types have been modelled, they are interconnected forming networks, which are aggregated into control applications. These networks are distributable, which means that each FB can be executed in a different resource while being part of the overall system. The execution of the FB is carried out by an execution runtime which is not defined by the IEC61499 itself. This distinction between modelling and execution aims to provide the model with greater abstraction so it can be executed in different kinds of devices, with the aim of allowing interoperability between heterogeneous devices.

Runtimes are in charge of executing the IEC61499 models, acting as an intermediate layer between the abstract model and the operating system or firmware of the device, which controls it physically. Runtimes are capable of interpreting IEC61499 models, made according to the standard, and at the same time they can utilize and access the device's interfaces such as networks, I/O, or memory, since each runtime is designed specifically for a device.

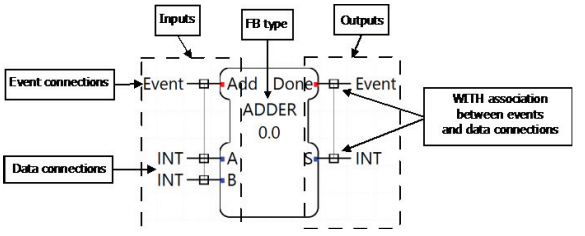

Fig.1.Example of functional block interface

1.3. Integrating the process plans into the manufacturing control

As stated above, the main objective of this project is to improve the reconfigurability, adaptability and performance of FMS, as a first step towards the design of an intelligent manufacturing system (IMS), capable of dealing with current market requirements in terms of autonomy, flexibility, responsiveness and cost reduction. To fulfil this objective, apart from adopting a new architecture, it is necessary to integrate the products' process plans into the manufacturing control in order to increase the performance of the scheduling and execution tasks of the jobs. Process plans contain all the information required to generate a product from its design. Among other things, they contain information about the operation sequence, the fixtures or the tools to be used. In conventional systems, process plans are defined as a simple sequence of operations to be executed by the manufacturing system in a predefined order. This type of planning reduces the complexity of the system but does not allow the utilization of the resources to be optimized according to their availability, thereby reducing, at the same time, the performance of the system since the parts have to wait until the resources that are needed are released before their manufacturing process can continue.

To solve this inconvenience another process planning approach appeared, namely non-linear process plans. These plans do not contain a simple operation sequence, but instead a route with alternatives, such as AND and OR paths, that the product has to follow to be completed. These plans, although more complex, allow more information about the process planning process to be captured and, if integrated with the control system, enhance resource utilization.

The main problem with this approach is the modelling of these non-linearities and the technical integration of the process plan into the control architectures. These two issues can be solved by adopting an approach based on the STEP-NC (ISO 14649) standard, which is defined as a new model of data transfer between CAD/CAM systems and CNC machines [8] based on, and harmonized with, the models for data product standardization defined by the STEP initiative. STEP-NC proposes a new way to model the process plans based on the widely known manufacturing features of STEP AP 224. These features enclose information about both the geometry of the feature and the manufacturing operations and resources needed to generate it. Moreover, this feature-based modelling also allows non-linear process plans to be created by including selective and parallel clauses. Finally, STEP-NC models are completely object-oriented (OO), which makes it easier to integrate the process plans into the control architecture, especially in an OO architecture like the one proposed in this paper, based on functional blocks.

2. Methodology of Development

2.1. The development platform

As mentioned above, to develop a holonic architecture it is necessary to adopt an infrequently used automation standard, the IEC61499, and therefore, before developing the control itself, a development platform which fully supports it has to be built. This platform is composed of two elements: the modelling software, used to create the control applications, and the runtimes to execute the models. In both cases, because the IEC61499 is quite recent, not many tools are available; in the case of the modelling software there are four options: FBDK, 4DIAC-IDE, nxtStudio and ISaGRAF. In this project, 4DIAC-IDE was used, since it is an open-source initiative supported by the community, and is maintained and updated regularly. It fulfils the standard's models and guidelines completely and its graphical interface is based on the renowned Eclipse IDE (Integrated Development Environment).

The selection of the runtime, on the other hand, is more important than the modelling tool since it will determine which devices can be used in the project, as runtimes are device-specific tools and not all support the same kind of hardware. As with the development tools, there are also four alternatives: FBRT, FORTE, nxtF61499RT and ISaGRAF Runtime. The first two are freeware and oriented towards PC-based devices, while the other two are commercial solutions oriented towards PLC-like devices. Of the four, in this project FBRT and FORTE will be used. The first one is a Java-based application, designed to be used mainly on PCs, and it is an interesting solution with which to develop HMIs using function blocks. FORTE, on the other hand, is developed by the same community as 4DIAC-IDE and is also open-source, and therefore modifiable. It is a runtime programmed in C++ that was developed to be executed on small devices and to be portable between infrastructures. FORTE can be executed on a wide range of operating systems such as win32 or POSIX compliant systems (Linux, VxWorks, etc.)

With these two tools we can model and implement an IEC61499 distributed control system, although in a flexible manufacturing system there is another component to take care of when describing the development platform, i.e. the workstations. As stated above, FMS are usually composed of general purpose machines, in our case CNC machining equipment. Industrial CNC machines are characterized by their closed controllers, which do not provide access to the machines' internal data. Moreover, the only way to communicate with them is through a vendor-specific ISO 6983 derived language. Such constraints complicate the development of more intelligent machining systems. To address this issue, over the last decade many efforts have been made to develop more open CNC controllers, one of the most notable advances being the appearance of PC-based CNCs. These devices replace some of the hardware control equipment with applications that simulate their operating, and that are run in a computer using a third party OS like Windows or Linux. This kind of controllers usually also include some kind of API or interface, which can be used to access the controllers' data or to command the machine remotely, thus improving the communication between workstations and the system control. This project takes advantage of these PC-based controllers, such as FAGOR 8070OL, to integrate the machines into the manufacturing control. This is accomplished thanks to the IEC61499 runtimes that can run in parallel to the CNC's controllers on a PC, thereby facilitating communication between devices, since once the interface runtime-controller has been developed, all the communications between the workstations will be carried out directly in function blocks. The result is a high degree of interoperability between devices and the possibility of commanding the machines easily from remote and distributed control nodes.

2.2. Manufacturing control requirements and main tasks

Once the development platform that supports holonic architectures using IEC61499 has been chosen and set up, the next step is to identify the control requirements and to elaborate an architecture to satisfy them. On the one hand, in an FMS there is a huge diversity of different components, and all of them must be able to work together, which means that the architecture must be able to integrate and control heterogeneous devices. In our system, for example, we can find milling machines and lathes from different vendors or other types of workstations like coordinate measuring machines or robot arms, and the control must be able to handle all of them.

On the other hand, the control, as stated above, must be capable of being quickly reconfigured, which means, for example, that workstations can be added and removed from the system without needing to reprogram the control. This capacity for reconfiguration, in addition to resolving one of the traditional problems of FMS, also adds two important features to the system: adaptability and expandability. The first allows the system performance to adapt dynamically, so for example we can think of a manufacturing floor with several flexible manufacturing cells, and those cells can dynamically share some of the machines, allowing an overall reduction in the size of the system. The second, the capacity to expand the systems, allows newer machines to be added if an increase in size is needed. In both cases, thanks to the dynamic reconfiguration, the system size can be better adjusted, thus avoiding oversizing it.

In addition to the above requirements, in the control architecture of an FMS we also expect portability and reusability. These two characteristics allow the control to be reutilized in other systems, with different components, hence avoiding the need to develop dedicated control systems. Portable control systems may be less optimized but make it possible to unify the control architectures on floors with several manufacturing cells.

Together with the control requirements of an FMS, we must also identify the control tasks that have to be carried out by it. According to literature ([5]) there are at least three main generic tasks in the manufacturing control of an FMS: short-term process planning, sequencing and plan execution. Short-term process planning is based on the existence of non-linear process plans, so before the execution of a part, the optimum sequence is calculated depending on the resources available in the system. The sequencing task is in charge of calculating the optimal sequence of jobs required to optimize the usage of the physical resources of the system (machines). This implies that complementary jobs should be processed at the same time to increase overall system performance.

The plan execution task can be broken down into three subtasks. The first one is jobs dispatching, the purpose of which is to route the parts to the resources upon the state of the system at the execution time, that is, to determine where each part should go next. This task is different to scheduling in the temporal domain; the scheduling task calculates the best job sequence for producing all the jobs, viewing the system from a global perspective. Dispatching only determines the best part sequencing according to the instantaneous state of the system resources,

without taking into account what the next part in the execution list is. The second subtask within the plan execution is monitoring, which is responsible for retrieving and stocking information about the work progression, such as what parts are being machined or what the current state of each part is. Finally, the third subtask is the reaction to disturbances, which is in charge of correcting possible errors that may occur while operating.

3. Proposed Architecture

In this section the resulting architecture design will be described while the detailed development of the system and its implementation will be discussed in future works. This architecture can be seen as a holarchy with four layers, as shown in figure 2. At first sight, it may look like a traditional hierarchical architecture, but in this case the control nodes behave in a very different way, since they are made up of function blocks, interconnected in networks and act like Holons. This design is intended to make it easy to add and remove resources. As we will detail later, to achieve this, the nodes in layers 2, 1 and 0 are modelled like branches, each branch being in charge of a particular machine in the system. This means that every workstation will be handled by a dedicated triplet of nodes. The number of branches connected to the layer 3 can change dynamically, therefore machines can be added or removed.

As said before, the proposed control system covers the ISA levels 2-3, leaving level 4 functions, that is to say, logistic-oriented functions among others, out of this architecture. The entry point of this system is the manufacturing orders that are passed on to the control. Each order specifies the type of product to be produced, the quantity and its delivery date. Also as input we find the parts' process plans, which are introduced in a database included in the control itself. Before being able to produce a part, its process plan needs to be included in the base, otherwise the orders referring to it will be rejected. The control outputs are the G-code commands used to control the workstations directly. All the communications between the four layers are performed exclusively using IEC61499 data and events. The internal behaviour of each layer is explained in the following subsections.

3.1. Layer 3: Orders, parts and resource managing

Layer 3 is composed of a single IEC61499 Application, that is, a distributable network of FBs, and is in charge of receiving lot production orders, and sequencing and dispatching them dynamically. Internally, we can distinguish three main components in this application:

- The order manager is a single CFB (L3_OrderManager) that registers the incoming lot orders, dynamically sequences them and starts their execution. Conceptually, this component can be viewed as a table of sequenced orders with their execution information.
- The the items' manager, is a network of CFBs where each one stores all the information about the state of all the items of the same part that have previously been sequenced. All the CFBs in this network are instances of the same CFB, parameterized via different inputs to identify each different part. These CFBs (L3_ProcessPlan) have access to the process plans database so they can retrieve the manufacturing information to be sent to the workstations once the dispatching negotiation has been completed. The entire network is involved in that negotiation through a collaborative mechanism to determine which item should be sent to which available resource, using CFB's local knowledge.
- The machines state manager is a single CFB (L3_MachineManager), which is in charge of handling the machines' changes of state. Every time a machine is released, taken, broken down, fixed, added or removed from the system, this CFB is notified and the change is registered. This CFB is also in charge of starting the dispatching negotiation process, detailed below, so the idle machines can be taken.

The negotiation process (figure 2, right) consists of a repetitive circular requesting mechanism. First of all, the machine manager announces the resources available to the first L3_ProcessPlans CFB connected with it, which checks if any of their managed items needs any of the available machines on offer, and if any are needed, they are assigned with a priority policy based, for now, on the parts' delivery date. This L3_ProcessPlan later sends the current machine assignment again to the next L3_ProcessPlan, which checks their items and updates the assignation of machines when its priority is greater than the previously established one. This process is repeated consecutively

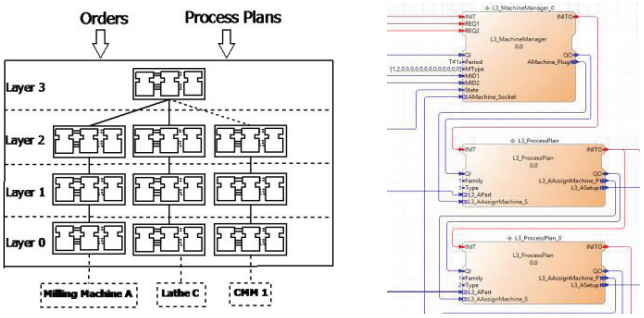

Fig. 2 Proposed 4-layer holonic control architecture (left) and Layer 3 fragment (right).

until the last CFB is reached and announces the final assignment to the machine manager, which verifies the feasibility of the proposal and if some machine assignment is found to be unfeasible, it is discarded. Finally, a second mechanism is started to allocate the final item-machine assignments, where each L3_ProcessPlan CFB retrieves process plan information for each part from the database and transmits it to the chosen machine. The corresponding item execution state is then updated according to the different setups of the process plans of the parts.

3.2. Layer 2: Machine Supervisory Control

Layer 2 consists of an undefined number of CFBs, each one being the virtualization of a real machine in the system. These CFBs are instances of some generic machine model definitions, that is, each type of machine has a generic model definition, and particular machines are seen as instances of this one. For example, in our system we have generic model definitions for milling machines (L2_MillingMachine) or for CMMs (L2_CMM). This makes it possible to have a CFB to control each machine but, at the same time, since all of them are instances of generic types, the development time is reduced and new elements can easily be added to the system due to their being suitable for any of the generic models available. The functionalities of the blocks in this layer are: tracking and monitoring of the state of the machines, receiving the manufacturing information to produce each item, controlling the correct execution of corresponding machining operations and handling possible disturbances. From a global point of view, this layer receives the process plan information corresponding to a setup of a part. Later, it then analyses the different operations to be executed by the machine and chooses, if available, between the different options (non-linear process plans) based on the current state of the machine (available tools and fixtures, etc.). Finally, it sends the manufacturing information corresponding to a single operation to layer 1. After finishing each operation, this layer is notified and the next one is sent until all the operations have been performed. When the setup is completed, this layer indicates back to layer 3 that the part setup has been completed and that the machine is free, and therefore a new part can be assigned to it.

3.3. Layer 1: Feature-FB mapping

This layer consists of a set of independent function blocks contained in a single CFB that is linked directly to the corresponding CFBs in layer 2. Those FBs map the STEP-NC features with IEC61499 FBs, that is, there will be an FB for each different feature. The aim of the mapping is to encapsulate the knowledge of each feature into an FB. For example, when an order to execute a pocket roughing is received from layer 2, it is mapped onto the L1_Pocket FB. This FB contains all the algorithms and information necessary to interpret the parameters of a pocket roughing operation correctly, and therefore it can generate the correct G-code commands to be sent to the machine to carry out the operation. It must be said that these FBs are seen as features, but they execute operations, this association operation-feature corresponding to the STEP-NC Workingstep concept, which will not be detailed here for brevity.

Additionally, this approach also allows other functionalities to be added, such as an online toolpath correction, which rectifies the feature geometric parameters based on the state of the tools or in the information gathered from previous machining operations of the same type. The FBs of this layer are instantiated for every machine in the system, and are based on definitions of generic models developed for generic machines, as in layer 2. This implies

that the output G-Code commands will be formatted in the standard ISO6983. If a non-standard G-code or a different pseudo-code needs to be used, a specific set of FBs can be designed for each particular machine.

3.4. Layer 0: Physical Interface

The last layer is composed of independent CFBs, connected to the blocks of layer 1. These CFBs are the interfaces between the proposed control system and the devices' controllers (CNCs, etc.), which must be developed for each particular device. The main task of these blocks is to receive the commands produced in layer 1 and send them to the devices. The mechanism of transmission between IEC61499 and the machines' controllers changes from one station to another. As an example, when using PC-based CNCs this transmission is usually carried out through shared objects, like COM technology, and their corresponding API. Finally, this layer is also in charge of detecting when the machining or inspection operations are completed, or when the state of the workstations changes, and communicated so to the upper layers. As a final note it must be said that all the control nodes described here, from any layer, can be distributed among different devices. A typical configuration would consist in using one or more central computers for the layer 3 application and embedding the layer 2-1-0 nodes of each machine into a computer

4. Conclusions

The main contribution of this paper is to demonstrate how the adoption of newer control architectures, such as holarchies, can improve flexible manufacturing systems in terms of reconfigurability, adaptability, scalability and performance. To do so, we propose a fully distributable four-layer architecture that facilitates the integration of heterogeneous devices and allows the manufacturing system to be dynamically resized. Moreover, in this paper it is also shown how the effective integration of non-linear process plans into the control system can greatly improve the adaptability and flexibility of the system, by adapting the manufacturing processes to the particular state of the system. Altogether, this architecture is seen as a basis for creating more intelligent flexible manufacturing systems that fulfil the requirements of today's markets in terms of quality, delay and responsiveness.

However, the industrial adoption of those newer architectures is conditioned by the existence of standards and norms that provide a formal reference for their design and implementation. This is the case of the IEC61499 reference architecture, which enables the design and implementation of general distributed control systems, and makes it possible use emergent object-oriented architectures, such as multi-agent and holonic ones, through the functional block concept. Norms, like STEP-NC (ISO 14649), allow the control system to be integrated easily with the CAD/CAPP/CAM chain. Thus, it is possible to take advantage of knowledge generated in the product development stages prior to manufacturing. This integration would be improved if STEP-NC compatible CNC were used, but this kind of machines are not yet available in industry, making it necessary to elaborate an interface to translate from STEP-NC objects to G-code commands that machines can understand.

References

[1] H. A. ElMaraghy, Flexible and reconfigurable manufacturing systems, International Journal of Flexible Manufacturing Systems, 2006, pp. 261–276, DOI: 10.1007/s10696-006-9028-7.

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

[3] Steffen N. Joergensen, Kjeld Nielsen, Kaj A. Joergensen, Reconfigurable Manufacturing Systems as an Application of Mass Customisation, International Journal of Industrial Engineering and Management, 2010, pp. 111-119.

[4] The Instrumentation, Systems and Automation Society. ANSI/ISA-95.00.01-2000: Enterprise Control System Integration, 2000.

[5] Paulo J. Pinto Leitão, An Agile and Adaptive Holonic Architecture for Manufacturing Control, Doctoral thesis in Industrial Automation, Porto: Faculty of engineering of University of Porto, 2004, 297 pages.

[6] Y. Xu, R.Brennan, X. Zhang, D. H. Norrie, A Reconfigurable Concurrent Function Block Model and its Implementation in Real-Time Java, Journal of Integrated Computer-Aided Engineering, 2002, pp. 263-279.

[7] V. Vyatkin, Software Engineering in Factory and Energy Automation: State of the Art Review, IEEE Transactions on Industrial Informatics, 2013, DOI: 10.1109/TII.2013.2258165.

[8] International Organization for Standardization,ISO 14649-10, Industrial automation systems and integration – Physical device control – Data model for computerized numerical controllers – Part 10: General process data, 2004, 153 pages.