IMPROVING PERFORMANCE IN FLEXIBLE MANUFACTURING BY A P-SOP APPROACH

Bo Svensson and Fredrik Danielsson

Department of Engineering Science, University West
SE-461 86 Trollhättan, Sweden

bo.svensson@hv.se

Abstract: This paper present a Part oriented Sequence of Operation (P-SOP) approach that address flexibility, robustness and deployment in manufacturing cells up to plant level. A P-SOP description language has been formulated where the control strategy based on actual circumstances easily can be defined. Furthermore, a P-SOP multi-agent generator has been created that compiles the strategy description to multi-agents that control the manufacturing. Deadlock free IEC 61131-3 PLC code are directly generated from the description language. The code is not optimised for a specific scenario since the general description contains all possible routing paths and all resources available.

Keywords: Multi-agent, Flexible manufacturing, IEC 61131-3, code generation, PLC.

1. INTRODUCTION

In the design phase of a part, as well as in the manufacturing phase, there are a number of well-established computer aided tools widely used in industry. However, the process planning phase in-between has not reached the same level of computer aid and still requires much work of qualified personnel, due to enormous complexities associated with task of the process planning (Xu, Huang, & Rong, 2007). The major output of the process planning is a sequence of operation (SOP) containing a step-by-step work instructions of the manufacturing operations and associated machine tools for assembly or process. Manual process planning is very time-consuming and results are based on the skills of the individual person performing the planning (Stryczek, 2011). Also, the planned SOP is dedicated to a specific production scenario and fixed sequence. In automated manufacturing with low volume or short product lifetime the effectiveness of the process planning is essential. The time to market and production equipment utilisation rate are crucial for survival. The manufacturers need to respond quickly to a dynamic and changing market. The manufacturers need to be productive in all predicted and unpredicted situations. If, for instance, one machine in the process chain breaks down, it is necessary to immediate re-plan the production. The production need to continue until the broken machine is replaced or up and running again.

An arrangement of production equipment grouped to effectively process a set of parts is defined as a manufacturing cell (Jang, Koo, & Nof, 1997). An industrial manufacturing cell is defined from the single machine level to the area level which consists of hundreds of machines (Larin, 1989). In an automated manufacturing cell, the operations such as material handling, machining and assembly are performed by automated equipment such as robots and NC machines under control of a cell controller, typically a Programmable Logic Controller (PLC). PLC's are common in industry because of characteristics such as reliability, modularity, multiple I/O channels, fast response time and cost efficiency. They are used to coordinate and synchronise cell equipment (SOP control) as well as motion control, process control, data management, and intra- and inter-cell communication (Johnson, 1987). PLC's are well known to be robust and easy to program and re-program following the IEC 61131-3 (Standard, 2003) for PLC code. However, the code validation is typically done by direct testing of the PLC through piecewise implementation. Because of the complexity of the PLC code and the manufacturing cells themselves, the implementation and validation are time consuming and expensive. Furthermore, PLC programming is often performed by external experts, which may be hard to access and cause further time delay to production start.

The aim of the research presented in this paper is to describe a multi-agent based control method to handle production with high flexibility. A Part oriented Sequence of Operation (P-SOP) description language has been formulated to assist in the daily process planning work. The general description contains all possible routing paths and all resources available. The description is based on actual circumstances, e.g., new parts, rebalancing of manufacturing cells due to market changes, scheduling of humans, and rerouting of the production, to handle flexibility. Another aim is to automatically generate multi-agent based control from a high-level description. A P-SOP multi-agent generator has been created that compiles the planning description to multi-agents that control the manufacturing. The P-SOP multi-agent generator generates robust deadlock free code directly from the description language to IEC 61131-3 PLC code. The overall aim is to create possibilities to change, rebalance and introduce new parts into a manufacturing cell up to a plant level. Furthermore, eliminate the need for an external expert in PLC programming and avoide code optimised for a specific scenario, i.e. a non-adaptable solution with a "single point of failure".

2. RELATED WORKS

Castillo and Smith present a survey and a comparison of several important formal modelling methodologies that have been used for manufacturing cell control (Castillo & Smith, 2002). Formal modelling methodologies provide a framework that supports formal verification and validation techniques. They are directed toward integrating the requirements specification, the design, and the implementation of the cell controller into a consistent process, supported by efficient analysis and development tools. A formal model is defined (Castillo & Smith, 2002) as an unambiguous, complete, verifiable, and consistent specification of a manufacturing cell in an implementation-independent language with precisely defined syntax and semantics. The search for a single methodology powerful enough to span all the stages of the development life cycle of a formal model capable of replacing existing non-formal practices, however, has been proved to be more difficult than expected. In practice the use of formal modelling methodologies for manufacturing cell control remains limited. The design is typically based on flow charts or textual control specifications. Furthermore direct testing of the cell controller through piecewise implementation is far more common than verification and validation using formal modelling methodologies. Formal modelling methodologies in the manufacturing cell control domain should support a thread from requirements specification to implementation to be useful in practice. Silva et al. present a survey of modelling methodology creations for Petri Nets compatible with the IEC 61131-3 standard (Silva, Benitez, Villafruela, Gomis, & Sudria, 2008). They did not find any creation that present a completely compatible methodology with the standard. To handle this they present an extended GHENeSys Petri Net model to develop PLC applications. This extended version, with added functions etc. in the Petri Net, allows the creation of a hierarchical model with high-level modularity, similar to IEC 61131-3 PLC code. However, it is only tested in laboratory practices. Though it claims to not require an industrial specialist, it is not obvious to be useful in industry.

An interesting approach of using an object oriented graphical program language to generate IEC 61131-3 PLC code is presented (Vogel-Heuser, Wisch, & Katzke, 2005). They used the Unified Modelling Language for Process Automation (UML-PA), a development of the standard UML. A prototype has been developed, which generates IEC 61131-3 structured text and SFC code automatically from the UML-PA model created with an UML tool. However, the benefits of object orientation as well as bottlenecks have not yet been demonstrated. They will become apparent during application in a more complex system, e.g., a real manufacturing cell. Thapa et al. present an approach with the objective to reduce the development time of cell controller by automating the task of code generation (Thapa, Park, Park, & Wang, 2009). They applied t-MPSG (Timed Message-based Part State Graph), an extended version of conventional MPSG (Smith, Joshi, & Qiu, 2003) with temporal properties embedded. The method deploys the t-MPSG to generate a generic PLC code. The similarity in the hierarchical structure of the t-MPSG and the IEC standard has made it convenient to transform from one form to another. However, only an experimental example is presented, no implementation in real industry has been done.

A new sequence planning approach, where sequences are viewed based on the relations among operations instead of manually constructing the sequence is proposed (Lennartson, o.a., 2010). In order to obtain a unified information flow from early product design to final production, an integrated framework for product, process and automation design has been developed. The framework is based on SOPs and includes a formal relation between product properties and process operations. A formal graphical language for hierarchical operations and SOPs is introduced and defined based on automata extended with variables. Since the operations are self-contained they can be grouped and viewed from different point of views, e.g., from a product or a resource perspective. One approach to limit the negative influence of uncertainty and unpredictable behaviour on manufacturing performance is to use real-time manufacturing information and intelligence in order to perform at a higher level with a minimum of unscheduled downtime. Wang et al. research objective is to develop methodologies for distributed, adaptive and dynamic process planning as well as machine monitoring and control, using event-driven Function Blocks (FBs) (Wang, Adamsson, Holm, & Moore, 2012). They utilise the concept of FBs as defined in IEC 61499 (Commission,

2005). The concept of FBs enables the use of real-time information for dynamic distributed decision making, and possesses dynamic control capabilities that are able to handle different kinds of uncertainty.

Approaches and tools like Sequence Planner (Lennartson, o.a., 2010) and Supremica (Åkesson, Fabian, Flordal, & Vahidi, 2003) can be used to provide a formal method to generate the SOP, to guarantee an optimal and deadlock free solution. However, to be flexible and robust it is also desirable to avoid code optimised for a specific scenario, i.e. a non-adaptable solution with the potential risk "single point of failure". A "single point of failure" is that one single failure stops the entire manufacturing plant or cell from working.

A Holonic Manufacturing System (HMS) enables the possibilities to reach a flexible and robust manufacturing cell and avoid code optimised for a specific scenario (Gou, Luh, & Kyoya, 1998). A HMS is a system where elements, such as machines, parts, humans, carriers, etc., are modelled as holons. A holon is characterised by autonomy and cooperation, and consists of a physical component and an informational component. One well-known software engineering technologies well suited to implement this kind of holonic concept is multi-agent systems which have been extensive researched recent 20 years because of its distributed and proactive nature (Colombo, Schoop, & Neubert, 2006), (Ulieru, Brennan, & Wakler, 2002) and others. The multi-agent system paradigm comes from the distributed artificial intelligence field, being characterized by decentralization and parallel execution of activities based on autonomous entities (Leitão, 2009). A multi-agent based solution can handle disturbances and failures in the manufacturing cell (Valckenaers & Van Brussel, 2005) and thus avoid "single point of failure", i.e. one single failure stop the entire cell from working.

3. P-SOP DESCRIPTION LANGUAGE

The aim of the P-SOP approach is to create possibilities to change, rebalance and introduce new parts into a manufacturing cell or indeed a manufacturing plant. This new multi-agent based approach will achieve a more flexible and robust manufacturing than previous works. The P-SOP approach handle break-downs and other failure scenarios in the PLC code and adapt to the new upcoming situations thus avoiding "single point of failure".

The planning task for a manufacturing cell can be described with the P-SOP description language. The P-SOP description language is prepared for a graphical interface in purpose to simplify the definition work even more, however this is not covered in this paper. A P-SOP description must contain two sections: (1) declaration of all parts and resources that belongs to the actual manufacturing cell, and (2) all possible sequence steps. All possible sequence steps are not necessary executed or used by the manufacturing cell. The decision to execute one single sequence step is made on-line by the agents. Hence, to be able to handle unpredictable situations the agents need alternative sequence steps to be more flexible and robust. The two defined section of the P-SOP description is then used to automatically generate individual agent heads and communicators. The agent head consist of agent threads based on the information from all tasks that are specified for that specific resource. The communicator consist of all possible communication paths in-between agents.

Declarations. The declarations section of the P-SOP description needs only to be updated when a new type of part is introduced or when the manufacturing cell is rebuilt, e.g., after installing a new resource. In the declaration section all k part types that are planned to be handled in the actual manufacturing cell are specified in the set

$$P = \{p_1, p_2, \dots, p_k\}. \tag{1}$$

Each single part type $p \in P$ must be declared with a part identity, $p.id$. Although each part is an element in a set, we take the freedom to define properties for such elements with a dot operator. Further, all resources that belongs to the actual manufacturing cell are specified in the declaration section in the set as

$$R = SOURCE \cup SINK \cup CARRIER \cup PROCESS \cup BUFFER, \tag{2}$$

where $SOURCE$ is the set of all point-of-entry of parts; $SINK$ is the set of all points for outgoing parts; $CARRIER$ is the set of resources able to transport parts from one place to another, e.g. robots, humans, conveyers, cranes, and AGV's; $PROCESS$ is the set of all resources capable of creating or refining parts, e.g. humans, CNC's, turning and milling machines; and $BUFFER$ is the set of all storage locations for parts, e.g. storehouses, magazines, or intermediate storage locations.

Each single resource $r \in R$ possess locations for parts going in and out of the resource. Locations for a specific resource are defined as $r^{(l)}$, where $l = 1$ to q, and $q =$ specified maximum number of locations the actual resource possesses. We also take the freedom to define two properties, booked and id, for all resource locations $r^{(l)}$. The property booked, $r^{(l)}.bk$, is a Boolean property, where $r^{(l)}.bk = FALSE$ corresponds to resource location $r^{(l)}$ available, and $r^{(l)}.bk = TRUE$ corresponds to resource location $r^{(l)}$ booked. The property id, $r^{(l)}.id$, is the current part identity, $p.id$, located at the resource location $r^{(l)}$.

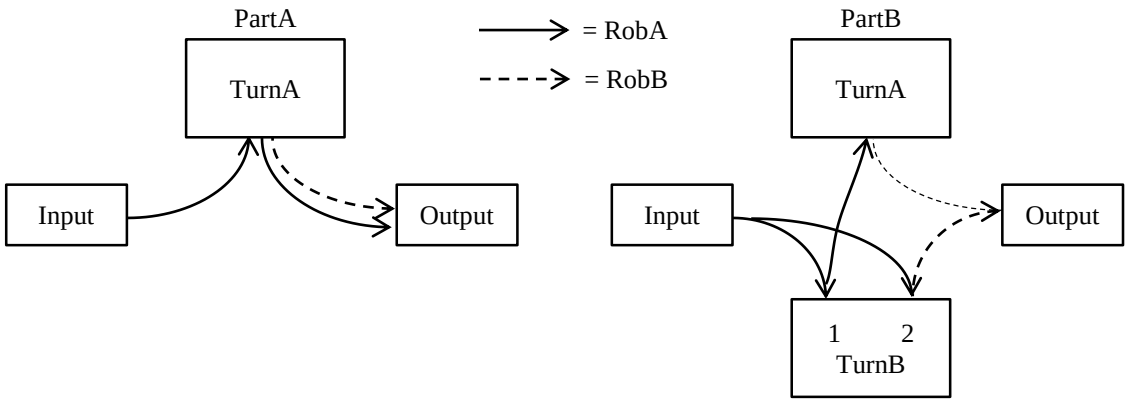

Fig. 1. Sequence example for two parts types, Part A and Part B, in one manufacturing cell comprehending two robots, RobA and RobB, and two turning machines, TurnA and TurnB. The resource TurnB possess 2 locations, 1 and 2, all other resources only one.

Sequences. In the sequence section of the P-SOP description all n possible sequences in the manufacturing cell are specified in the set

$$S = \{s_1, s_2, \dots, s_n\}. \tag{3}$$

Each specific sequence, $s_i \in S$, consist of one or more tasks to be carried out on a specified subset of parts, $P_i \subseteq P$, where $i = 1$ to n. A single task in a sequence s_i is denoted $s_{i,j}$, where $j = 1$ to m, and $m =$ number of tasks to be carried out in the specific sequence s_i. Each task, $s_{i,j}$ includes two types of operations, one carrier operation, $op^c_{i,j} \in OP$, and one resource operation, $op^r_{i,j} \in OP$, where OP is the set of all possible operations that can be carried out by resources in the manufacturing cell. We define the task $s_{i,j}$ as:

$$s_{i,j} := \langle label_{i,j}, CA_{i,j,} RF_{i,j}, RT_{i,j} \rangle, \tag{4}$$

where, $label_{i,j}$ is a user-defined label for identification (it has nothing to do with the sequence order); $CA_{i,j} \subseteq CARRIER$ is a subset of carriers; $RF_{i,j} \subseteq R$ is a subset of from-resources; and $RT_{i,j} \subseteq R$ is a subset of to-resources, all specified for the specific task $s_{i,j}$. Hence, a task $s_{i,j}$ always embraces three resources, a carrier, a from-resource and a to-resource. These resources forms a three-tuple. We define the resource three-tuple $r_{i,j}$ for task $s_{i,j}$ as:

$$r_{i,j} := \langle ca_{i,j,} rf_{i,j}, rt_{i,j} \rangle. \tag{5}$$

The following example will demonstrate the P-SOP description language as shown in Fig. 1. It illustrates the handling of two part types, PartA and PartB, in a small manufacturing cell. The cell in the example comprehends two robots, RobA and RobB, two turning machines, TurnA and TurnB, a source, Input, and a sink, Output. All resources possess only one location each, except TurnB that possess two locations, 1 and 2.

The sequence of PartA:

[carried by RobA from Input to TurnA, processed in TurnA, and carried by RobA or RobB from TurnA to Output]

The sequence of PartB:

[carried by RobA from Input to TurnB (location 1), processed in TurnB, carried by RobA from TurnB (location 1) to TurnA, processed in TurnA, and carried by RobB from TurnA to Output] ***OR***

[carried by RobA from Input to TurnB (location 2), processed in TurnB, and carried by RobB from TurnB (location 2) to Output].

The P-SOP description of the sequence example in Fig. 1 is shown in Fig. 2. In the declaration section, on the left side, all resources and parts are declared. All resources are default declared with one location, however, in this example, TurnB possess two resource locations and are therefore declared as TurnB[2]. Each part type must be declared with a part identity, $p.id$, in this example PartA[1] and PartB[2]. The P-SOP description sequence section of the example is on the right side in Fig. 2. When it is necessary to pick or place from a specific resource location it has to be defined in the sequence, e.g. TurnB(1) and TurnB(2). However, if all locations of a resource can be used, the * operator can be used, e.g. TurnB(*). The * operator is a general operator and can be used whenever the

entire set is selected. When two or more alternative resources can be used, the / operator can be used, e.g. RobA /RobB.

```
/* P-SOP declaration */
/* Resources */
SOURCE  {Input};
SINK    {Output};
CARRIER {RobA, RobB};
PROCESS {TurnA, TurnB[2]};
/* Part types */
PART    {PartA[1], PartB[2]};
```

```
/* P-SOP sequences */
SEQUENCE(PartA)
{a: RobA        Input   ->  TurnA;
 b: RobA /RobB  TurnA   ->  Output;
};
SEQUENCE(PartB)
{c: RobA  Input           ->  TurnB(*);
 d: RobA  TurnB(1)        ->  TurnA;
 e: RobB  TurnA /TurnB(2)->  Output;
};
```

Fig. 2. P-SOP description of PartA and PartB sequences in manufacturing cell example in Fig. 1. The P-SOP description declaration section on left side and sequence section on right.

4. P-SOP MULTI-AGENT GENERATOR

The P-SOP multi-agent generator automatically generates robust agents in IEC 61131-3 PLC code out from the P-SOP description. A generated agent consist of a head and a communicator. Where the head consist of all generated agent threads for a specific resource. The P-SOP multi-agent generator establishes a decentralised control strategy based on multi-agent techniques and a non-hierarchical structure. All generated agents co-operate and communicates with each other to manage the overall tasks. A single task in the P-SOP description must include two operations, a carrier operation and a resource operation. Hence, two separate agent threads are generated by the P-SOP multi-agent generator, a carrier thread and a resource thread. A single operation are guarded by its precondition and postcondition. These conditions are automatically generated from the P-SOP description. A condition is always dependent on the status of neighbouring agents and are therefore communicated through the communicator part of the agent. Furthermore, for each agent thread a preoperation and a postoperation are automatically generated to handle the mutual exclusion logic, this to guarantee that an operation will be executed and only executed by one agent thread.

Precondition. The specific task $s_{i,j}$ can only be executed if the precondition $c_{i,j}^{\uparrow} \in C^{\uparrow}$ is fulfilled. $C^{\uparrow}$ is the finite set of all physical and logical preconditions in the manufacturing cell. The precondition, $c_{i,j}^{\uparrow}$, is automatically generated by the P-SOP multi-agent generator. We define the precondition as:

$$c_{i,j}^{\uparrow} = \left(AC_{i,j} \neq \emptyset\right) \wedge \left(AF_{i,j} \neq \emptyset\right) \wedge \left(AT_{i,j} \neq \emptyset\right) \wedge \left(RD_{i,j} \neq \emptyset\right) \wedge \left(r_{this} \subset \langle ca_{opt}, rf_{opt}, rt_{opt} \rangle\right). \quad (6)$$

Hence, a resource will only be activated if there are: a carrier available AND a part to fetch AND a resource available AND a deadlock free path AND the resource is one of the selected preferred resources. The precondition includes a set of available carriers $AC_{i,j}$, a set of available from-resources $AF_{i,j}$, a set of available to-resources $AT_{i,j}$ and a set of deadlock free resource three-tuples $RD_{i,j}$. The set of available carriers, i.e. carriers specified for task $s_{i,j}$ with a not booked location, are defined as:

$$AC_{i,j} = \left\{ r \in CA_{i,j} \mid \left(r^{(l)}.bk = FALSE \right) \right\}. \quad (7)$$

The set of available from-resources, i.e. resource locations that possess a part with an identity that are handled by the task $s_{i,j}$, are defined as:

$$AF_{i,j} = \left\{ r \in RF_{i,j} \mid \left(r^{(l)}.bk = FALSE\right) \wedge \left(\left\{p \in P_i \middle| p.id = r^{(l)}.id\right\} \neq \emptyset\right) \right\}. \quad (8)$$

The set of available to-resources, i.e. processing resources specified for task $s_{i,j}$ with a location not booked or occupied by another part, are defined as:

$$AT_{i,j} = \left\{ r \in RT_{i,j} \mid \left(r^{(l)}.bk = FALSE\right) \wedge \left(r^{(l)}.id = \emptyset\right) \right\}. \quad (9)$$

The set of deadlock free resource three-tuples, i.e. all combinations of available resources for task $s_{i,j}$ formed as $\langle ca_{i,j}, rf_{i,j}, rt_{i,j} \rangle$ that does not generate a deadlock, are defined as:

$$RD_{i,j} = \left\{ \langle \forall ca_{i,j} \in AC_{i,j}, \forall rf_{i,j} \in AF_{i,j}, \forall rt_{i,j} \in AT_{i,j} \rangle | d\left(R, P, S, \langle ca_{i,j}, rf_{i,j}, rt_{i,j} \rangle\right) = TRUE \right\}. \quad (10)$$

A method for deadlock avoidance has been implemented in the P-SOP agent generator. Deadlock avoidance functions, $d \in D$, are automatically generated from the P-SOP description. A specific deadlock avoidance function d is evaluated according to $d(R, P, S, \langle ca_{i,j}, rf_{i,j}, rt_{i,j}\rangle) \in \{TRUE, FALSE\}$. If d yields $TRUE$ it is a guarantee that deadlock situations never occurs for the specific task and resource selection. The implemented function has been shown to generate deadlock free solutions. Nevertheless, the deadlock avoidance function could be further improved and there are many different approaches described in the literature, among all (Huang, Pan, & Zhou, 2012), (Mohan, Yalcin, & Khator, 2004), (Fahmy, Balakrishnan, & ElMekkawy, 2011).

The selected preferred resource three-tuple combination of a carrier, a from-resource and a to-resource, based on a decision and the mutual exclusion logic, are defined as:

$$\langle ca_{opt}, rf_{opt}, rt_{opt}\rangle = opt(RD_{i,j}), \tag{11}$$

where, $ca_{opt} \in AC_{i,j}$ is the selected preferred single carrier location, $rf_{opt} \in AF_{i,j}$ is the selected preferred single resource location, and $rt_{opt} \in AT_{i,j}$ is the selected preferred single resource location for task $s_{i,j}$. The P-SOP generated agents has a priority function opt to select a preferred resource three-tuple combination $\langle ca_{opt}, rf_{opt}, rt_{opt}\rangle$ among all deadlock free combinations of available resources in the set $RD_{i,j}$. The priority function can be designed to prioritise different scenarios, e.g., rush orders or balancing. A simple opt function could be formulated as the sum of all predicted processing times for a specific three-tuple, i.e.

$$\langle ca_{opt}, rf_{opt}, rt_{opt}\rangle = \min_{arg} \sum \Big(time\big(ca_{i,j}\big) + time\big(rf_{i,j}\big) + time\big(rt_{i,j}\big)\Big). \tag{12}$$

However, this priority function could be further improved and is an open research question.

Preoperation. When the precondition $c_{i,j}^{\uparrow}$ is fulfilled a preoperation, $op_{i,j}^{\uparrow}\big(\langle ca_{opt}, rf_{opt}, rt_{opt}\rangle\big)$, will be executed by related agent thread. Based on decision and the mutual exclusion logic the following resource bookings will be performed to guarantee that the operation will be executed and only executed by one resource:

$$ca_{opt}.bk \coloneqq TRUE, \tag{13}$$

$$rf_{opt}.bk \coloneqq TRUE, \tag{14}$$

$$rt_{opt}.bk \coloneqq TRUE. \tag{15}$$

Note that the two booking ca_{opt} and rf_{opt} are executed in the carrier agent thread, and the booking rt_{opt} is executed in the resource agent thread, see Fig. 3.

Operation. When the preoperation is done, the main operation $op_{i,j}(p_i.id)$ is executed. The operation carries out the resource operations, e.g. robot motion, machining, welding. However, each task $s_{i,j}$ in the P-SOP description includes two types of operations. Hence two separate operations are generated by the P-SOP multi-agent generator, one carrier operation, $op_{i,j}^{c}(p_i.id)$, and one resource operation $op_{i,j}^{r}(p_i.id)$. The carrier operation is executed in the carrier agent thread and the resource operation is executed in the resource agent thread.

Postcondition. The operation is ready when the postcondition $c_{i,j}^{\downarrow} \in C^{\downarrow}$ is fulfilled. $C^{\downarrow}$ is the finite set of all physical and logical postconditions, e.g. part placed at the resource to-location, welding operation done, inspection finished. Since each task includes two operations, $op_{i,j}^{c}$ and $op_{i,j}^{r}$, also two separate postconditions exists, one carrier postcondition $c_{i,j}^{c\downarrow} \in C^{\downarrow}$ and one resource postcondition $c_{i,j}^{r\downarrow} \in C^{\downarrow}$. These two postconditions are not synchronized and thus independent of each other and may thereby be fulfilled independently.

Exceptions are anomalous or exceptional events requiring special handling, e.g., machine failures, part drop-out. If an exception is continuable, the agent may resume the handling at current location. If it cannot be resolved it might be necessary to allow carriers or humans to remove a part from the resource. In order to reach flexibility and avoid single point of failure due to exceptions this must be handled without disturbing the rest of the production. Hence, all postconditions $c_{i,j}^{\downarrow}$ generated by the P-SOP multi-agent generator include an 'operation cancelled' event. Humans or other agents in the manufacturing cell can fire this event if needed.

Postoperation. A postoperation, $op_{i,j}^{\downarrow}\big(\langle ca_{opt}, rf_{opt}, rt_{opt}\rangle\big)$, will be executed by the related agent thread after the postcondition $c_{i,j}^{\downarrow}$ is fulfilled, it performs the following release of resources concerned:

$$ca_{opt}.bk \coloneqq FALSE, \tag{16}$$

$$rf_{opt}.bk \coloneqq FALSE, \tag{17}$$

$$rt_{opt}.bk \coloneqq FALSE. \tag{18}$$

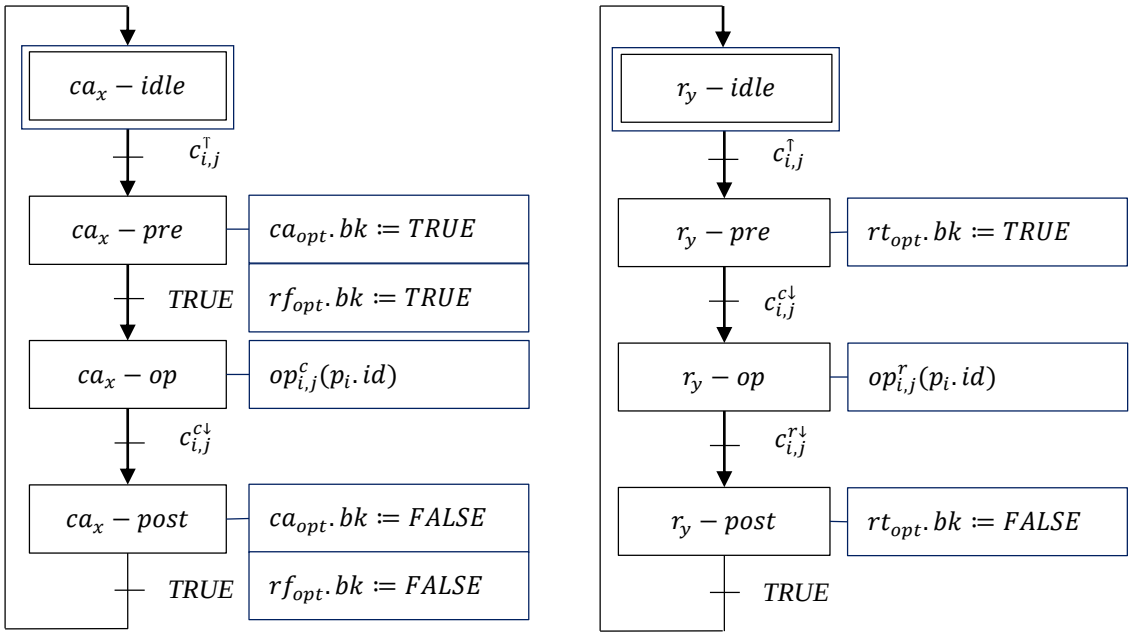

Fig. 3. Two generated agent threads described in two SFC-like charts. The agent threads corresponds to the P-SOP description of the single task $s_{i,j} := \langle label_{i,j}, CA_{i,j,}\, RF_{i,j}, RT_{i,j} \rangle$. The left chart describes the corresponding agent thread for the carrier ca_x and the right chart for the resource r_y.

Note that the two release ca_{opt} and rf_{opt} are executed in the carrier agent thread, and the release rt_{opt} is executed in the resource agent thread unsynchronized, see Fig. 3. This allows the carrier and from-resource to be booked by a new task even if the to-resource still is booked by the actual task.

An explanatory figure of two generated agent threads from one task $s_{i,j} := \langle label_{i,j}, CA_{i,j,}\, RF_{i,j}, RT_{i,j} \rangle$ can be found in Fig. 3. Observe that both agent threads in Fig. 3 originates from the same task $s_{i,j}$ since each task includes two types of operations. The chart on the left side describes the agent thread for the carrier $ca_x \in CARRIER$ and the chart on the right side the agent thread for the resource $r_y \in R$, each with adherent precondition, preoperation, operation, postcondition, and postoperation. Note that the carrier return to its idle state when done, i.e. ready to perform another carrier operation meanwhile the resource refine the part. The P-SOP multi-agent generator generates similar agent threads out from all tasks in all possible sequences, *S*, of the manufacturing cell. Finally, all generated agent threads for a specific resource are gathered and form that resource's agent head.

5. CONCLUSION

The presented method in this paper is a novel approach, based on P-SOP, to handle multi-agent based control. The P-SOP multi-agent approach address flexibility, robustness and deployment in the best possible manner with the least waste of time and effort. The main contributions are:

(a) a description language, able to handle planning on a high level in a flexible way (the P-SOP description language), and

(b) a multi-agent generator, capable of generating IEC 61131-3 PLC code with a guarantee of a robust and deadlock free control strategy (the P-SOP multi-agent generator).

The P-SOP multi-agent generator compiles the planning description to multi-agents that are robust and flexible. In this way it is possible to avoid code optimised for a specific scenario, i.e. a non-adaptable solution with "single point of failures". In the deployment phase the approach eliminate the need for experts in PLC programming and reduce the required time dramatically. Additionally, the generated code is free from programming mistakes and errors. The P-SOP approach is designed to handle and program PLC's at manufacturing cell level up to plant level in a multi-agent based way.

Furthermore, the P-SOP approach has been implemented and evaluated in two test case studies, (I) with a virtual manufacturing cell, (II) with a real manufacturing cell. The virtual manufacturing cell (I) included a conveyor,

four robots, five machines, and a number of storage places. The virtual cell was built in the simulation tool Process Simulate and an OPC-interface was used for communication with the PLC. The real industrial manufacturing cell (II) is placed at Production Technology Centre in Trollhättan, Sweden. The real cell includes a conveyor, four robots, two machines, and a number of storage places. Both test cases confirmed the applicability of the P-SOP description language and the P-SOP multi-agent generator. The P-SOP description language assist in the process planning work to be able to handle flexible manufacturing scenarios. Rebalancing due to production changes, introduction of new parts, and rerouting due to a machine break down or planned service were verified. The P-SOP multi-agent generators automatic generation of the robust deadlock free IEC 61131-3 PLC code eliminates the need of a PLC programming expert and reduce the required time dramatically. Re-scheduling and changing of manufacturing part mix in the test cases have been done in a few minutes during on-going production.

ACKNOWLEDGEMENTS

This work was supported in part by the Swedish Foundation for Knowledge and Competence Development.

REFERENCES

Castillo, I., & Smith, J. (2002). Formal Modeling Methodologies for Control of Manufacturing Cells: Survey and Comparison. *Journal of Manufacturing Systems, 21*(1), 40 - 57.

Colombo, A., Schoop, R., & Neubert, R. (2006). An Agent-Based Intelligent Control Platform for Industrial Holonic Manufacturing Systems. *IEEE Transactions on Industrial Electronics, 53*(1), 322-337.

Commission, I. E. (2005, January). *IEC 61499 Function Blocks - Part 1: Architecture.* Retrieved from INTERNATIONAL ELECTROTECHNICAL COMMISSION FUNCTION BLOCKS –: http://webstore.iec.ch/preview/info_iec61499-1%7Bed1.0%7Den.pdf

Fahmy, S., Balakrishnan, S., & ElMekkawy, T. (2011). Deadlock prevention and performance oriented supervision in flexible manufacturing cells: A hierarchical approach. *Robotics and Computer-Integrated Manufacturing, 27*(3), 591-603.

Gou, L., Luh, P., & Kyoya, Y. (1998). Holonic manufacturing scheduling: architecture, cooperation mechanism, and implementation. *Computers in Industry, 37*(3), 213 - 231.

Huang, Y.-S., Pan, Y.-L., & Zhou, M.-C. (2012, March). Computationally Improved Optimal Deadlock Control Policy for Flexible Manufacturing Systems. *IEEE Transactions on Systems, Man, and Cybernetics, 42*(2), 404-415.

Jang, J., Koo, P., & Nof, S. (1997). Application of design and control tools in a multirobot cell. *Computers and Industrial Enginering, 32*(1), 89-100.

Johnson, D. (1987). *Programmable Controllers for Factory Automation.* New York, USA: Marcel Dekker Inc.

Larin, D. (1989). Cell control: what we have, what we need. *Manufacturing Engineering*, 41-48.

Leitão, P. (2009). Agent-based distributed manufacturing control: A state-of-the-art survey. *Engineering Applications of Artificial Intelligence, 22*, 979-991.

Lennartson, B., Bengtsson, K., Yuan, C., Andersson, K., Fabian, M., Falkman, P., & Åkesson, K. (2010). Sequence Planning for Integrated Product, Process and Automation Design. *IEEE Transaction on Automation Science and Engineering, 7*(4), 791 - 802.

Mohan, S., Yalcin, A., & Khator, S. (2004). Controller design and performance evaluation for deadlock avoidance in automated flexible manufacturing cells. *Robotics and Computer-Integrated Manufacturing, 20*(6), 541-551.

Silva, J., Benitez, I., Villafruela, L., Gomis, O., & Sudria, A. (2008). Modeling extended Petri nets compatible with GHENeSys IEC61131 for industrial automation. *The International Journal of Advanced Manufacturing Technology, 36*(11 - 12), 1180 - 1190.

Smith, J., Joshi, S., & Qiu, R. (2003). Message-based Part State Graphs (MPSG): A formal model for shopfloor control implementation. *International Journal of Production Research, 41*(8), 1739-1764.

Standard, I. (2003). *IEC 61131 Programmable controller - Part 3: Programming Languages* (2nd ed.).

Stryczek, R. (2011). A Hybrid Approach for Manufacturablilty Analysis. *Advances in Manufacturing Science and Technology, 35*(3), 55-70.

Thapa, D., Park, C., Park, S., & Wang, G. (2009). Auto-Generation of IEC Standard PLC Code Using t-MPSG. *International Journal of Control, Automation, and Systems, 7*(2), 165-174.

Ulieru, M., Brennan, R., & Wakler, S. (2002). The holonic enterprise: a model for internet-enabled global manufacturing supply chain and workflow management. *Integrated Manufacturing Systems, 13*(8), 538-550.

Valckenaers, P., & Van Brussel, H. (2005). Holonic Manufacturing Execution Systems. *CIRP Annals - Manufacturing Technology, 54*(1), 427-432.

Wang, L., Adamsson, G., Holm, M., & Moore, P. (2012). A review of function blocks for process planning and control of manufacturing equipment. *Journal of Manufacturing Systems, 31*, 269 - 279.

Vogel-Heuser, B., Wisch, D., & Katzke, U. (2005). Automatic code generation from a UML model to IEC 61131-3 and system configuration tools. *International Conference on Control and Automation*, (pp. 1034 - 1039). Budapest, Hungary.

Xu, N., Huang, S., & Rong, Y. (2007). Automatic setup planning: current state-of-the-art and future perspective. *International Journal of Manufacturing Technology and Management, 11*(2), 193-208.

Åkesson, K., Fabian, M., Flordal, H., & Vahidi, A. (2003). Supremica – A Tool for Verification and Synthesis of Discrete Event Supervisors. *Proceedings of the 11th Mediterranean Conference on Control and Automation.* Rhodos, Greece.