THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Automated Verification and Generation of Flexible Automation Control

Johan Richardsson

Department of Signals and Systems
Chalmers University of Technology
Göteborg, Sweden, 2007

Automated Verification and Generation of Flexible Automation Control

Johan Richardsson

ISBN: 978-91-7291-962-4

Doktorsavhandlingar vid Chalmers tekniska högskola

Ny serie nr: 2643

ISSN 0346-718X

Department of Signals and Systems

Chalmers University of Technology

SE-412 96 Göteborg

Sweden

Telephone +46 (0)31-772 10 00

Cover illustration: A model of six robots assembling the front part of the body of a Volvo S60 by spot welding.

Printed by Chalmers Reproservice

Chalmers University of Technology

Göteborg, Sweden 2007

Abstract

Consumer product life-cycles are constantly shortening; the automotive industry is an illustrative example. As a consequence, the introduction of new products into the manufacturing system necessarily becomes more frequent. Inherently, this brings a performance reduction for the manufacturing system. The reduced performance is caused by a down-time and a ramp-up-time. During the down-time the mechanical equipment is rebuilt and the new control programs are debugged. During ramp-up there are a large number of errors mainly caused by mechanical devices not being properly adjusted, bugs in the control programs and operators not used to new procedures. Thus, in order to maintain the productivity level and to achieve full cost-efficiency both the down-time and the ramp-up time must be reduced. One way to reduce these lead times is to verify the control programs in offline mode. However, efficient and reliable offline verification requires some major improvements of the current development process of manufacturing systems. Information handling and development of control programs based on information reuse are the two most important improvement areas.

The work presented here addresses four industrial problems related to this, lack of tools for offline verification of control programs, lack of information reuse in the development process of a manufacturing system, lack of operator support in error situations, and lack of tools for analyzing the control of complex manufacturing cells.

We propose a development method where information from different tools in the development process of a manufacturing system is reused and processed by tools for verification and optimization. Then the control programs are generated by combining the processed information with a library of standardized software components. The proposed method solves the above-mentioned industrial problems without adding work to the development process. On the contrary, the amount of work will be reduced since the control program development will be automated and the time for debugging the control programs on the shop floor will be drastically reduced, due to the new mathematically based verification process.

Acknowledgments

First I would like to thank my industrial supervisor Christer Gullbrandsson, Volvo Car Corporation and Professor Bengt Lennartson, Chalmers for their work in getting this research project started. Once started, my academic supervisor Martin Fabian has been of great help by long discussions usually ending in a solution or in a couple of new problems. Martins excellent grasp of the grammar of the English language has also been valuable, but even more his ability to always answer yes to the question, "is it possible to do this with supervisory control theory?"

For a project long evaluation of my work with a viewpoint from outside the Automation group I thank Professor Tomas McKelvey, who's yearly feedback have strengthened the project.

My colleges at Manufacturing Engineering at Volvo and in the Torslanda plant, whose input has helped me to keep my feet on the shop floor, thanks to all of you.

At Chalmers I'm a part of a group that have had many interesting discussions, so, thank you Tord Alenljung, Petter Falkman, Oscar Ljungkrantz, Marcus Sköldstam, Kristin Andersson, Knut Åkesson, Hugo Flordahl, Goran Cengic, Avenir Kobetski, Arash Vahidi, Anders Hellgren, past and present members of "Diskreta Klubben". An extra thanks to Kristin for good cooperation over the years. At Chalmers, I also would like to thank Madeleine Persson for always knowing the answers to my questions about the ever changing administrative routines, and to Lars Börjesson for good computer service, especially for handling the connectivity between Volvo and Chalmers.

A special thanks goes to Patrik Holmström and Astrid von Euler at KTH for making the field studies a fun experience and for introducing me into the world of information modeling.

Finally, a thanks to my family, Camilla and Jesper, for their support and for reminding me about the things that are important.

Johan Richardsson

Göteborg, May 2007

Publications

The work presented in this thesis is based on the following papers. Paper 4, 12, 15 and 16 are appended.

1. Richardsson, J. “A Survey of Tools and Methods for Design of Automated Production Plants”, Proceedings of the 33rd Intl. Symposium on Robotics, Stockholm, Sweden · October 2002.
2. Richardsson, J. Fabian, M. “Automatic Generation of PLC programs for Control of Flexible Manufacturing Cells”, Proceedings of the, Intl. Conf. on Emerging Technologies and Factory Automation, Lisbon, Portugal · September 2003
3. Richardsson, J. Fabian, M. “Reuse of Information as a Base for Development and Verification of Control Programs for Flexible Manufacturing Cells”, Proceedings of the 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, Las Vegas, USA · October 2003
4. Richardsson, J. Danielsson, K. Fabian, M. "Design of Control Programs for Efficient Handling of Errors in Flexible Manufacturing Cells" Proceedings of the 2004 IEEE Intl. Conference on Robotics and Automation, New Orleans, USA · April 2004
5. Richardsson, J. Danielsson, K. Holmström, P. Fabian, M. Lennartson, B. "Steps Toward Automatic Generation, Verification and Optimization of Control Programs for Flexible Manufacturing Cells”, Proceedings of the Intl. Conference on Flexible Automation and Intelligent Manufacturing, Toronto, Canada · July 2004
6. Sharique, F. Gore, G. Richardsson, J. Fabian, M. "Verification of a Novel Approach in Control of Flexible Manufacturing Cells”, Proceedings of the Intl. Conference on Flexible Automation and Intelligent Manufacturing, Toronto, Canada · July 2004
7. von Euler-Chelpin, A. Holmström, P. Richardsson, J. "A Neutral Representation of Process and Resource Information of an Assembly Cell - Supporting Control Code Development, Process Planning and Resource Life Cycle Management”, 2nd International Seminar on Digital Enterprise Technology, Seattle, USA · September 2004
8. Danielsson, K. Richardsson, J. Lennartson B. and Fabian, M. "Automatic Scheduling and Verification of the Control Function of Flexible Assembly Cells in an Information Reuse Environment ", 6th IEEE International Symposium on Assembly and Task Planning, Montreal, Canada · July 2005.
9. Richardsson, J. Fabian, M. "Modeling the Control of a Flexible Manufacturing Cell for Automatic Verification and Control Program Generation", 3rd International Conference on Reconfigurable Manufacturing, Ann Arbor, USA · May 2005

10. Richardsson, J. "Development and Verification of Control Systems for Flexible Automation", Thesis for the degree of licentiate of engineering, Chalmers University of Technology, 2005.
11. Richardsson, J. "PLC program input from eM-Simulate ", specification of requirements of information for the PLC program development process, Volvo Car Corporation, 2005.
12. Richardsson, J. Fabian, M. "Modeling the Control of a Flexible Manufacturing Cell for Automatic Verification and Control Program Generation", Journal of Flexible Service and Manufacturing, Volume 18, Number 3, September 2006. (this journal was previously named International Journal of Flexible Manufacturing Systems)
13. Andersson, K. Richardsson, J. Lennartson, B. Fabian, M. "Synthesis of Hierarchical and Distributed Control Functions for Multi-Product Manufacturing Cells", 2nd IEEE Conference on Automation Science and Engineering, Shanghai, China · October 2006
14. Ljungkrantz, O. Åkesson, K. Richardsson, J. Andersson, K. "Implementing a Control System Framework for Automatic Generation of Manufacturing Cell Controllers" Proceedings of the 2007 IEEE Intl. Conference on Robotics and Automation, Rome, Italy · April 2007.
15. Richardsson, J. Andersson, K. Fabian, M. "Reliable Control of Complex Manufacturing Cells" Proceedings of the 2007 IEEE International Symposium on Assembly and Manufacturing, Ann Arbor, USA · July 2007.
16. Richardsson, J. Andersson, K. Fabian, M "Automated Verification and Generation of Control Programs for Flexible Manufacturing Cells ", submitted to the IEEE Transactions on Automation Science and Engineering.

Contents

Introduction

This thesis is about how to make correct control programs for a certain type of systems, namely flexible manufacturing cells, and how to make those programs by minimal effort. By correct is meant that the program will control a system according to a given specification of the behavior of the controlled system. Basically there are two ways to make correct programs. One way is to develop the programs according to a fully specified development process where it is inherent that the resulting program will fulfill the specification. The other way is to more freely develop the program and then check that the program fulfills the specification. The main theme in this thesis is a fully specified development method but, in order to minimize the program developing effort, information reuse is also an important part.

Industrial Problems

The work presented here addresses four industrial problems, lack of tools for offline verification of control programs, lack of information reuse in the development process of a manufacturing system, lack of operator support in error situations, and lack of tools for analyzing the control of complex manufacturing cells.

The lack of tools for offline verification is the main problem and the motive for starting this research project. The need for offline verification is caused by the constantly shortening consumer product life-cycles. Consequently, the introduction of new products into the manufacturing system necessarily becomes more frequent, and connected to each introduction is a performance reduction of the manufacturing system. As shown in Figure 1, the reduced performance is caused by a down-time and a ramp-up-time.

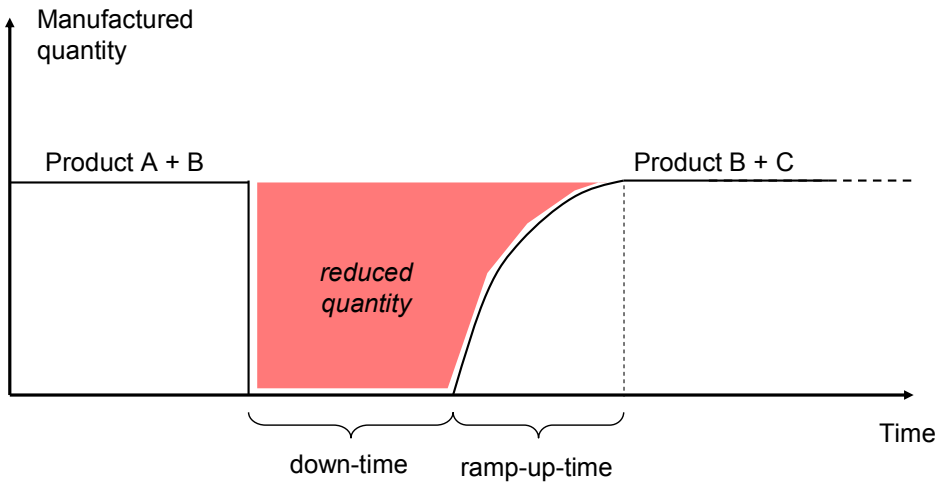

Figure 1. Reduced manufactured quantity when the new product C replaces product A

During the down-time the mechanical equipment is rebuilt and the new control programs are debugged. The ramp-up phase is the time between the start of a rebuilt plant and the time

when it reaches full production rate. During ramp-up there are a large number of errors mainly caused by mechanical devices not being properly adjusted, bugs in the control programs and operators not used to new procedures. Furthermore, the ramp-up time impairs the performance at the worst possible time, when the market demand is peaking after the launch of a new product. Products that cannot be produced due to errors during ramp-up are considered lost. Due to declining market demands it will not be possible to compensate for the loss by increased production once the error rate goes down. Thus, in order to maintain the productivity level and to achieve full cost-efficiency both the down-time and the ramp-up time must be reduced. One way to reduce these lead times is to verify the control programs offline, or to develop the control programs in a way that guarantees fulfillment of the specification.

The need for information reuse is illustrated with an example from the automotive industry, the development of control programs for assembly cells. First a mechanical engineer designs the cell and its control function. Then the same control function is implemented, typically manually, in a robot simulation tool where robot paths are added. After that the mechanical design is verified by a 3D-simulation. Finally the same control function is implemented a third time, in the PLC (programmable logic controller) program. One reason for the multiple implementation of the control function is that contemporary manufacturing systems development largely consists of isolated sub-processes that generate proprietary information; CAD drawings, different models for simulation, mechanical timing diagrams, electrical schemas etc. Efficient information handling is essential in improving this situation.

A problem in many manufacturing systems is that, due to a malfunction, the work in a cell stops without any indication of the cause of the stop. It is then up to the knowledge and experience of the operator or the maintenance personnel to identify the problem. In the automotive industry and other processes with high volume manufacturing systems, it is not possible to spend several hours to find the cause of a stop since the drop in manufactured quantity would be too costly. Therefore, in some cases it is more cost-efficient, but not optimal, to scrap the material and reinitialize and restart the cell. Besides the down time of the manufacturing system this type of problem also causes disturbances in the material logistics and additional costs for new material.

Looking at current implementations of manufacturing cells in the automotive industry, it is not obvious *why* the error indications are insufficient when a cell stops. Therefore, we will assume that there are two different reasons. The first one is that the detection of errors is simply insufficient, so that useful error indication is not possible. An example of this type of error is when a cell stops due to an unfulfilled condition caused by a mechanical malfunction and there is no function in the control program indicating the cause of the stop. The second reason is that the control system and the cell have lost their synchronization. The latter means that the control system expects the cell to be in a specific state while the actual state of the cell is another. In such situations the system usually gets into a locked state without indications of errors.

Regarding insufficient error detection it is clear that in contemporary manufacturing systems of the automotive industry, only a small part of all possible errors are monitored. The reason is that the programming effort to monitor all errors would simply be too big. The programming effort is further increased by the shortened product life cycles, that bring more frequent introductions of new products into the manufacturing system, and consequently more frequent rebuilds of the cells and reprogramming of the error monitoring functions.

Lack of tools for analyzing the control of complex manufacturing cells is a problem, the consequences of which are functional errors and in some cases a lower throughput than expected. Typical properties of those cells are processing of several products at the same time and that it is not possible to separate the cell into smaller functional cells processing one

product at the time, not without reducing the throughput. One reason for the problems connected to the control of this type of manufacturing cells is the typical tools for developing manufacturing control in today's industry, such as, Gantt charts and timing diagrams. These tools are limited in expressing variants. A Gantt chart or a timing diagram describing all combinations of the order of operations of a cell processing several products of different types at the same time would be incomprehensible. A common way to specify the work of a multi product cell is to functionally separate the cell into smaller parts handling one product at a time and describe them individually. Then the PLC programmer has to manually define the interaction between the parts of the cell. A difficult task for a cell containing eight robots that process between one and six products at the same time. The programmer will eventually make a program that works but it is not guaranteed that the programmer will find the optimal way to control the cell.

References to related work are found in the appended papers. Development of correct control programs are referenced in Paper 1, operator support and error handling in Paper 3, and control programs for manufacturing cells in Paper 4.

The Proposed Method

The main idea of the method proposed in this paper is to reuse information from different tools in the development process of a manufacturing system, to process that information in tools for verification and optimization. Then, the processed information is combined with a set of reusable software components, Function Blocks, and converted into PLC-programs.

Figure 2 shows an outline of the proposed method. Different development tools exchange information in a standardized format through a data base, the Manufacturing system data. A certain part of this information represents the control function of the system, the Control Information. Since the information about the control of a manufacturing cell is stored according to a standardized data format, instead of today's practice of being embedded in a control program, it will be possible for new development tools to access that information.

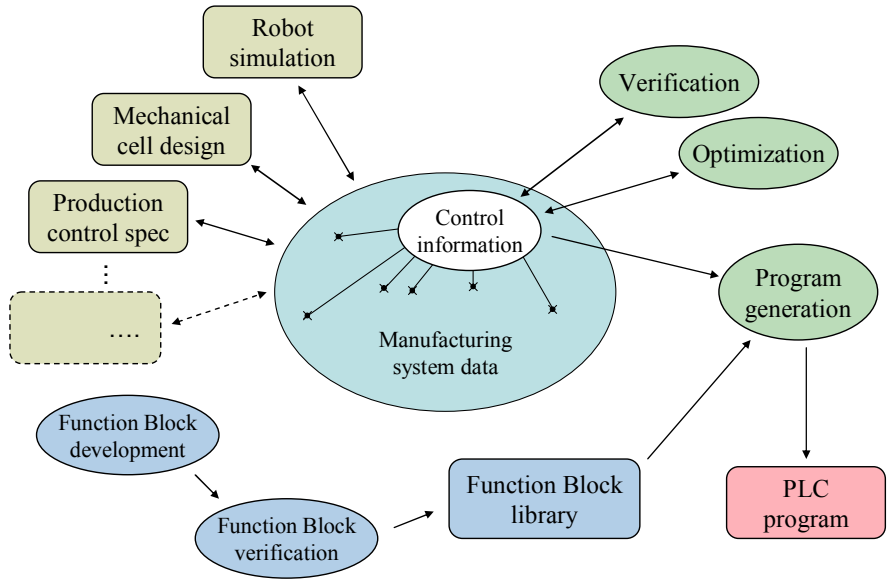

Figure 2 Outline of the proposed development method.

An important aspect of the method is to decompose the information about the control of a cell. Decomposition of the control information means that a complete schedule of a cell, a timing diagram from the mechanical design or a SOP (sequence of operations), from the robot simulation is broken down into a set of operations, each accompanied with scheduling information. The decomposition makes it possible to schedule the operations in a time optimized way and so that collisions between the machines are avoided and that other

requirements of the behavior of the cell are fulfilled. The information needed to describe the control of a cell is divided into eight categories,

1. Declaration of Operations, DOP
2. Relations of Operations, ROP
3. Execution of Operations, EOP
4. Cycle Start Condition, CSC
5. Interlocks, IL
6. Coordinated Operations, COP
7. Mechanical Components and Functions, MCF
8. Function Blocks, FB

The DOP, Declaration of Operations, and the ROP, Relations of Operations, define information about when an operation can be executed while the EOP, Execution of Operations, defines information about the states a machine passes through while an operation is executed. Information about how a machine is to be controlled to go between different states is stored in FB, Function Blocks, as program code. The CSC, Cycle Start Condition, is used sort operations into different work cycles, which are triggered by different combinations of states of a set of the components of the cell. A typical component used to trigger a CSC is the one detecting the type of product in the cell. IL, Interlocks, define the conditions, in terms of states of the components in the cell that have to be fulfilled before a machine can execute an operation without causing any damage to the cell.

DOP, ROP, EOP, CSC, and IL are used as input to the verification process and the output is the COP, Coordinated Operations. The COP defines the order of the execution of the operations and is used to control the cell.

MCF, Mechanical Components and Functions, is a list of the mechanical components in a cell and a list of general functions, not directly connected to the mechanical equipment, needed to control a cell. The MCF is an input to the program generation process and it points out which FB's that are to be instantiated in the PLC program of a specific cell.

The principle structure of the control of a cell is outlined in Figure 3. When a new product has entered the cell it is detected by the Product Data Handler which forwards information about the type of product to the Coordinators. Each Coordinator loads the COP for the current type of product and CSC, and distributes the operations to the machines according to the COP. When a machine receives an order it loads the corresponding EOP and executes the operation accordingly. When finished, the machine signals back to the Coordinator that the operation is completed and the Coordinator can move on in the COP. This is repeated until all operations in the COPs are executed and the work cycle of the cell is finished.

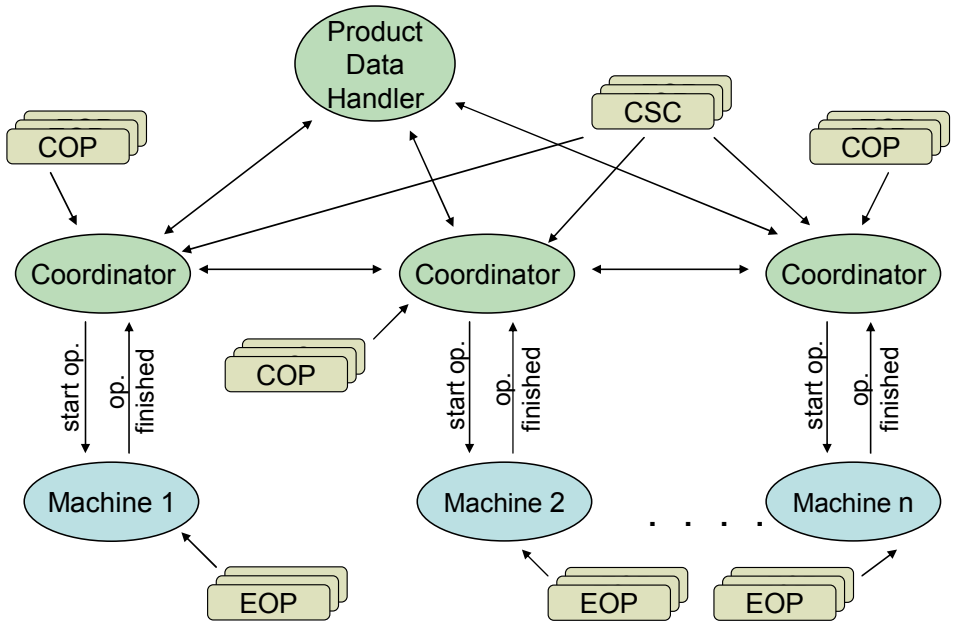

Figure 3 The principle structure of the control of a cell.

The Ph.D. Process

This Section describes the work done throughout the Ph.D. student period, from the start in February 2002 until completion in May 2007. The sectioning of the time comes from Volvo Cars yearly Ph. D. student evaluation, which takes place in the fall. The sources of the references in this Section are found in Publications, page vii.

Feb 2002 – Sep 2002

The project started with a survey of three tools for generation and offline verification of PLC-programs. The work was documented in [1].

Based on limitations found in the survey, a method for efficient handling of the information in the development of a manufacturing system was developed. Generation and offline verification of the control programs is included but only briefly described. The work was documented in "Reuse of Information as a Base for Development and Verification of Control Programs for Flexible Manufacturing Cells", submitted to IEEE Int. Conf. on Robotics and Automation, 2003, but rejected, later rewritten, submitted to, and accepted as [3].

The work done during the period also included making a plan for the research of the coming years. The approach was to start with identifying what information is needed for development of control programs. Next, a standardized format is to be designed for representing the identified information. An important aspect of the format is that the information is to be accessible for other tools in the development process, and in particular new tools for verification, optimization, and program generation. The next step is to find the origins of the input information to the program generation. This will be done by mapping the complete development process of a manufacturing system. Then links or functions are to be developed in order to connect the program generation process with the origins of its input information, and thereby make it possible to automatically calculate consequences in the manufacturing system due to changes on the product.

Sep 2002 – Sep 2003

The part about generation of PLC programs in the previously proposed method was further developed and the result, a method for generation of PLC programs based on information reuse, was documented in [2]

The focus on the previously presented material has been the automatic control of a cell, as a complement a method for generation of control program handling manual interaction was developed. Problems like insufficient indication of errors, lack of support for resynchronization of the cell and its control system and lack of operator support in manual

mode are taken care of. The work is almost finished and a paper is planned to be submitted in the beginning of November 2003.

A study of what information is required for developing PLC and robot programs for a cell was started. The study was made in cooperation with two PhD students at KTH, Patrik Holmström and Astrid von Euler-Chelpin. The two main purposes of the study was to do a formal analysis of the information used for control program development in order to validate the previously experience based assumptions, and to evaluate the possibilities to represent the information required for developing control programs by an established standard.

A master thesis project was started in August. The purpose is to verify the previously proposed method for generation of PLC programs. The master thesis project will consists of three parts. The aim of the first part is to make it possible to control a 3D model of a manufacturing cell by an external control system, a PLC. This can be achieved by modification of an existing model in a new simulation tool, eM-PLC from Tecnomatix. In the second part the PLC program will be implemented. Finally the 3D model and the PLC will be connected so the PLC can control the model and it is then possible to verify the proposed development method for PLC programs.

Sep 2003 – Aug 2004

The work about error handling and operator support in manufacturing cells was finished and resulted in a method documented in [4].

The master thesis project started in August was finished in February. The students implemented a PLC program according to the method proposed in [2]. After some difficulties the students managed to connect a PLC to a 3D simulation tool and by having the PLC controlling a model of a simplified welding cell it was possible to do a first verification of the proposed method for development of PLC programs. The work was documented in [6].

In order to influence and to get useful results from adjacent research areas, a broad study of future possibilities for automatic generation, verification, and optimization of control programs were made. The work was documented in [5].

The previously started study of what information is required for developing PLC and robot programs for a cell was finished. The work was exemplified by designing and implementing an information model according to ISO 10303 for a sample cell from the Torslanda plant. Astrid and Patrik made the design of the information model. The work was documented in [7].

Sep 2004 – Aug 2005

The research during the last year concerned three areas, formalization of the proposed method, evaluation of the industrial ability of the proposed method, and extension of functions of the proposed method.

The formalization and the development of a method for converting control information to state machines was done in cooperation with Kristin Danielsson, PhD student at Chalmers Automation. The work done together was on a principal level and all development of algorithms realizing the ideas was made solely by Kristin. The work was completed in November and documented in [8].

The evaluation of the industrial ability of the proposed method has started as a field study. So far, two cells at the Torslanda plant have been mapped in detail and the control of one of them has been successfully modeled according to the proposed method. The modeling of the control of a cell from the field study was documented in [9].

The extension of functions of the proposed method concerns a capability to process several products in the same cell at the same time. Up to now, the research has focused on cells where the work cycle is completely finished before the next one starts. The extension will allow several overlapping work cycles but still controlled by a set of static schedules. The work started in August and a paper is planned to be finished in February.

The result of the first three years was summarized in a licentiate thesis, [10].

In order to prepare for a prospective future industrialization of the research results, discussions have been held with two suppliers of tools for development of control programs, Tecnomatix and Schneider. Currently Volvo Cars has a possibility to influence the finalization of the new robot simulation tool from Tecnomatix by giving input to about the required information for the PLC-program development process. The work with the new tool from Tecnomatix is documented in [11].

The research plan was revised and the part about mapping the development process of a manufacturing system was removed in favor of the multi product extension.

Sep 2005 – Aug 2006

The research over the last period has involved three areas, continued verification of the usability of the concept, continued work on the extension of the concept for multi product capability, and rework of some of the original functions of the concept.

The evaluation of the industrial ability of the concept has continued with mapping of a third cell at the Torslanda plant. The results have shown an unexpected need for multi product capability. The need is caused by the fact that some cells in normal operation act as single product cells but when the material flow is disturbed, the cells may act as multi product cells; they buffer products within the cell.

The development of the theoretical part of the multi product capability was finished in March and documented in [13]. The work was done in cooperation with Kristin Andersson, who developed the algorithms realizing the ideas. The need of multi product capability found in the field study has motivated continued work with development of the control program model and definition of new information types to make it possible to implement the new algorithms for the multi product capability.

The rework of the concept concerns handling alternatives and the format of the sequence of the operations of a cell, the SOP. The previous version of the concept was able to handle alternative operations dynamically, even though the SOP was calculated beforehand. The problem was that the consequences of all possible outcomes of an alternative operation was grouped together, which may give an operation a higher cost in shape of time or booked work space (zones). If an operation books more zones than necessary or if the scheduled duration of an operation is longer than it actually is, this might lead to a lower efficiency of the cell. So far the field study has not detected any situation where the previous way of handling alternatives would cause a lower efficiency. However, it is not unlikely that such situations may exist, so in order to increase the general usability of the concept, an improvement of the calculation of alternatives has been decided. The improved algorithm will treat the different outcomes of an alternative individually and the efficiency of a cell will not be affected. The new algorithm will be developed by Kristin Andersson and is planned to be finished in October 2006.

The second change of the concept has been to go from a single SOP for all operations of a cell to individual SOP's where each machine in a cell has its own SOP. The purpose of individual SOP's is to allow overlapping work cycles of the machines in a cell; a prerequisite for multi product control. A problem here is that the time optimization algorithm used for the single

SOP does not work for the individual SOP's. A new time optimization algorithm will be developed by Avenir Kobetski, Ph. D. student at Chalmers Automation. The work is expected to be finished in the spring of 2007. The last change of the SOP was renaming it to COP, Coordinated Operations.

An implementation of the concept according to the programming standard IEC 61499 has been done by Oscar Ljungkrantz, Ph. D. student at Chalmers Automation. The work was documented in [14].

On request from an editor of the International Journal of Flexible Manufacturing Systems the paper about the modeling of the control of a cell, [9], was extended and submitted to the same journal, [12].

Sep 2006 – May 2007

The previously started development of the control program model and the information types for multi product capability was completed and documented in [15].

The previously published material on the base concept, (without multi product functions), was summarized, logical gaps were filled and the whole was documented in [16].

Finally, everything was put together in this thesis.

An Alternative Control Program Model

This Section presents an alternative to the Control Program Model, CPM, presented in Paper 1, page 40. The purpose of the alternative CPM is to show that it is possible to convert the Control Information into a PLC program of a more traditional style. In this case, the CPM is based on the three constructs Sequential Function Chart, (SFC), Function Blocks, (FB), and Ladder Diagram (LD) as defined in IEC 61131-3. Figure 4 shows how the alternative CPM realizes the main functions for automatic operation. The CPM presented in Paper 1 will hereafter be called CPM-A, while the CPM presented here will be referred to as CPM-B.

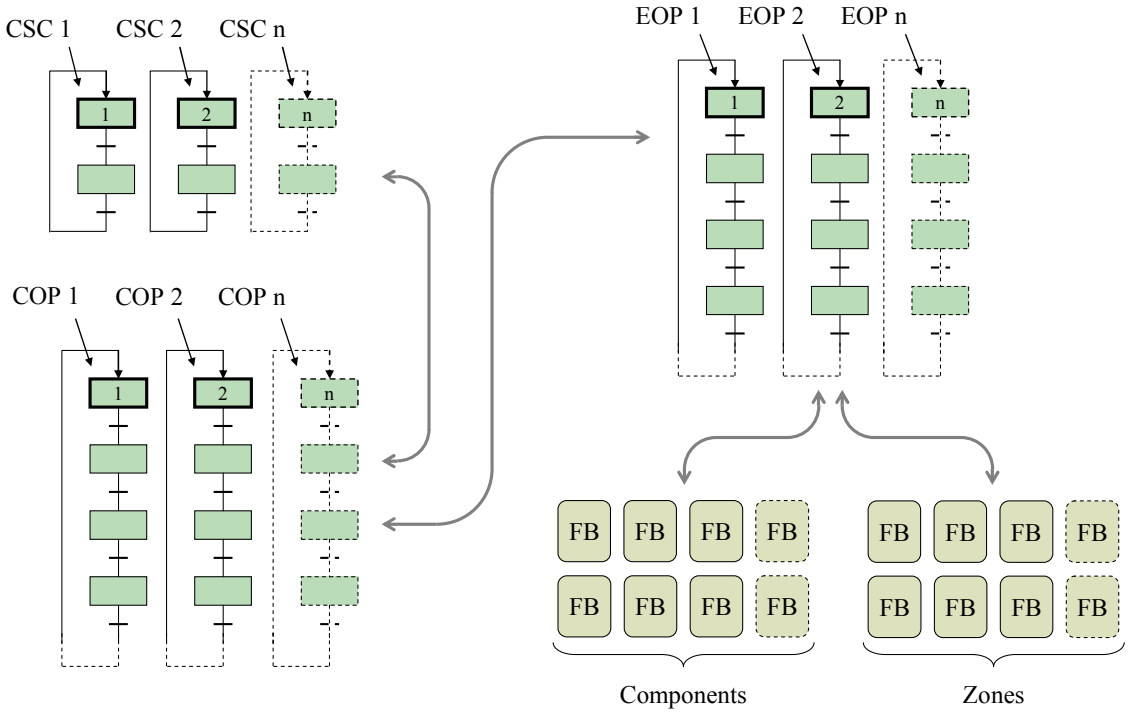

Figure 4 The main parts of a Control Program Model based on IEC 61131-3, SFC, FB and LD.

The object for CSC Service, Cycle Start Condition, of CPM-A is replaced with a set of SFC sequences CPM-B, see Figure 4 top left. One CSC-sequence is generated for each CSC defined in the Control Information for a specific cell. Each CSC-sequence has two steps and the transition condition for the first (initial) step is the condition defined in the corresponding CSC. The action that takes place when step two becomes active is that a variable representing the execution of the corresponding CSC is set true. The transition condition for the second step is that all operations connected to the CSC are finished.

The Coordinators in CPM-A are also replaced with a set of COP-sequences, Figure 4 bottom left, one for each COP and machine. The first transition condition is the start variable of a specific CSC, controlled by the CSC-sequence as described above. The reminder of the transition conditions of a COP-sequence are variables representing that different operations are finished, exactly as stated in the corresponding COP. Thus, the steps of a COP-sequence represent the operations defined in the corresponding COP. When a step becomes active, the variable starting the execution of the corresponding EOP is set true.

At the machine level of CPM-A the Machine Controller is replaced with a set of EOP-sequences (Figure 4 top right), one for each EOP. An EOP-sequence has an initial step, one step for each row in the EOP and one additional last step for the acknowledgement of completed execution of the operation. The first transition condition of an EOP-sequence is the start variable of the COP-sequence with the operation connected to the EOP. The other transition conditions consist of the conjunction of the states of the internal components and the external sensors of each row. In each step corresponding to an action in the EOP, the components to change state are controlled to do so by variables.

The components inside the machine object in CPM-A exists also in CPM-B (Figure 4 bottom right), though slightly changed. A set of parameters replaces the part of the interface concerning message exchange. The components have the same functions as in CPM-A, but orders to adopt different states and feedback about the current state go through variables connected to the parameters.

An Example

Figure 5 shows the order of the execution of the operations of the sample cell described in Paper 2, page 59. Note that the SOP, Sequence of Operations, used in Paper 2, has been replaced with the COP, Coordinated Operations, as defined in Paper 1, page 33. The main difference between a SOP and a COP is that operations of all machines are included in a SOP while a COP holds only the order of operations for one machine. Thus, the presentation of the order of operations differs slightly between Paper 2 and Figure 5.

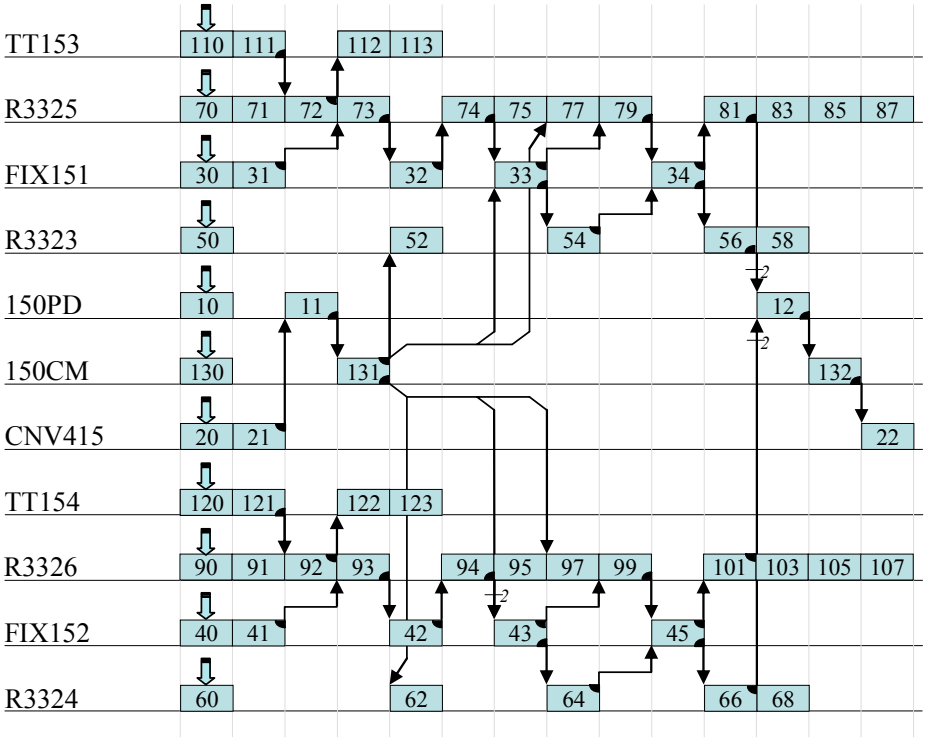

Figure 5 COPs for the example described in Paper 2.

There are 11 COPs in Figure 5, each defining the order of the operations for one machine. The arrows from one box to another indicate predecessor requirements, that is, that the operation from which the arrow starts must be finished before the operation pointed at is allowed to start.

The left part of Figure 6 shows the beginning of a sequence representing the information in a COP for Fixture 152. Figure 6 also shows how the COP-sequence communicates with an EOP-sequence.

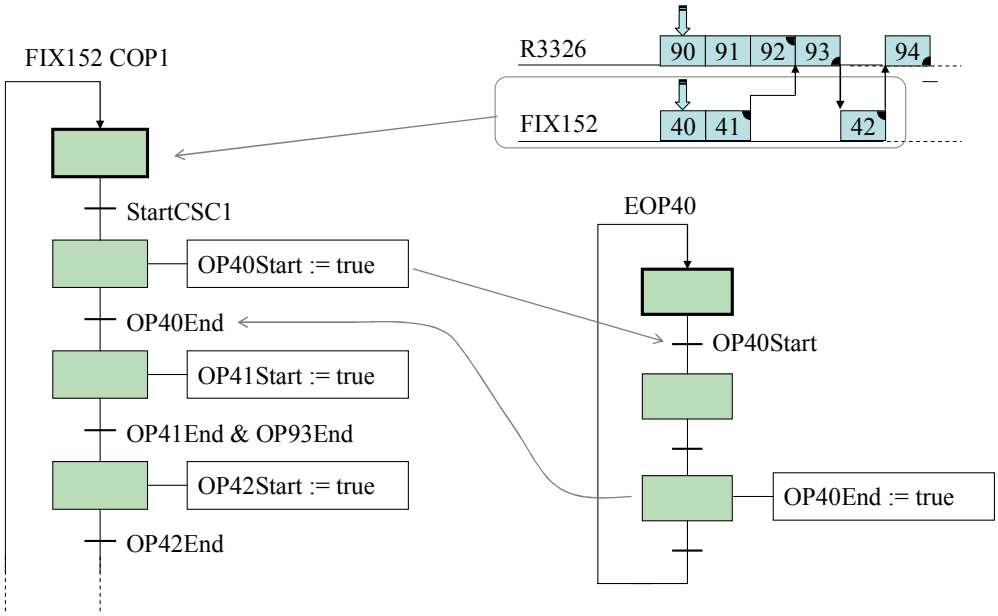

Figure 6 COP-sequence for Fixture 152 with EOP.

Figure 7 shows the first four steps of an EOP-sequence generated from the EOP controlling Fixture 152 to move from its home position to the work position. The EOP is described on page 63 in Paper 2 and the fixture on page 62.

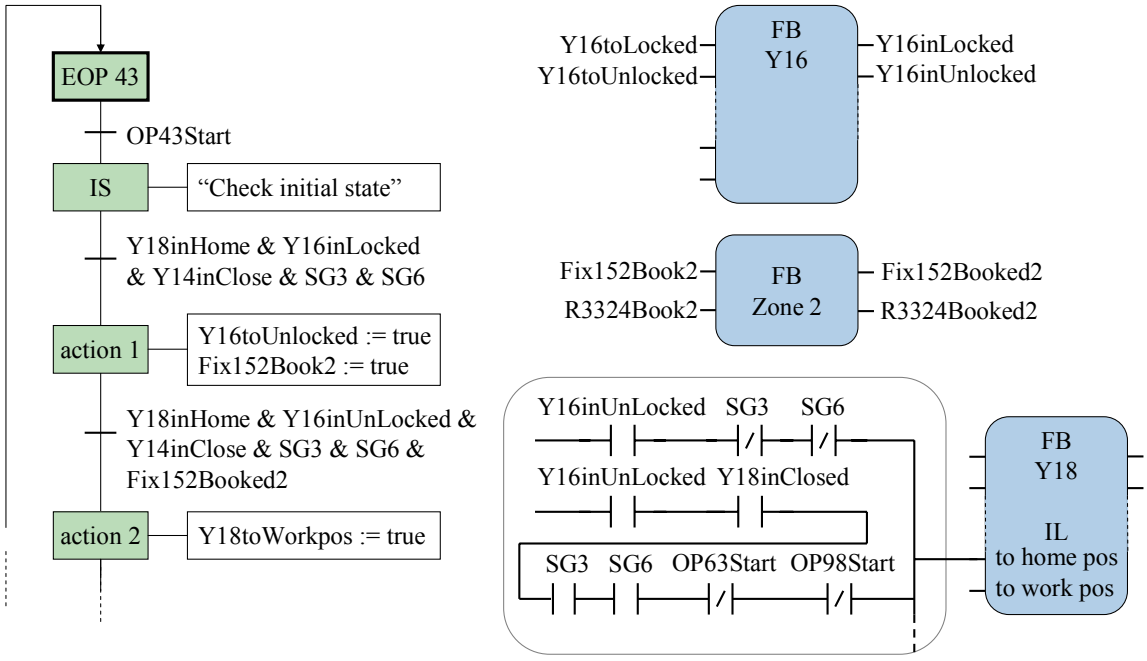

Figure 7 EOP-sequence for Fixture 152 to move from its home position to the work position.

To illustrate the communication between the sequence and the components the FBs of one actuator and one zone are included in Figure 7. Conventionally, the parameters on the left side

of the FBs are inputs and the ones on the right side are outputs. Due to space restrictions, only one of the zones, Zone 2, is mentioned in action 1 and in the following transition condition. The bottom right part of Figure 7 shows how a part of the IL on page 64 in Paper 2 is realized in LD.

A major difference between CPM-A and CPM-B is how the different parts of the programs communicate with each other. In CPM-A the main part of the communication is message-based. Some communication goes through variables but those variables are all standardized. In CPM-B, all communication is achieved through variables.

CPM-B is just one way to model a control program made from the Control Information, and in particular, a straightforward conversion of CPM-A. Many variants are possible, for instance the communication could be implemented as a mix between messages and non-standard variables. Another variant could be to have the EOP-sequences included in the COP-sequences, and thus to get a less hierarchical program.

Included Papers

Paper 1

Automated Verification and Generation of Control Programs for Flexible Manufacturing Cells

Johan Richardsson, Kristin Andersson, and Martin Fabian, submitted to the IEEE Transactions on Automation Science and Engineering.

This paper contains material from paper 2, 3, and 8, in Publications, page vii, and newly developed material making the overall method consistent. The paper is organized in eight sections where the first one presents the industrial problems motivating the work and an extensive part about related research in development of correct control programs. The second section describes the targeted process, its mechanical components, control systems and characteristic properties affecting the control. The third section gives a brief overview of the proposed method and a set of required properties of a system to which the proposed method can be applied. The fourth section describes the information needed to control a cell and a standardized format in which the information is to be represented. The fifth section explains the verification of the information described in Section 4 and the sixth one gives a detailed description of the control program model to be generated from the information described in Section 4. The seventh section argues for the correctness of the proposed method and the eight one, finally, conclude the paper.

Paper 2

Modeling the Control of a Flexible Manufacturing Cell for Automatic Verification and Control Program Generation

Johan Richardsson and Martin Fabian, Journal of Flexible Service and Manufacturing, Volume 18, Number 3, September 2006 (this journal was previously named International Journal of Flexible Manufacturing Systems).

This paper presents the results of a field study from Volvo's Torslanda plant. The information representing the control of an existing cell was mapped and modeled according to the previously presented method.

Paper 3

Design of Control Programs for Efficient Handling of Errors in Flexible Manufacturing Cells

Johan Richardsson, Kristin Andersson, and Martin Fabian, Proceedings of the 2004 IEEE Intl. Conference on Robotics and Automation, New Orleans, USA • April 2004

This paper is based on the method presented in Paper 1 and presents a specialization regarding error situations and operator aspects. Three problems are addressed; insufficient indication of errors, resynchronization of the cell and its control system, and lack of operator support for manual control. The paper proposes a method where control programs with integrated functions for error detection, resynchronization, and support for manual control are generated out of information that already exists in the development process of a manufacturing system.

Paper 4

Reliable Control of Complex Manufacturing Cells

Johan Richardsson, Kristin Andersson, and Martin Fabian, Proceedings of the 2007 IEEE International Symposium on Assembly and Manufacturing, Ann Arbor, USA • July 2007

The work presented in this paper is an extension of the method presented in Paper 1. The extension concerns new functions for multi product control. The paper describes new objects of the Control Program Model needed for the multi product capability. An example cell is used to illustrate problems occurring in multi product cells and to show how the problems can be avoided by the proposed method.

Conclusions and Future Work

The method proposed in this thesis provides results that solve the industrial problems mentioned in page 1, without adding work to the development process. On the contrary, the amount of work will be reduced since a part of the PLC program development will be automated and the time for debugging the PLC program on the shop floor will be drastically reduced, due to the new mathematically based verification process.

The multi product extension of the method makes it possible to calculate the cycle time and the throughput for any given combination of number of products in the cell and their types. This is done from a set of local specifications describing one machine and one product at a time.

The most important result is the modeling of the control of a cell. It makes standardized representation of the control information possible, which enables information reuse and simplifies the automation of the verification and program generation processes. Since the modeling of the control separates information about coordination and execution of operations, the complexity of the information needed for verification of the coordination is essentially reduced. This will significantly reduce the number of states in the verification process and thereby also reduce the effects of the combinatorial state-space explosion problem. The modeling of the control also impacts the handling of error situations in a cell. Since the interlocks are separated from the program code and represented by data, it is possible to design standardized software components that interpret the data and give improved operator support and resynchronization functionality.

Future work involves a continuation of the verification of the industrial ability of the method, development of algorithms for analysis of the result of the supervisor synthesis, an extension of the three-level control hierarchy to a multi-level control hierarchy, and development of algorithms for time optimization of the order of the execution of the operations.

The verification of the industrial ability of the method is to be done by a study of manufacturing cells from assembly plants within the automotive industry. The study will include mapping of control properties of a number of cells from different processes. In the analysis of the study, the control of the cells will be modeled according to the method.

In case the supervisor synthesis does not find a solution that fulfills the specification, the result gives little indication of why the synthesis failed. Therefore it is important to develop a feedback mechanism identifying parts of the specification that might have caused the unsuccessful outcome.

The purpose of the extension of the three-level control hierarchy is to extend the scope of the proposed method from the control of the work inside a cell, as presented in this thesis, to the control of the material flow between cells.

Paper 1

Automated Verification and Generation of Control Programs for Flexible Manufacturing Cells

Johan Richardsson, Kristin Andersson, and Martin Fabian

Submitted to the IEEE Transactions on Automation Science and Engineering

(The paper is reformatted for readability)

Automated Verification and Generation of Control Programs for Flexible Manufacturing Cells

Johan Richardsson, Kristin Andersson, and Martin Fabian

***Abstract*— Consumer product life-cycles are constantly shortening; the automotive industry is an illustrative example. As a consequence, the introduction of new products into the manufacturing system necessarily becomes more frequent. Inherently, this brings a performance reduction for the manufacturing system. The reduced performance is caused by a down-time and a ramp-up-time. During the down-time the mechanical equipment is rebuilt and the new control programs are debugged. During ramp-up there are a large number of errors mainly caused by mechanical devices not being properly adjusted, bugs in the control programs and operators not used to new procedures. Thus, in order to maintain the productivity level and to achieve full cost-efficiency both the down-time and the ramp-up time must be reduced. One way to reduce the down-time and the ramp-up time is to verify the control programs in offline mode. However, efficient and reliable offline verification requires some major improvements of the current development process of manufacturing systems. Information handling and development of control programs based on information reuse are the two most important improvement areas.**

I. Introduction

This paper is about how to make correct control programs for a certain type of systems, namely flexible manufacturing cells, and how to make those programs by minimal effort. By correct is meant that the program will control a system according to a given specification of the behavior of the controlled system. Basically there are two ways to make correct programs. One way is to develop the programs according to a fully specified development process where it is inherent that the resulting program will fulfill the specification. The other way is to more freely develop the program and then check that the program fulfills the specification, or more commonly, that the program does not violate some important parts of the specification. The main theme in this paper is a fully specified development method but, in order to minimize the program developing effort, information reuse is also an important part.

The work presented here addresses two industrial problems, lack of tools for offline verification of control programs and lack of information reuse in the development process of a manufacturing system.

The need for offline verification is caused by the constantly shortening consumer product life-cycles. As a consequence, the introduction of new products into the manufacturing system necessarily becomes more frequent and connected to each introduction is a performance reduction of the manufacturing system. As shown in Figure 1, the reduced performance is caused by a down-time and a ramp-up-time.

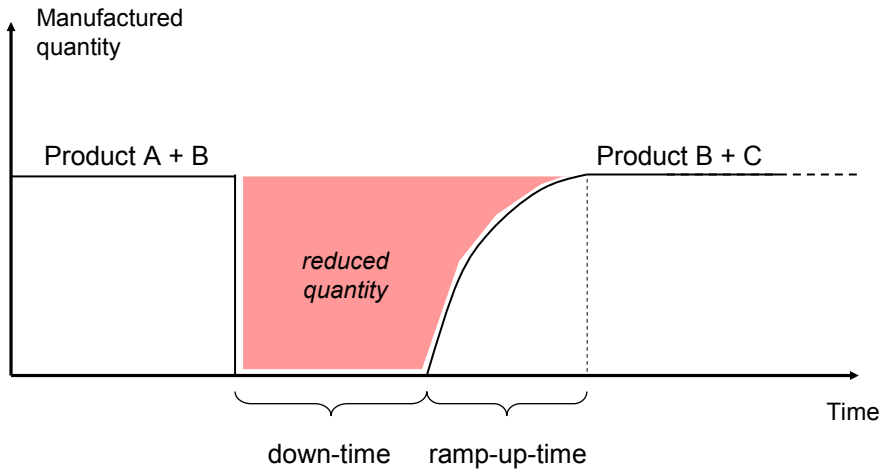

Figure 1 Reduced manufactured quantity when product C replaces product A.

During the down-time the mechanical equipment is rebuilt and the new control programs are debugged. The ramp-up phase is the time between the start of a rebuilt plant and the time when it reaches full production rate. During ramp-up there are a large number of errors mainly caused by mechanical devices not being properly adjusted, bugs in the control programs and operators not used to new procedures. Furthermore, the ramp-up time impairs the performance at the worst possible time, when the market demand is peaking after the launch of a new product. Products that cannot be produced due to errors during ramp-up are considered lost. Due to declining market demand it will not be possible to compensate for the loss by increased production once the error rate goes down. Thus, in order to maintain the productivity level and to achieve full cost-efficiency both the down-time and the ramp-up time must be reduced. One way to reduce the down-time and the ramp-up time is to verify the control programs offline, or to develop the control programs in a way that guarantees fulfillment of the specification.

The need for information reuse is illustrated with an example from the automotive industry, the development of control programs for assembly cells. First a mechanical engineer designs the cell and its control function. Then the same control function is implemented, typically manually, in a robot simulation tool where robot paths are added. After that the mechanical design is verified by a 3D-simulation. Finally the same control function is implemented a third time, in the PLC (programmable logic controller) program. One reason for the multiple implementation of the control function is that contemporary manufacturing systems development largely consists of isolated sub-processes that generate proprietary information; CAD drawings, different models for simulation, mechanical timing diagrams, electrical schemas etc. Efficient information handling is essential in improving this situation.

A. Current Practice of development of correct control programs

Development of correct control programs is a well established research area but industrial applications are still very few, especially for flexible manufacturing systems, targeted by this work. This section briefly describes three methods for development of correct control programs and connected reasons for the low usage of the methods in control of manufacturing cells.

The first method is SCT (the Supervisory Control Theory), [1]. A model of the system and a specification of the behavior are processed together. The result is a logical model, typically a state machine, of the control of the cell that by construct fulfills the specified behavior. There are three assumed reasons for the low usage of SCT in industry. First there is a theoretical

problem; the combinatorial state-space explosion of the logical model when the control of a complete cell is processed together generally results in unacceptable calculation times or even makes the calculations intractable. The second problem is the fact that the output is a logical model that has to be converted into program code. Several applications, for instance [2] and [3] have shown that it is possible to automatically convert the output to program code but the applications concerned small systems and it is not obvious that it can be done in the same way on a larger system. A connected problem is that the applications usually concern only the automatic operation of a system which is about 10% of a program. An important question is then if it is possible, in an efficient way, to represent the functions of a complete program by state machines. The third reason has to do with a need of increased resources for development of a control program. In addition to the work needed in traditional development a detailed logical model of the cell has to be made.

In the second method, Formal Verification ([4] [5]), an existing control program is converted into a format that is interpretable by a verification tool. A model of the system to be controlled is developed and the verification tool will confirm or reject statements, typically expressed as temporal logic formulas. The reasons for the low usage of Formal Verification are similar to the ones of SCT. The two methods share the state-space explosion problem and Formal Verification requires even more additional work since the properties that are to be verified have to be stated individually. However, in contrast to development of one of a kind manufacturing cells the additional work for using Formal Verification is financially motivated in development of high volume products with complex control functions.

The third method is simulation, which in this context means that a control program is verified by having it controlling a 3D model of a cell, while a designer is watching that everything works as planned. Like in SCT and Formal Verification additional work for building a model is needed, but for the simulation of a cell the 3D model can typically be reused from the robot simulation.

A problem with simulation of manufacturing cells is that the efficiency depends on the type of control that is used in the cell, whether it is deterministic or not. In a deterministically controlled cell, where the operations of the machines are always performed in exactly the same way, simulation is a very good verification method. However, most cells are controlled in a nondeterministic way, which may result in a very large number of situations that have to be simulated individually to guarantee the desired behavior of a program. The main reason for having nondeterministic control is to minimize the cycle time of a cell. Deterministic control affects the cycle time negatively in two different ways. Firstly, in the case of deterministic control the programmer has to minimize the cycle time by manually scheduling the order of all operations. This is a difficult task considering the fact that a work cycle usually includes more than 50 operations with typically four to ten operations executed in parallel. In a nondeterministically controlled cell the machines operate individually and are synchronized only at specific instants, for instance to avoid collisions or to pass material between machines. The latter does not guarantee a lower cycle time but letting the machines execute as many operations as possible between the synchronization points is expected to result in a lower cycle time on average. Secondly, if the control of a cell is to be considered as deterministic, so it is possible to verify a control program by simulation of a single work cycle, it is necessary that every state change of the machines in a cell is synchronized. Otherwise, every time a machine is delayed, the cell will be in a new state not included in the verification. A series of state changes of a machine describe an operation and since the duration of operations is different, the deterministic control will cause the machines to wait for each other at the synchronizations and may thereby increase the cycle time.

Information reuse or exchange of information between different tools based on standards is

another well established research area but little previous research results have been found concerning information reuse for development of control programs.

Standardized representation of information has many benefits. First, and for the problems of this paper most important, benefit is the possibility to automatically exchange information between different development tools. This simplifies the automation of computer based activities like verification and program development. The second benefit is improved consistency of the information since information of a certain type only has to be entered only once in one development tool.

Automatic exchange of information between development tools gives three benefits. Firstly, it is possible to guarantee that a verification of a specification of the control also holds when the specification is converted to a control program. Secondly, changes of the system only have to be implemented in one development tool. For instance, when a mechanical component is added to a cell it is defined in a mechanical design tool and information about the new component is accessible by the program development tool. If not automated, the control properties of the new component have to be implemented also in the program development and other properties has to be implemented in other tools. Consequently, automation of computer based activities gives a possibility to effectively evaluate different configurations of a manufacturing cell. The third benefit is that the absence of manual work will reduce the number of errors.

This paper is organized in eight sections. The second section describes the targeted process, its mechanical components, control systems and characteristic properties affecting the control. The third section gives a brief overview of the proposed method and a set of required properties of a system to which the proposed method can be applied. The fourth section describes the information needed to control a cell and a standardized format in which the information is to be represented. The fifth section explains the verification of the information described in Section 4. The sixth section gives a detailed description of the control program model to be generated from the information described in Section 4. The seventh section argues for the correctness of the proposed method and the eight one, finally, conclude the paper.

II. The Process – Flexible Manufacturing Cells

The process that is the main concern of the proposed method is the body shop and the automated parts of the assembly shop in the automotive industry. The body shop of a car factory is usually highly automated and consists of a large number of cells, typically organized in one of two alternative ways. Most common is the *line* concept where the products are moved between similarly structured cells on a conveyor. In the second way of organizing cells, here called *sub-assembly*, the products are moved between the cells by the machines in the cells. Other differences between the two types are that in the sub-assembly the structure of the cells and the number of the machines vary, and the layout of the overall production system is not bound to a straight line.

A. Mechanical Components in a Cell

Inside a cell there are a number of different machines, of which the most common are *robots*, *fixtures*, *racks*, and *pallets*. The robots are equipped with tools for different types of welding, screwing, material handling, and application of glue and sealing material. The task of a rack is to hold a body part in a predetermined position, with good precision, and thereby make it possible for the robot to pick the part in a reliable way. A fixture is a robust machine that clamps one or several parts and holds them while they are processed by a robot. Fixtures can be stationary as the one in Figure 2, or moving between two or more fixed positions. The

moving type of fixture is used when a part is to be welded to the body.

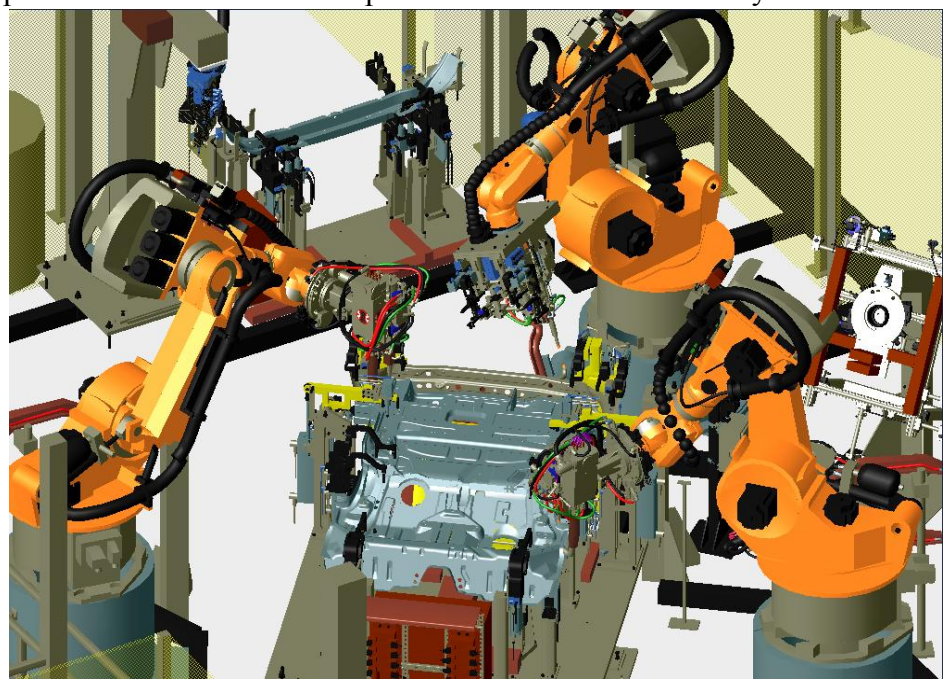

Figure 2 A snapshot from a robot simulation tool where three robots are assembling a part of the floor of a car by putting parts in a fixture and welding them together.

A pallet is a kind of fixture but for larger parts like a side or a floor. A pallet moves the part from cell to cell while a fixture only moves within a cell.

A robot picks the part from a rack, puts it in a fixture which positions the part against the body, and finally a robot welds the part to the body. The positioning is normally not done by robots since they do not have the required precision. Occasionally, in some cells robots do position parts but in those cases the robots do not use predefined positions in the robot program, instead they measure where to position the part.

B. Control Systems in a Cell

In a manufacturing cell, there are a number of control systems. Each robot has its own, and there is one that coordinates the work in the station and controls other devices such as fixtures, racks, and pallets. The control systems for the robots are called robot controllers and the coordinator is typically a PLC. A PLC is a robust computer, specially designed for manufacturing system control. But PLCs and computers with PLC-like behaviour also have their use in other areas, such as logic control of embedded systems.

PLCs have a number of properties that are adapted to the control of machines and production. The three most important ones are described here. First reliability; PLCs hardly ever stop or go down; they execute the same program, continuously, year after year. Secondly, the program development tools offer a simple way to handle complex real time situations. The third, and maybe less obvious property, is the possibility to read the states of variables from the program while the program is running. The main purpose of this function is as a tool to detect errors in the process and to debug the executing code.

C. Process Characteristics Affecting the Control

Due to the high production volume and the complex, product specific, and cost intensive equipment the manufacturing system is realized with a large number of serially connected cells, typically 100 to 150 cells with buffers at certain points. The high volume and the large number of cells result in short cycle times for the cells, typically between 45 and 180 seconds.

The large number of serially connected cells puts three demands on the performance of the

cells. Firstly, to avoid bottlenecks no cell may exceed a certain maximum cycle-time, and to ensure a high degree of utilization of the equipment the cycle time of the cells should not be lower than the maximum cycle-time. Secondly, since a stop in one cell shortly affects neighboring cells and, after emptying the down stream buffer, also other sections of the manufacturing system, a high availability of the equipment in the cell is necessary. A typical value of the availability of a cell with four robots for spot welding is 98,7 %, including transport of the product and the operation of the personal safety system. The availability figure means that all equipment in the cell simultaneously is operational 98,7 % of the time it is expected to operate. Consequently the availability of an individual robot is very high, typically 99,9 %. Thirdly, in order to improve the tracking of causes to geometrical deviations of the product the order of the execution of the operations in a cell should be the same every time a product of the same type is processed in a cell.

Considering the requirements of equal cycle time of the cells and of a fixed order of the execution of the operations it can be concluded that the control of a cell should be as fixed as possible. A drawback of a fixed order of the operations is that a stop of one machine may cause other machines to wait, even though they from a functional viewpoint could have continued their work. However, due to the high availability and the short cycle-times the advantage of a flexible order of the operations in error situations will not outweigh the advantage of a constant order of the operations.

III. The Proposed Method

The main idea of the method proposed in this paper is to reuse information from different tools in the development process of a manufacturing system, to process that information in tools for verification and optimization. Finally the processed information is combined with a set of reusable software components, Function Blocks, and converted into PLC-programs.

Figure 3 shows an outline of the proposed method. Different tools involved in the development of a manufacturing system exchanges information in a standardized format through a data base, the Manufacturing system data. A certain part of this information represents the control function of the system, the *Control Information*. Since the information about the control of a manufacturing cell is stored according to a standardized data format, instead of today's practice of being embedded in a control program, it will be possible for new development tools to access that information. This paper concerns new tools for verification and optimization of the control function and generation of the PLC program.

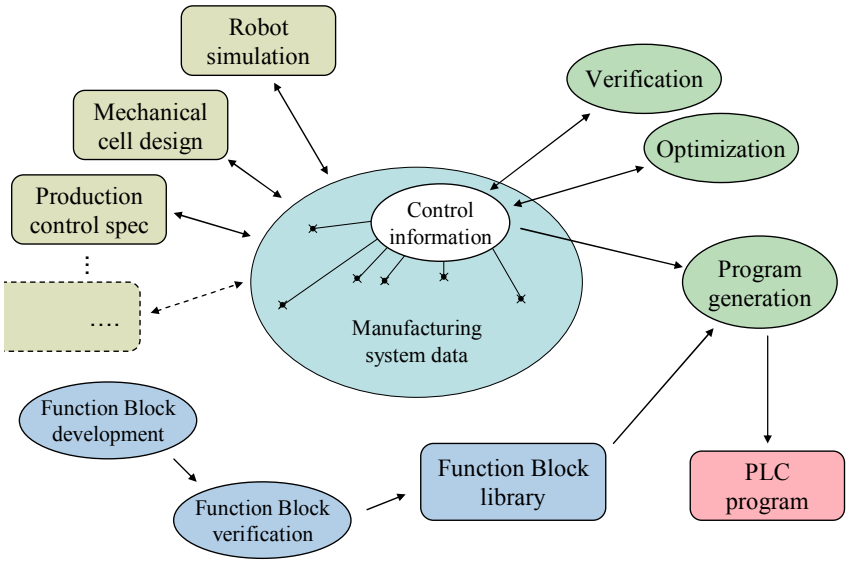

Figure 3 Outline of the proposed development method.

A. Applicability of the Method

A consequence of the standardized representation of the information describing the control of a cell is a limitation of which control situations that can be expressed by the method. A system to be controlled by programs developed by the proposed method must hold the following physical and functional properties,

1. The physical parts of the system can be represented as a number of machines each consisting of a set of active or passive components able to adopt a limited number of states, and by a number of passive resources to be employed by the machines.
    a. An active component is able to receive order about which state to adopt, to change state according to orders, and to give requested or un-requested feedback about its current state.
    b. A passive component is able to give requested or un-requested feedback about its current state.
2. The work performed by the system can be divided into a set of operations each describing a state or a series of state changes of the components of a machine, of involved passive components of other machines and of involved passive resources.
    a. A machine can execute only one operation at a time.
    b. In an operation consisting of a series of state changes each defined state comprises all involved components and passive resources.
    c. During the execution of an operation affecting passive components of other machines, no other machine can affect the same passive components.
3. The work performed by a machine can be described by a limited number of alternative sequences, each describing the order of the execution of a set of operations constituting a part of the work a machine is able to perform.
    a. The order of the operations in a sequence is predetermined and fixed.
    b. All none alternative operations of a sequence are executed while the sequence is active.
    c. One operation out of each set of alternative operations of a sequence is executed while the sequence is active.
    d. The last operation of a sequence always puts the machine in a state from which all sequences of the machine can be started.
    e. All sequences of a machine have a unique start condition consisting of a combination of required states of the components of the system.
4. The desired behavior of the system can be described by conditions for starting each operation of a sequence. The conditions are always logical expression stating which operations, (predecessors), that are to be finished before a specific operation can be started
5. The allowed behavior of the system can be described by combinations of states of the operations and the components of the system, (interlocks), that must be fulfilled in order to allow an active component to change state, and by rules stating the maximum number of machines simultaneously employing a common passive resource. All state changes of the components involved in the description of the allowed behavior of the system must be declared in the operations defined in item 2 above.

Use of the proposed method also sets the following restrictions of the behavior of the system,

6. All machines of the system must be in a predefined initial state before the system

can be started.

7. All machines must complete their sequences before a new sequence can be started.

The complexity of the modeling of the control information depends on how the detection of a completed state change of an active component is done. The complexity increases with the index number of the items in the list below. Only item 8 is considered in the work presented in this paper.

8. The completion of a state change of an active component is detected only by actuators and sensors belonging to the same component.
9. The completion of a state change of an active component is detected by a sensor or actuator not belonging to the same component.
10. Which sensor or actuator to be used for detecting a completion of a state change of an active component varies and is depending on the current state of the cell.

The proposed method will allow the following behavior of a system

11. Several machines may simultaneously execute one operation together.
12. An operation may be executed in several different ways.

IV. Information for Controlling a Cell

This section describes the information needed to control a cell and a standardized format in which the information is to be represented.

An important aspect of the method is to decompose the information about the control of a cell. Decomposition of the control information means that a complete schedule of a cell, a timing diagram from the mechanical design or a SOP (sequence of operations), from the robot simulation is broken down into a set of operations, each accompanied with scheduling information. The decomposition gives the possibility to schedule the operations in a time optimized way and so that collisions between the machines are avoided and that other requirements of the behavior of the cell will be fulfilled. The information needed to describe the control of a cell is divided into eight categories,

1. Declaration of Operations, DOP
2. Relations of Operations, ROP
3. Execution of Operations, EOP
4. Cycle Start Condition, CSC
5. Interlocks, IL
6. Coordinated Operations, COP
7. Mechanical Components and Functions, MCF
8. Function Blocks, FB

The DOP, Declaration of Operations, and the ROP, Relations of Operations, define information about when an operation can be executed while the EOP, Execution of Operations, defines information about the states a machine passes through while an operation is executed. Information about how a machine is to be controlled to go between different states is stored in FB, Function Blocks, as program code. The CSC, Cycle Start Condition, is used sort operations into different work cycles, which are triggered by different combinations of states of a set of the components of the cell. A typical component used to trigger an CSC is the one detecting the type of product in the cell. IL, Interlocks, defines the conditions, in terms of states of the components in the cell that have to be fulfilled before a machine can execute an operation without causing any damage to the cell.

DOP, ROP, EOP, CSC, and IL are used as input to the verification process and the output is the COP, Coordinated Operations. The COP defines the order of the execution of the operations and is used to control the cell.

MCF, Mechanical Components and Functions, is a list of the mechanical components in a cell and a list of general functions, not directly connected to the mechanical equipment, needed to control a cell. The MCF is an input to the program generation process and it points out which FB's that are to be instantiated in the PLC program of a specific cell.

The above described eight categories of information is also referred to as the *specification*, of the control of a cell. Items 1 to 7 are stored as data in Control Information, in Figure 3, while item 8 is stored as program code in the Function Block Library, also shown in Figure 3. Thus, the control of a cell is represented by two different formats, neutral data and program code. One reason for this is the large amount of different types of data that are needed for development of a complete PLC program, which besides the resources in a cell also controls e.g. field bus systems and communication with external systems. It would be a difficult task to design a single information model for the complete set of information representing a PLC program. Another reason is the fact that entering the information for a complete PLC program in the corresponding positions of a data base designed according to the information model would probably be more difficult than writing the program in a modern program development tool.

The principle structure of the control of a cell is outlined in Figure 4. When a new product has entered the cell it is detected by the Product Data Handler which forwards information about the type of product to the Coordinators. Each Coordinator loads the COP for the current type of product and distributes the operations to the machines according to the COP. When a machine receives an order it loads the corresponding EOP and executes the operation accordingly. When finished the machine signals back to the Coordinator that the operation is finished and the Coordinator can move on in the COP. This is repeated until all operations in the COPs are executed and the work cycle of the cell is finished.

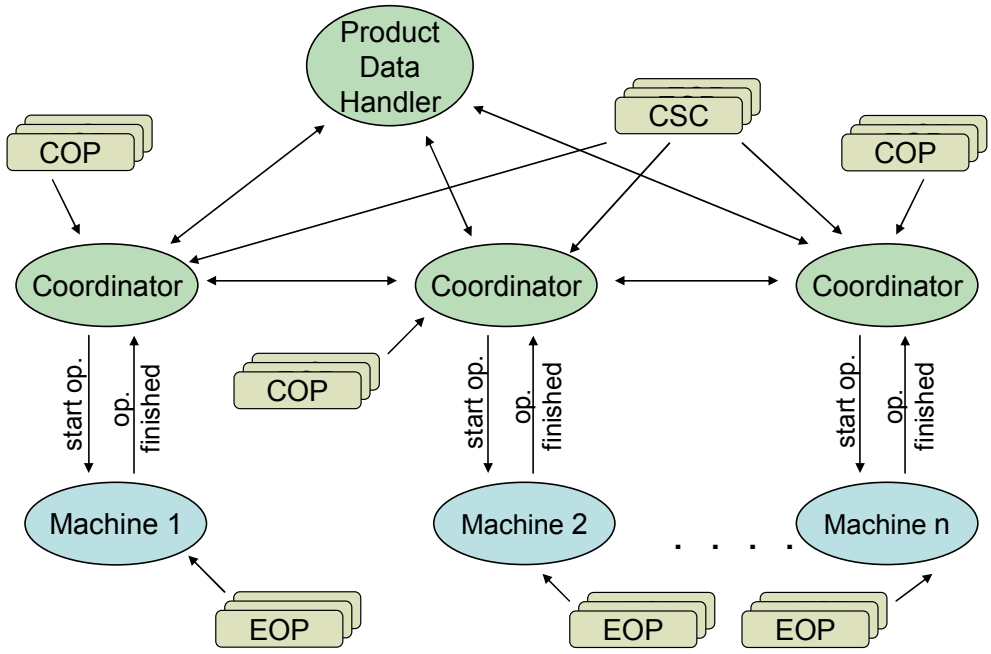

Figure 4 The principle structure of the control of a cell.

A. Declaration of Operations, DOP

An operation is defined as a small part of the work cycle of a cell containing one or a series

of actions of an actuator or a group of actuators. In DOP each operation is declared by the following seven attributes.

1. Identity, an identification of the operation.
2. Comment, a description of the operation.
3. CSC association, the identity of the CSC's the operation is active in.
4. Machine, the machine that is to execute the operation.
5. Zones, a zone represents a physical volume in the cell and to prevent collisions only one machine at a time may be in a specific zone.
6. Duration of the operation.
7. Frequency of the operation, the operation is executed every n-th work cycle.

B. Relations of Operations, ROP

The order of the operations for a machine and conditions for starting each operation are defined in the ROP. With exception for the first operation of each machine, the conditions for starting an operation are always that one or several other operations are finished. If operation *i* has to be finished before operation j is allowed to be started operation i is called a *predecessor* of *j*. If an operation is alternative or parallel this information is also defined in the ROP, by logical expressions of finished operations in the conditions for starting the operations. One ROP contains all operations that are to be executed in a work cycle triggered by a specific CSC. An operation is classified by the following five attributes,

A. Basic
B. Alternative execution
C. Alternative operation
D. Exception
E. Parallel

Type A, a basic operation, is a straightforward operation executed by a single machine in the same way every time. Type B, the alternative execution, is also executed by a single machine but the way it is executed may vary from time to time and is dependent on the current state of some of the components in the cell. There is one EOP defined for each way an operation of type B can be executed in. In an operation of type C, alternative operation, a machine will execute one operation out of several. Which one to be chosen is depending on the current state of some of the components in the cell. An Exception, type D, is equivalent to an Alternative operation but treated separately since it is trigged by an error and therefore not scheduled in a normal work cycle. The last type, E, indicates that the operation is executed by several machines simultaneously.

The format of a ROP is a set of ordered lists where each list states the order of the execution of the operations for a certain machine. Each element in the list represents an operation or, in case of alternative operations, a set of operations.

To each element in the list is connected a logical expression, stating which operations that are to be finished before the specific operation can be started. A default condition for starting an operation is the completion of the previous operation in the same list. In case of alternative operations, type C above, only the fact that some operations are alternative is stated in the ROP. The condition for which of the alternative operations that is to be executed is stored in the corresponding EOP's and used only in the execution of the resulting control program.

C. Execution of Operations, EOP

A cell can on the lowest abstraction level be viewed as merely a set of actuators and sensors, but in the proposed method the actuators and sensors, called *subcomponents,* are aggregated

into *components* and *machines*. Figure 5 shows an example of how a set of sensors and actuators can be aggregated into four components and two machines.

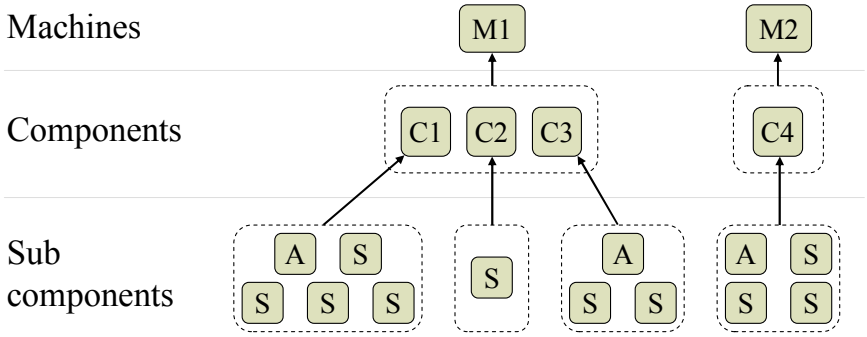

Figure 5 The three levels in the aggregated modeling of the machines in a cell.

From a control point-of-view a machine is represented by a set of independently operating *components*. The components in their turn consist of actuators, variables and sensors. A variable represents a property of a machine that is not possible to detect by sensors only. The control program of a component may include a number of variables but only those representing a property that is needed for an IL or for selection of alternative operations has to be defined in the EOPs. Functionally a variable is treated as an actuator and hereafter not explicitly referred to. If a variable is declared in an IL the variable can change state only when it is ordered to by an EOP. The latter is necessary since the verification process uses the EOPs to make relations between operations that fulfill or unfulfill an IL.

In the modeling of the control, the subcomponents are never explicitly mentioned, their states are reflected by the component they are connected to. For example, the states of sensors detecting if a clamp is open or closed are not included in the specification of the control, only the state of the clamp. The software component, the FB, representing the clamp must then be able to receive orders of which state the clamp is to be in and to give feedback about which state the clamp is in. A component can consist of only one sensor and in those cases the sensor cannot be affected by any of the actuators of the machine. An example of such a sensor is one that detects the presence of a part loaded by an operator or by a robot. Besides the components of a machine, an EOP also contains the zones employed by the machine executing the operation.

The specification of the execution of an operation, EOP, is represented as one or a series of state changes of the involved components. As shown in Table 1, an EOP consists of four parts. The first part is the *internal components* that represent the components of the machine to execute the operation. The second part represents the *zones* employed by the machine. The third part, the *external sensors* represent passive components of other machines affected by the execution of the EOP in question. The fourth part, *external components*, represents both active and passive components of other machines and they are used to select EOP in case of an operation with alternative execution.

	internal components				zones			external sensors			external components		
	C1	C2	…	Cn	Zi	…	Zn	Mi Si	… …	Mn Sn	Mi Ci	… …	Mn Cn
initial state	*a*	*b*		*a*	*		*	*a*		*a*	*a*		*c*
action 1	*a*	*a*		*c*	*b*		*b*	*a*		*a*			
⋮													
action n	*a*	*a*		*a*	*b*		*u*	*b*		*a*			

Table 1 The structure of an EOP.

With the exception of the first row, the format of an EOP is a n*m matrix, where [1, m_i] holds the states component one passes through when the EOP is executed. The corresponding states of the second component are stored in [2, m_i], and so on.

The first row of an EOP, [n_i, 1], defines the supposed initial state of the machine when the operation is to be executed. If the initial state does not match the current state of the machine, an alarm is set of. The type of alarm to be set off and its delay time, if any, is declared for each state in the EOP. In case of an EOP belonging to an alternative operation or an operation with alternative execution, the first row may also hold states of components of other machines. When an operation with alternative behavior is to be executed, the EOP with initial state matching the current state of the defined components is selected. This means that the initial states of EOP's belonging to operations involved in the same alternative behavior must be different.

During execution of an EOP the states of [n_i, 1] are compared with the actual states of the components and zones, and if they match, the components and zones are ordered to adopt the states of, [n_i, 2], called action 1. When the states of the components and zones equals [n_i, 2] the components and zones are ordered to adopt the states of [n_i, 3], action 2. This procedure is repeated until the complete EOP has been executed.

For each machine there must be one EOP defined that represents the initial position of the machine, called the *home position*. All ROPs of a machine must start with the EOP checking the home position. This EOP consists of one row only and its purpose is to ensure that the cell is in its initial state when started, a prerequisite of the validity of the result of the verification process.

Additional information of an EOP, not shown in Table 1, are the identity of the corresponding operation, the type of the operation, and, in case of a parallel operation, the identities of the involved machines.

D. Cycle Start Condition, CSC

In the case a cell has several basic behaviors, for instance to process different types of products, a CSC is used to identify which basic behavior to run at a certain instant. There is one CSC defined for each basic behavior of the cell.

The format of a CSC is a set of n*2 matrices where each matrix represent one CSC. The first entry of the first row is the identity of a particular CSC. All other entries of the first row are identities of a set of the components in the cell. With exception for the first entry, the entries of the second row hold the states of the components that, when equaling the actual state of the physical components, makes a particular CSC valid.

E. Interlocks, IL

There are two main purposes of the IL. From the viewpoint of the proposed method the main purpose is to offer a simple and clear way to describe the allowed behavior of the system. Due to the simple structure of the IL it is assumed that specification errors most likely will appear in parts of the specification other than the IL. From the viewpoint of the process, the main purpose is to prevent operators controlling the cell in manual mode from accidentally starting operations that will damage the cell.

Interlocks are defined for each state change of the actuators, called actions, and in some cases for complete operations. The latter is typical for robots, since many operations of a robot involve only its built in actuators for realizing movements, which never have individually defined actions. In case of ILs belonging to operations with alternative execution one IL has to be defined for each alternative way to execute the operation. Hereafter, when possible, only the actions are mentioned when the source of an IL is discussed. An action is allowed to be executed only if its IL is fulfilled, which means that the states of a set of components, zones, and operations, defined in the IL, equals the corresponding states also defined in the IL.

Table 2 shows the structure of an IL. Expected states of internal and external components, zones, and operations are the four parts of an interlock. Each column represents one component, zone or operation and each row, called *term*, represents one alternative combination of states that can make the IL fulfilled. Terms representing states expected to occur when the cell is in automatic mode are marked. When an IL is evaluated in manual mode any term can be considered, but in the verification process only the terms marked for automatic mode are considered.

The first part of an IL, *internal component states*, shows the required states of the components of the machine executing the action. The second part, *external component states*, also defines required states but of components belonging to other machines than the one executing the action. Normally an IL should be fulfilled throughout the execution of an action, but in some cases the IL can be fulfilled only initially. In order to handle those cases each component state in the IL can be defined as initially valid.

internal component states				external component states					booked zones		operations	
C_1	C_2	…	C_n	M_i C_i	M_j C_j	…	M_n C_n	product type	before	after	not started	not ongoing
a	*		*b*	*a*	*		*		*i,j*	*j*	-	-
a	*a*		*	*a*	*c*		*b*		*i,j*	*j*	*i*	-

Table 2 The structure of an IL

In the third part, *booked zones*, the zones that are going to be occupied by the machine while executing the action are entered in the column *before* and zones still occupied after completing the action in the column *after*. The latter is not a condition for executing actions but is included in the IL since the information in both columns is used to update the zones when the cell is controlled in manual mode. In the last two columns *operations* that affects the possibility to execute an action are entered. Only two of the states of an operation are of interest. From a specification viewpoint, the states are called *not started* and *not ongoing*. The corresponding names from a mathematical viewpoint are *init* and *not executing*, given that a state machine with the three states init, executing, and complete, represents an operation.

Typical operations likely to appear in an IL are those concerning irreversible operations, like welding two plates together. When the state of a component is not relevant for the fulfillment of a term of an IL the corresponding element in the term is marked "∗". In the same way are terms with no requirements on states of zones or operations marked"-".

F. Coordinated Operations, COP

The result of the verification process is a set of COP's. One COP per machine and CSC is calculated. The format of a COP is the same as of a ROP, a sequence of operations where the condition to start an operation is that one or several other operations are finished. The main difference between a COP and a ROP is that the order of the execution of the operations in the COP guarantees that the IL will not be violated and other requirements declared in the DOP and in the ROP will be fulfilled.

G. Modeling Alternatives

There are three possibilities to model alternative behavior of the cell. The alternative behavior of a machine can be represented by two different basic operations connected to two different CSC, or by two alternative operations connected to the same CSC, or by one operation with alternative execution. Common for all three is that the decision of which of the alternative to choose depends on the current state of the components in the cell.

Different CSC are best suited when the alternative behavior involves multiple operations, for instance to process different types of products in the cell. A requirement for using different CSC is that the situation causing the state triggering the choice must be known before the cycle is started.

A typical situation when alternative execution is to prefer in favor of alternative operation is when the work done by the alternative operations appears the same on a higher level of abstraction, for instance when a robot picks a part from a rack. On a detailed level the operation could be carried out in different ways depending on the current position of the part in the rack, but that is not relevant for the high level specification of the work in the cell.

For an operation with alternative execution the consequences of all possible outcomes of the operation are grouped together in the verification process, which may give an operation a higher cost in shape of time or booked zones. If an operation books more zones than necessary or if the scheduled duration of an operation is longer than it actually is, this might lead to a lower efficiency of the cell. This makes the alternative execution best suited for situations when the different ways to execute an operation occupies the same zones and takes approximately the same time. In the calculation of alternative operations, on the other hand, only the execution times and the occupied zones of each alternative are charged to the individual alternatives.

Examples where the information for control of flexible manufacturing cells are modeled according to the proposed method are found in [6] and in [7].

V. The Verification Method

As shown in Figure 3, verification is done at two different points in the outlined development process. The upper verification activity we call *level one verification*. This concerns coordination of operations, and deciding when operations are to be executed. The lower verification activity we call *level two verification*; it concerns the control of individual machines and devices, the execution of operations.

There are two reasons for separating the verification into two parts. The first one is the difficulties to automatically generate logical models of the PLC program. As explained in Section IV, it is difficult to represent the complete set of information needed for a PLC

program in a standardized way. Consequently, it is also difficult to establish an automated information exchange with the verification tool. The second reason is the combinatorial state-space explosion of the logical model when the control of a complete cell is processed together. In the worst case the number of states grows exponentially with the number of components. This may make the verification practically intractable if the entire system is regarded at once. The two level verification method does not entirely solve the combinatorial state-space explosion problem but the effects are reduced in a way that so far has been sufficient. [8] and [9] describes other approaches to fight the combinatorial state-space explosion.

A. Level One Verification

The verification method used in level one is based on the Supervisory Control Theory, [1], often referred to as synthesis, and just principally described in this paper. A detailed presentation of the core functions is found in [10]. Figure 6 shows how a part of the control information described in Section IV, the DOP, ROP, CSC, EOP, and IL is transformed into state machines, processed and converted back to neutrally stored data.

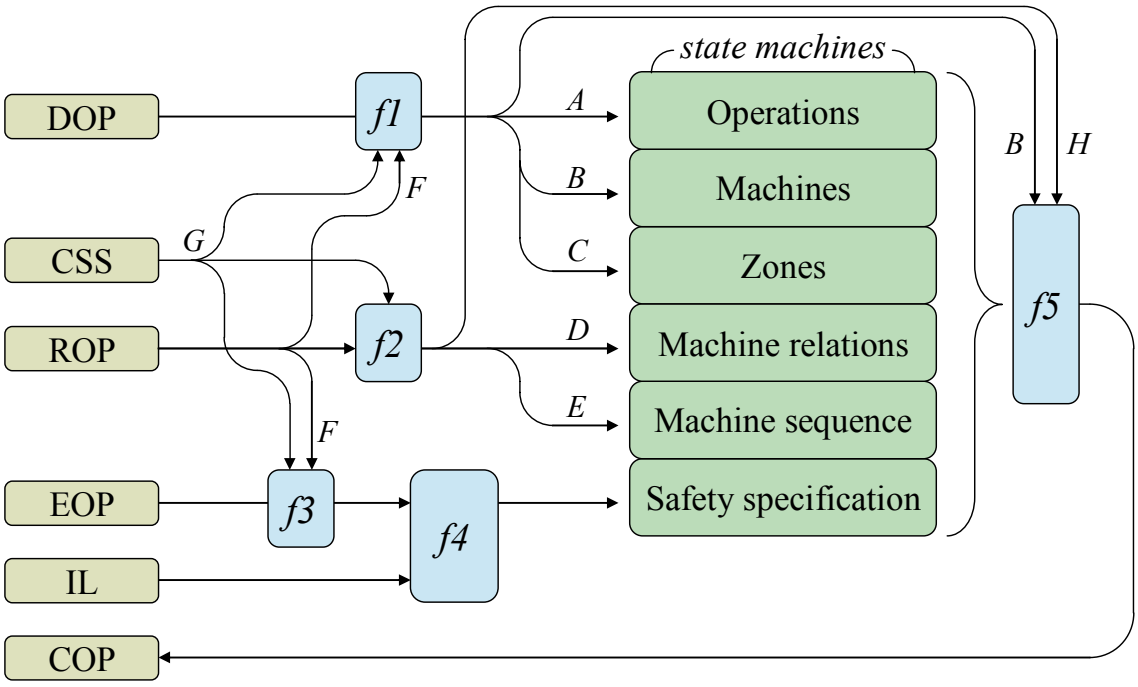

Figure 6 Information flow of the level one verification process

The following list shows the part of the control information used in the different functions in Figure 6.

A. the identity of each operation
B. the identities of each operation and the machine it use
C. the identities of each operation occupying a zone, the corresponding zone, and the operation releasing the zone.
D. the identities of each pair of operations comprising a predecessor-successor relation between two machines.
E. the predecessor-successor relations of the operations for each machine and the identities of each group of alternative operations.
F. the identities of each set of operations comprising a parallel execution of an operation.
G. the identity of the CSC to be processed.

H. the identities of each set of operations comprising a parallel execution of an operation and the identity of each operation with alternative execution, type B in Section IV.B.

Common for *f1*, *f2*, and *f3* in Figure 6 is that the operations to be processed are selected by the identity of the corresponding CSC.

B. Generation of State Machines from the DOP

Three different sets of state machines are generated in *f1*. The set representing the operations (Figure 6 top) contains instantiations of the state machine in Figure 7, left. For each selected operation a state machine with three states is generated. The events are labeled with the identity of each operation.

In the generation of the state machines representing machines and zones two generic rules of the behaviour of the cell is included in the generation process. The first rule is that a machine may only execute one operation at the time and the second rule is that a zone may be booked only by one machine at the time. The state machines to the right in Figure 7 show how the two rules are realized. In the verification tool the state machines are synchronized and a basic rule for synchronizing state machines is that an event is allowed to occur only if all involved state machines are in states from which the event can occur. Looking at Figure 7, if the event *start O1*, start operation 1, has occurred, the only event that can occur in all state machines is *endO1*. Consequently, the machine cannot start another operation and the zone cannot be booked by another operation, which was the purpose of the two rules.

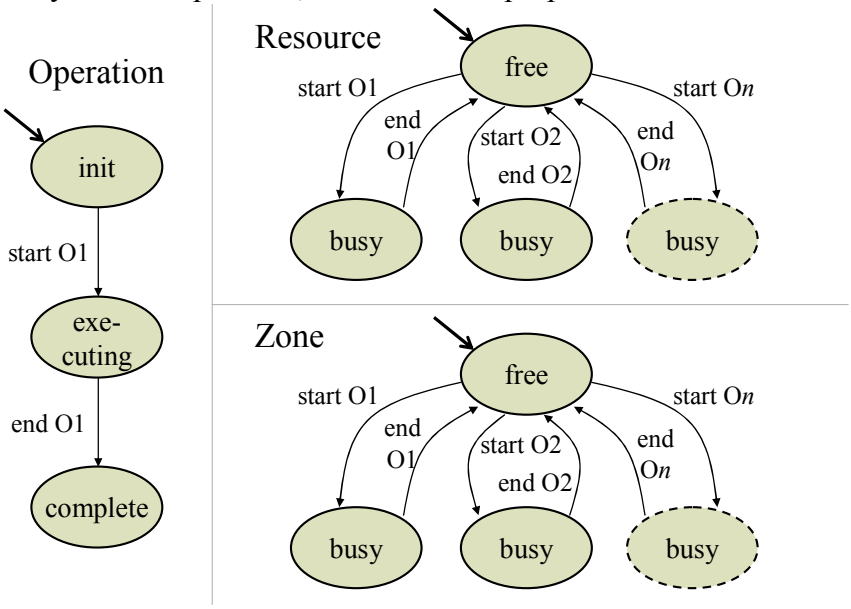

Figure 7 Examples of state machines generated from the DOP.

C. Generation of State Machines from the ROP

Two different groups of state machines describing the relations of the operations, ROP, are generated in *f2*, Figure 6. Before generating the state machines all parallel operations are relabeled so that they all have the same name. Synchronization will then ensure that a set of parallel operations are scheduled so they can be executed simultaneously. The parallel operations are also relabeled in f1 and in f3.

Operations with alternative execution, type B in Section IV.B, are transformed into a set of *Alternative operations*, type C, where each alternative way to execute the primary operation forms one *Alternative operation*. This is done to prevent false deadlocks that may occur in *f5* if the interlocks connected to the alternative EOPs are requiring contradictive states of the same components, to allow execution.

The *Machine sequence* is one of the two groups of state machines generated in *f2*. One *Machine sequence* represents the order of the operations of one machine in one cycle connected to a specific CSC. The list of operations defined in the ROP, Section IV.B, is converted into a state machine where the order of the operations is represented by the start and end events of the operations, as shown in Figure 8. Alternative operations are included in the Machine sequence as branches.

The *Machine relation* is the second group of state machines generated in *f2*. Each predecessor-successor relation of operations belonging to different machines forms one *Machine relation*. The meaning of the state machine in the lower part of Figure 8 is that operation *i* must be finished before operation *j* can be started, operation *i* is a predecessor of operation *j*.

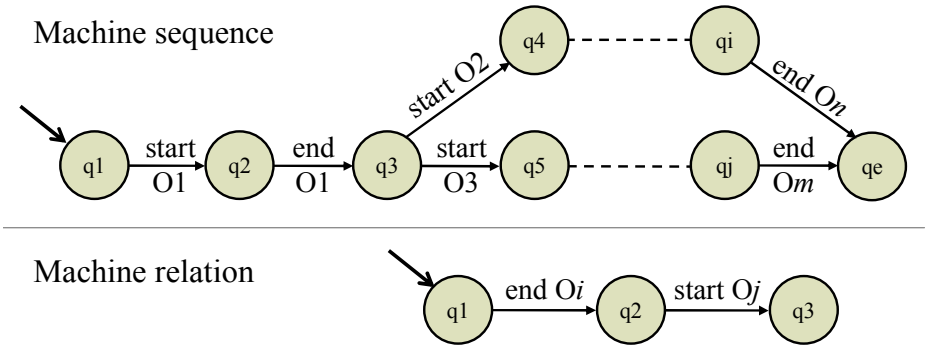

Figure 8 Examples of state machines generated from the ROP

D. Generation of State Machines from the EOP and the IL

The *Safety specification* is the result of *f3* and *f4* in Figure 6. As previously mentioned, in *f3* the EOP's connected to the operations of the current CSC are selected, those belonging to an operation with alternative execution are connected to the new Alternative operation, and those belonging to parallel operations relabeled.

The processing of the EOP's and the IL's in *f4* involves three steps. First it is checked that the EOP's fulfill the internal IL. For each state change of an actuator defined in an EOP it is checked that the states of the internal components and the zones before the state change is executed fulfill the required states defined in the corresponding IL. All terms of an IL that are fulfilled by the EOP are marked for further processing in step two and three.

In the second step of *f4,* the external interlocks are related to operations that make an IL fulfilled or unfulfilled. The resulting state machine is generated by an evaluation of the external IL of the, in step one, marked terms of the interlocks of each state change of an actuator defined in an EOP. First the components of the external interlocks are identified. Then, all EOP's are searched and those that contain state changes of the identified components that fulfill or unfulfill the external interlocks are marked as safe and unsafe operations, respectively, for the operation connected to the specific actuator. The left part of Figure 9 shows an example of a state machine generated by the method described in step 2.

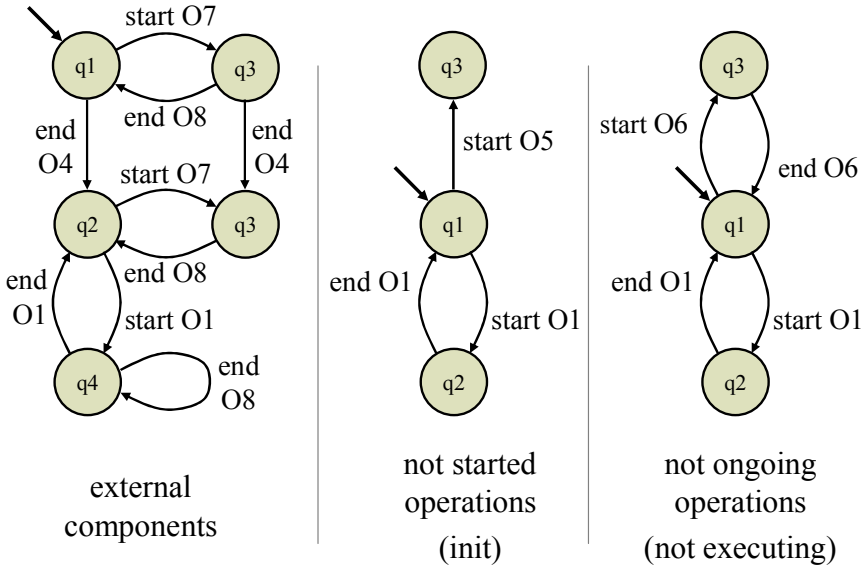

Figure 9 Examples of state machines generated from the EOP and the IL

The meaning of the left part of Figure 9 is that operation 1 uses an actuator which external interlock is fulfilled if operation 4 is finished, and operation 7 is not started, or operation 7 is started but operation 8 is finished. The purpose of the repeated events of operation 4, 7, and 8 are to allow these events to occur whenever possible with respect to operation 1 that is the concern of this state machine. The only restriction from operation 1 is that the start of operation 7 is blocked as long as operation 1 is ongoing.

In the third step of *f4* the required states of other operations are transformed into state machines. Operations in the marked terms classified as *not started* and *not ongoing* are converted into the state machines shown in the middle and in the left part of Figure 9, respectively. The meaning of the middle one is that operation 1 can be started only as long as operation 5 is not started, and the meaning of the left state machine is that operation 1 can be started only when operation 6 is not ongoing.

E. The Level One Verification Process

After converting the control information to state machines the actual verification takes place, *f5* in Figure 6. The previously described state machines are all synchronized and further processed in a tool, [11], based on Supervisory Control Theory, [1]. There are two possible outcomes of the verification process; a successful one and a failure. In case of a successful outcome the result is a supervisor that contains a set of states and transitions, that brings the cell from the initial state to a final state where all required operations are executed, without violating neither the IL nor the two rules applied in *f1*. By the design of the verification algorithm the supervisor will also be non-blocking. The second outcome of the verification process, failure, indicates a contradictive specification and that it is not possible to find an order of the execution of the operations that fulfills the requirements stated in the state machines used as input to the verification process.

In order not to violate the specified behavior of the cell and to avoid blockings, the verification process might add further restrictions on the order of the execution of the operations, as additional predecessors of the operations. This means that the verification process at some extent also can be viewed as a scheduling process. The degree of scheduling in the verification process depends on the degree of freedom in the specification and it decreases with an increased number of predecessor-successor relations between the operations of different machines.

Another consequence of the verification process is the possibility that the supervisor can include arbitrary order of some operations, which, as stated in Section II.C, is an undesired property. An example is when two machines use the same collision zone. In this case it is possible that the supervisor can allow the first or the second machine to start using the zone. The arbitrary order can be removed by adding additional requirements in the ROP, but a better way is of course to calculate the time optimal order of which machine that is to start using the zone. The latter is addressed in ongoing research.

Given a successful outcome of the verification, the next step of f5 is to generate the COP. Using the mapping between operations and machines, defined in the DOP, the order of the operations and their start conditions are extracted by searching the supervisor from its initial to its final state. The relation extraction is done once for each machine.

In the final part of f5 all previously modified parallel operations and operations with alternative execution are altered back to their original form. The alternative operations are regrouped and transformed back to their respective original single operation with alternative execution and the parallel operations are relabeled with their original identity. A consequence of the latter is that any operation having a parallel operation as predecessor will now have a set of operations as predecessors, the involved parallel operations.

F. Level Two Verification

Which method to be used for verification at level two is a matter of further research but simulation, as described in Section I.A, is expected to be a suitable method. The main reason for this assumption is the fact that the verification of the control of the execution of an operation can be divided into verification of the behavior of the individual components executing the operation. Since the control of a component mainly involves bringing the component between different states it is to be considered as deterministic control and thereby suitable for simulation.

VI. The Control Program Model

The information presented in Section IV can be converted into PLC programs with different structure, from traditional programs written in Ladder, IEC 61131-3, to modern object oriented programs, IEC 61499. Implementations of the in this paper described control program model are found in [12], according to IEC 61131-3 and in [13], according to IEC 61499. This paper presents a model of a control program that is object oriented with interpreting objects. The basic idea is to "assemble" the control program out of objects representing the mechanical components in a cell and basic cell functions. These objects are instantiated from a library of standardized software components. The control function, that constitutes the "glue" that coordinates the objects, can then be loaded as data to the PLC and interpreted by a set of standardized software components.

There are three main reasons for choosing the program structure with interpreting objects. Firstly, since the relevant parts of the information used as input to the verification process and its result, the COP, is interpreted in its original format by the control program the verification of the control information is also valid for the control program. Secondly, the chosen program structure gives a high degree of reliability and flexibility. Thirdly, from research point of view the interpretable program structure is easier to analyze and to realize regarding the program generation function than for a program structure where the control function is embedded in the generated program code.

The program structure is object oriented [14] but has a hierarchical control structure. Modern PLC programming languages do not fully support object oriented principles. At best, the only construct available for implementation reuse is containment rather than inheritance.

In [15] the hierarchical control concept is questioned. One reason is that it may cause a communication bottleneck, but in this case this is not likely. Since the order in which the messages are sent is calculated so that a machine is always ready to receive it when the message arrive. There will be no rejected requests.

In this section the software components are called objects, due to the object oriented structure of the control program model. The corresponding name in a PLC program is function block, FB.

The description of the objects in the control program model is divided into three parts, the cell level, the machine level and the component level.

A. Objects at the Cell Level

Figure 10 shows the objects at the cell level, the main communication links between the objects, and the information involved in the control of the cell. The communication between the objects is mainly done by messages but predefined variables are also used.

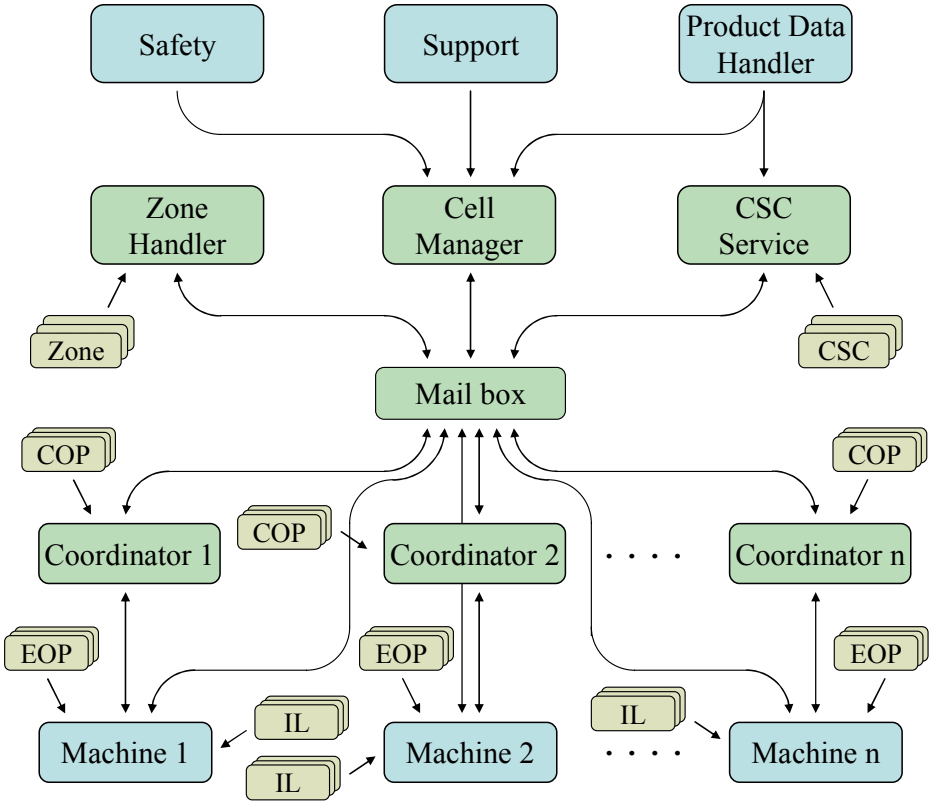

Figure 10 The objects at the cell level.

The *Safety* object detects all emergency stop push-buttons, photocells, and similar equipment used to prevent people from entering the cell while it is running. Communication with the *Cell Manager* object is done via one variable only; when all safety devices are unaffected the safety object holds the safety_ok variable high. The Support object communicates with the Cell Manager object in a similar manner. When the supply of electricity, compressed air, and cooling water are working properly, this object holds high a support_ok variable monitored by the Cell Manager.

To detect and forward information about which type of product to be processed next is the task of the Product Data Handler. The Support, Safety, and the Product Data Handler are not standardized objects but they are supposed to be composed by a set of standardized objects.

The *Zone handler* prevents the machines from colliding with each other. The Zone handler

holds a number of zones, described in Section IV.A. Before a machine is allowed to enter a zone it must book the zone. In the same way, the machine must release the zone when it has left it. During normal operation in auto mode, the Zone handler is not necessary, since the operations are scheduled to avoid collisions. However, to enable a safe switch from automatic mode to manual the zones are updated also in automatic mode.

The Cell Manager handles general functions in the cell, like operation modes and start and stop of the machines. When to start a new cycle and the completion of a cycle is also detected by the Cell Manager. Which CSC to use when a new cycle is started is detected by the *CSC Service*. The current states of the components in the cell and the type of product to be processed are compared with those defined in the CSCs and the CSC matching the current state is selected. The *Mail Box* is a mutual message queue used to pass data between the objects. As described in [16], a mail box is possible to implement in any PLC.

The *Coordinators* order their respective machines to execute the operations defined in the COP, and coordinate the machine with other machines when so stated in the COP. The Zone handler, the Cell manager, the CSC service, the Mail box and the Coordinators are all standard objects.

Among the specific objects at the cell level the Machine object is the main concern of this paper. Specific objects for the Product data handler, the Support, and the Safety objects are supposed to be handled in the same way as the machine objects but their internal structure is not treated here.

B. The Objects at the Machine Level

Like the objects in a cell, a machine object also consists of a standardized framework of objects, with some specific objects adapted to the specific machine. There are two types of machine objects, one for machines that have their own control system and one for those without. Figure 11 shows the internal objects of a machine object without own control system.

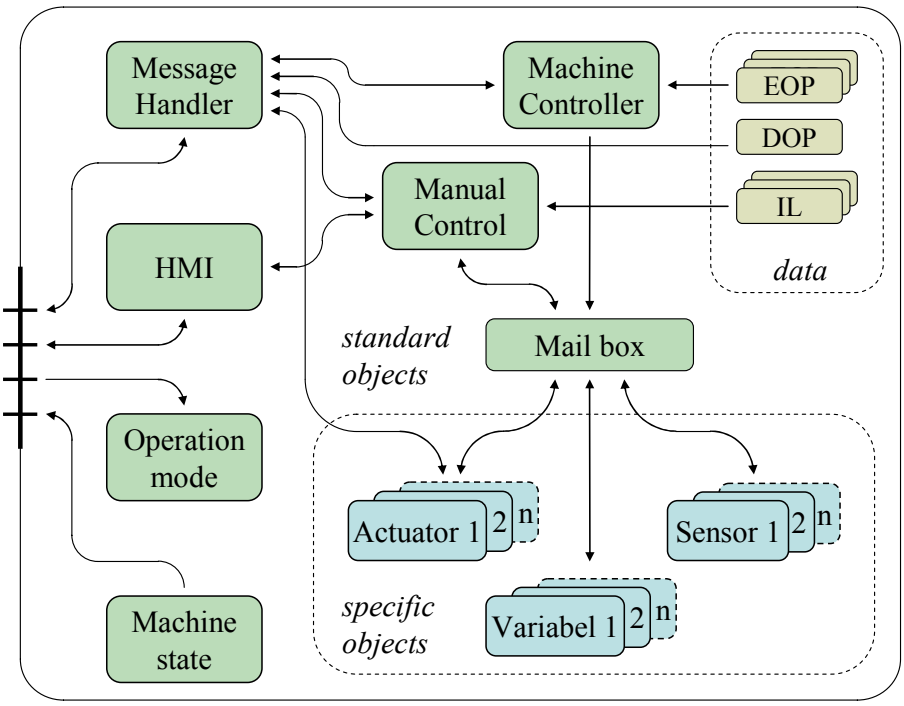

Figure 11 The internal objects of a Machine Object.

From the left we first have the standardized interface with four connections, which handle all communication between an object and its surrounding environment. The standardized interface is a prerequisite for generating the program from instantiations of objects in a

standardized framework. The communication through the interface can be divided into three parts. The first part handles the message communication and the second part concerns the interaction with an operator, via a Human Machine Interface, HMI. The third part involves the control from the Cell Manager and the feedback to it

The *Message Handler* manages the message exchange with the other objects. Internally the Message Handler forwards orders to be executed to the Machine Controller and receives back acknowledgements of completed operations. Requests about the state of components belonging to other machines are communicated through the Message Handler from the Machine Controller and from the *Manual Control* object. As indicated in Figure 11 a part of the DOP is used by the Message Handler and it is the identities of each operation and the machine it employ. This information is used to identify which machine a request about the state of an operation is to be sent to and it is done when an action is to be executed manually and the external interlock of the action contains requirement of the state of an operation. The third internal object type the Message Handler communicates with is actuator objects and this communication concerns synchronization of actuators in different machines while executing a parallel operation.

How the communication with the HMI is to be implemented depends heavily on the properties of the particular PLC and the particular HMI to be used in a specific application. Therefore, this part is just principally discussed. A solution fitting the method presented in this paper is that the graphics of the HMI are instantiated in the same way as the machine object and that it then would be possible to link them together, graphical object to control object. Such systems are offered by some suppliers of PLC and HMI equipment.

The last two objects connected to the interface, *Operation Mode* and *Machine State*, handle the control from and the feedback to the Cell Manager. The Operation Mode object offers variables to the other internal objects indicating the current operation mode and when the machine is allowed to operate. The Machine State object uses variables to summarize the state of the components of the machine. If a machine has an error that may affect other machines the Machine State object signals the error to the Cell Manager which stops the cell.

When the machine is to execute an operation, a message comes in to the Message Handler that interprets the message and forwards relevant parts to the *Machine Controller*. The corresponding EOP is loaded by the Machine Controller, which starts to control the components, actuator, variable, and sensor objects, according to the states defined in the EOP. The communication between the Machine Controller and the components is done by message exchange, over a local mail box inside the Machine object.

The first row of the EOP is the supposed initial state of a machine when a specific operation is to be executed. The initial states are compared with the states of the components, and if they do not match an alarm is set off and the machine is stopped. If the states match the Machine Controller will order the components to adopt the states of row two, called Action 1. When the states of the components are the same as the ones in row two, Action 1 is finished and the Machine Controller will order the components to adopt the states of row three, Action 2. This procedure is repeated until all rows have been executed. Then the Machine Controller sends a message back to the Coordinator, via the Message Handler, to acknowledge that the operation has been executed.

When the Machine Controller is instantiated, it is dimensioned for the number of actuators, variables and separate sensors of the machine in question. In order to instantiate the none-standard components, actuator, variable, and sensor objects, in an efficient way they are equipped with a standardized interface.

The last part to describe at the machine level is the Manual Control-object. It offers a secure way for operators to manually execute actions or operations. When an action is requested it is

checked that its IL is fulfilled before the action is started. First the states of all parts of the IL are colleted. Only the current states of the operations and the internal components are requested, by messages, but the states of the external components are monitored during the execution of the action. The states of the zones are evaluated by booking all zones and those possible to book are marked booked and those not free are marked occupied. If a zone was booked by the machine that is to execute the action the zone is marked as previously booked. After collecting the states of all variables defined in the IL the different terms of the IL are evaluated, and if one of them is fulfilled the action is started. If the states of the external components changes so that the IL no longer is valid the Manual Control-object is to set off an alarm and the cell stops. When the action is completed the zones present in the before column of the IL but not in after column are set free. If none of the terms were fulfilled the monitoring of the external components are turned off and the zones marked booked and not previously booked are set free.

At the machine level all objects except the actuator, variable, and sensor objects, are standardized.

The type of machine object that has been described so far is the one without its own control system. The machine object for a machine with its own control system, a robot for instance, has neither Machine Controller nor internal mail box. The main purpose of such an object is to be an interface between the PLC program and the control system of the machine in question.

C. Objects at the Component Level

Like the objects in a cell and in a machine object, the actuator object, Figure 12, also consists of a standardized framework of objects, but the actuator object has only one specific object, called the *Sub Component*. Another difference to the cell and the machine object is that the main part of the communication between the standardized parts and the specific object is done by a set of standardized variables. The only exception is the execution of a parallel operation where message exchange is used to synchronize actuators in different machines.

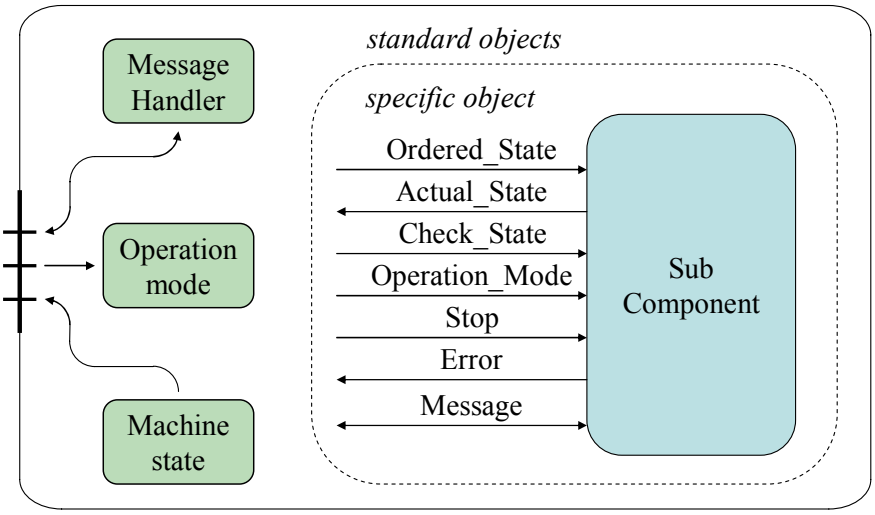

Figure 12 The internal objects of an actuator object.

The Machine Controller can send three different messages to an actuator object: *state request*, *state check*, and *order*. When a state request is received the actual state of the sub component is read and sent back to the Machine Controller. The result of a state request is used when the Machine Controller has a branch situation or in manual control when an IL is evaluated. An option of the state request function is a subscription service. When a state request message is received from an external machine, for evaluation of an IL or for

synchronizing actuators, the subscription service is invoked. After the answer to the state request is sent the component sends a new message every time its state changes, as long as the service is not cancelled.

A state check is sent when the first row of an EOP is executed and when automatic mode is entered after a stop or after a manual interaction. The state check message contains a desired state that is compared with the actual state of the sub component. If they differ, the desired state is copied to the sub component which is to generate an alarm. Since the state of an actuator object may depend on several sensors it is necessary that the alarm is generated within the actuator component.

The last type of message, order, also contains a desired state that is copied to the variable *Ordered_State*. Up to this point the desired state holds no more information than the characters in ASCII format that it is represented by. Inside the actuator component there must then be some logic that compares *Ordered_State* with the possible states of the actuator and activate the corresponding outputs. When the actuator component has reached the new state an acknowledgement is sent back to the Machine Controller.

The description of the Actuator object above is also valid for Variable objects, but not for Sensor objects. The difference between an actuator object and a sensor object is that the sensor object does not have the order function.

D. Error Detection and Alarm Generation

Alarm generation is an important property of a control program, both for maintenance and for device protection. The following three types of alarms will detect the errors in all controlled actions.

The first type is used when a machine is to execute an order. Since the state of all actuators and sensors of a machine are stored in the EOPs it will be possible to detect any unexpected state when an operation is to be started. The expected state of an actuator or sensor is sent from the Machine Controller as a state check message. If the expected state does not match the actual state the corresponding actuator or sensor object will generate an alarm.

The second kind of alarm concerns the execution of an operation at the actuator level, and is activated if the actuator fails to carry out the order. An example of how to check that an operation is fulfilled is to measure the time it takes to open a clamp.

The third kind of alarm is used to check that an actuator remains in its position. The state monitoring means that an actuator is to remain in its state until it gets a new order in auto mode; otherwise an alarm is set off. An example of a situation when this is necessary is when a robot moves close to a fixture. During normal operation, there is no risk that a fixture can enter an incorrect position and thereby cause a collision. However, the fixture could come into an incorrect position due to a mechanical fault or a manual interaction. The fact that the robot only checks that the fixture is in the right position before it starts the movement, due to the message communication, makes it then necessary to stop the cell when an actuator loses its position. A sensor object needs a separate control of the state monitoring function. The Machine Controller turns on the state monitoring via a message in the initial row of an operation, and turns it off when the state of the sensor changes in the following row. The state monitoring will also be turned off when the operation is finished.

All three types of alarm are implemented at the sub component level. A detailed description of how the method proposed in this paper improves operator support in error situations is found in [17].

E. The Message Communication between the Objects

Ten different types of messages realize the message communication between the objects,

shown in Figure 13. A message consists of the following four parts. The first part is an address, the identity of the object to receive the message. The type of the message is stated in the second part. In the third part, the identity of the sender is declared and in the fourth part, the actual information is stored.

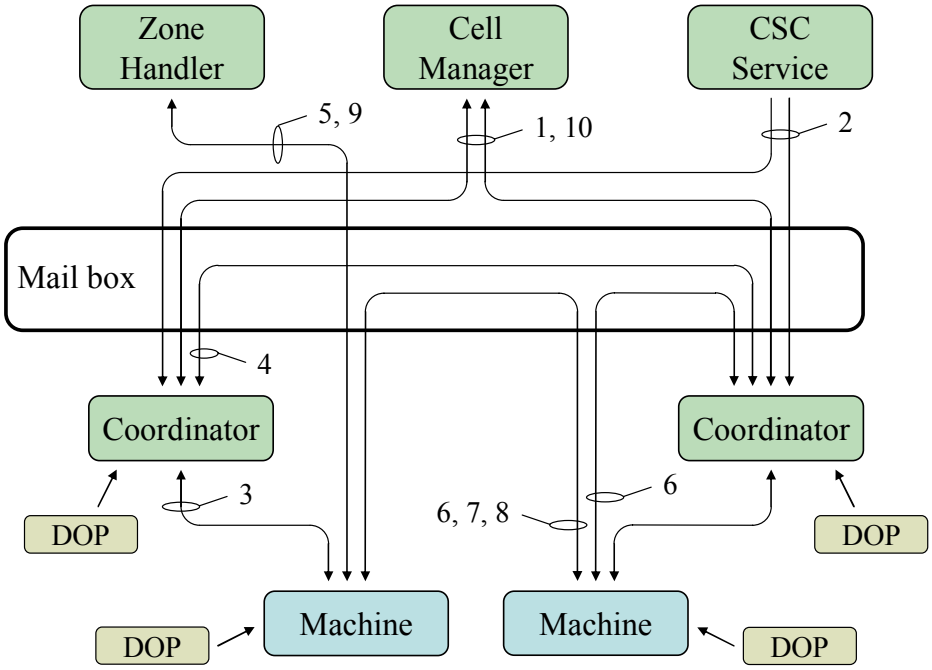

Figure 13 The message communication between the objects at the cell level.

The message exchange begins when the cell is started in automatic mode. First, the Cell manager checks that all machines are in their home position. This is done by sending a message of type 1 to all Coordinators. The response of the Coordinators is to order their respective Machine object to execute the EOP representing the home position, the first operation of each COP. When the EOP of a machine is executed its Coordinator sends a message of type 1 back to the Cell Manager acknowledging its machine in home position. When the Cell Manager has got acknowledgement from all Coordinators the state of the cell is the same as the initial state of the supervisor, in Section V.E, and the cell can be started. If the cell is started without being in the home position, the result of the verification is not valid.

When the home position of the cell is verified, the CSC Service sends a message of type 2 containing the CSC for the product to be processed to all Coordinators. Receiving the CSC is the start signal for the Coordinators that respectively loads the COP connected to the CSC and sends the identity of the first operation of the COP to their Machine object as a message of type 3. Messages between a Coordinator and its Machine object are direct while the communication with other coordinators goes via the Mailbox. The Machine object executes the operation as described in Section VI.B and sends an acknowledgement back to the Coordinator when the operation is completed, as a message of type 3.

During the execution of the COPs messages of type 4, 5, 6, and 7 can be sent. Some of the operations in a COP have predecessors executed by another machines and in those cases massages of type 4 are used to detect if or when the external predecessor is finished. A Coordinator processing an operation with an external predecessor sends a message of type 4 containing the identity of the operation and the receiving Coordinator acknowledge by a message of type 4 when the operation is completed. Since a COP only holds the operations and their order additional, information about where to send the request is needed. This information is obtained from the part of the DOP holding the identities of each operation and

the machine the operation employ, stored in each Coordinator.

The zones can be booked in two different ways depending on the operation mode. When a machine books a zone in automatic mode, it sends a message of type 5 to the zone containing the booking request. The zone sends an acknowledgement back to the machine when the zone is booked. Normally the response of a zone-booking request is sent immediately, since the COP is calculated in a way that a zone is free before it is booked. If, however, a requested zone is occupied, an answer will not be sent until the zone is free and booked by the machine in question. In manual mode the request of booking a zone is done in the same way, but as a message of type 9. The response of the zone is different. In manual mode, as there are three possible answers of a zone booking request. The zone always sends the answer immediately and it can have the following meanings. The zone can be booked, previously booked, or occupied. The answer previously booked means that the requesting machine has already booked the zone.

When operations with alternative behavior are to be executed, the selection of which alternative to choose may involve states of external components and in those cases a message of type 6 is used. The machine handling the alternative sends a message to the machine holding the component which state is required. As a response the component sends back its current state. A message of type 6 is also used when a manually controlled action is to be started and the IL of the action contains operation states. The Manual control object of the machine to execute the action sends a request to the Coordinator of the machine connected to the operation stated in the IL and the Coordinator returns the current state of the operation.

During the execution of a parallel operation the objects use messages of type 7. The information exchanged between the actuators of the involved machines is states of a set of variables needed to synchronize the operation. A message of type 6 also communicates states of components but there are two major differences between a message of type 6 and 7. The first one is the way a component sends its answer. In type 7 the component sends its current state and then it continuously sends all state changes until the parallel operation is finished. The second difference is that the concerned variables in type 7 are not defined in the EOP, but in the control programs of the actuators to be synchronized.

When an action is to be started in manual mode, a message of type 8 detects the states of the external components of the IL. This is done in the same way as when a message of type 7 is communicated. Apart from type 7 the components used in type 8 are defined in the IL.

The Coordinators send the last type of message, type 10, to the Cell Manager when they have completed their COPs.

VII. The Correctness of the Proposed Method

The line of argument showing the correctness of the proposed method is divided into four steps and held on a principal level. Details about the algorithms for the conversion of the control information to and from state machines are found in [10].

The first step argues that a non-blocking supervisor synthesized from only the models of the operations, machines and zones will guarantee that the system will be able to finish all operations without colliding or blocking. The second step claims that adding the Machine Relations, Machine Sequences, and other requirements defined in the ROP to the supervisor synthesis will guarantee that the system will not violate the order of the operations defined in the ROP, while still guaranteeing the previously mentioned properties of the system. The third step argues that including the Safety specification in the supervisor synthesis guarantees that the system will not violate the IL, while still guaranteeing the previously mentioned properties of the system. The fourth step, finally, shows that the properties of a synthesized supervisor,

as described above, are valid also for the resulting control program.

Note that, though the degenerate "null" supervisor is maximally permissive with respect to an overly restrictive specification, it is of no practical use, simply an indication that the specification is too limiting. Thus, we will assume that the synthesis does not result in the null supervisor.

A. Synthesis of DOP Related Information

The first part describes the synthesis of the three sets of state machines generated from the DOP, that represent Operations, Machines, and Zones, as shown in Figure 6. As described in Section V.B, the synchronization of the three sets of state machines results in a state machine describing all states where only one machine at a time occupies a zone. Executing the operations in one of the many orders described by the synchronization, will guarantee that the machines in the cell will not collide with each other.

However, it is not guaranteed that the system will always be able to execute all defined operations. Avoiding collisions may result in blocking.

The state machine is further processed according to the Supervisory Control Theory, [1], into a new state machine, the supervisor. Besides being collision free the supervisor is also non-blocking, which means that there exists a path of events leading from the initial state of the supervisor to a marked state. Looking at Figure 7 and assuming that the *complete* state of an operation is marked and that the *free* state of the machines and zones are marked a corresponding supervisor will have a number of alternative paths of events leading from the initial state to the marked state, where all operations are completed and all machines and zones are free. The initial state of a supervisor calculated from state machines representing Operations, Machines, and Zones, Figure 6, is a state where all operations are in their *init* state and all machines and zones are *free*. In Figure 7 the initial states of the individual state machines are marked with a bold arrow. Due to the fact that the supervisor has alternative paths and due to the structure of the COP, it is not possible to extract a set of COPs from such a supervisor, see Section V.E. However, assuming that an optimization algorithm is applied, selecting the shortest path with respect to the time of the execution of the operations a set of COPs can be extracted.

B. Synthesis of ROP Related Information

In the second part of the analysis of the correctness of the proposed method, the state machines called Machine Relation and Machine Sequence, Figure 6, are included in the synthesis. A Machine Sequence has one initial state representing the home position of the corresponding machine. For each alternative operation a branch is inserted in the state machine, Figure 8 top. At the end the different branches converge to the same final marked state. Operations with alternative execution are converted into Alternative operations. Since a state in the supervisor is marked only if the corresponding states in all involved state machines are marked the marking of the state machine representing the operation in Figure 7, has to be modified. In order to make it possible to mark states in the supervisor that represent the marked states of the alternative branches both the *init* and the *complete* state of operations of the type Alternative are marked, Figure 7 left. Since the *executing* state is not marked, the supervisor guarantees that an operation, once started, will always be able to finish; otherwise it will not be allowed to start. Since the start and end events of all operations are included in the Machine Sequence the property of the supervisor guaranteeing that all operations are completed is maintained, with the specialization that all *required* operations are completed. In addition, the supervisor also guarantees that the required order of operations defined in the ROP will not be violated.

When the information stored in the ROP transforms into Machine Relations and Machine Sequences also parallel operations are processed. They do not, however, in principal affect the synthesis. In order to enable parallel operations to be executed simultaneously each set of operations parallel with each other are given the same identity, as described in Section V.C. Since the operations are to be started at the same time and the corresponding zones are to be booked at the same time the relabeling will not affect the synthesis in any other way than the synchronization of the parallel operations. A relabeled set of parallel operations can be viewed as one operation employing several machines.

Synthesizing a supervisor from the Operations, Machines, and Zones together with the Machine Relation and Machine Sequence imposes further restrictions on the order of the execution of the operations. Thus, the resulting supervisor will be more restrictive than the one previously described but it may still contain a number of alternative paths from the initial state to the marked state. Again assuming a time optimization of the paths through the supervisor, or as described in Section V.E, adding Machine Relations, will make it possible to extract the COPs. The extracted COPs holds an order of the execution of the operations that, avoids collisions between the machines, guarantees that all required operations will be executed, and that the required order of operations defined in the ROP will not be violated.

C. Synthesis of EOP and IL Related Information

In the Safety specification the IL for an action of an actuator is represented by a set of state machines, Figure 9, showing combinations of the states of a set of operations that have to be fulfilled in order to allow start of the operation where the actuator executes the action. Since the state machines in the safety specification contains only *start* and *end* events of operations, the requirements from the IL can be related to previously described parts of the supervisor synthesis. Including the Safety specification in the synthesis will result in a supervisor where the order of the execution of the operations, in addition to the above described properties, will not violate the IL.

A prerequisite of the validity of a synthesis including a Safety specification is that the information hold by the state machines in the Safety specification equals the information in the original specification of the IL. The reasoning below shows the equality between the Safety specification and the original IL.

As described in Section V.D, the detection of operations fulfilling or not fulfilling the IL is made by a search for state changes of external components in the EOPs. Due to the fact that the EOPs constitute a model of the state changes of the components in a cell during one work cycle, all relevant state changes will be detected. Since the resulting state machine allows execution of the action only when an operation putting the external component in a desired state has ended and no operation has started that puts the component in a state not fulfilling the interlock, the Safety specification equals the IL.

D. Execution of the Synthesized Information

To control the cell the COPs are loaded to the Coordinators in the control program of the cell. A Coordinator is a standardized object interpreting a COP and acting accordingly, by ordering its Machine object to execute operations in the same order as stated in the COP, and by waiting for other Machine objects to finish certain operations, as stated in the COP. At the machine level when executing an operation, the Machine object loads an EOP and orders the components of the machine to adopt the states defined in the EOP. Consequently, the machines in the cell will be ordered to start their operations in a way that is consistent with the supervisor and the execution of the operation is consistent with the IL. Thus, the described properties of the supervisor are transferred to the control program. The cell will be able to

process all its operations without collisions, and blocking, and it will violate neither the IL nor the required order of the operations. When all COPs are executed, the cell is in the same state as the marked state of the supervisor and all operations have been executed, consequently. A prerequisite for the validity of the transfer is that the initial state of the cell matches the initial state of the corresponding supervisor. In the control program the Cell Manager, as described in Section VI.E secures this.

VIII. Conclusions and Future Work

The method proposed in this thesis provides results that solve the industrial problems mentioned in Section I, without adding work to the development process. On the contrary, the amount of work will be reduced since a part of the PLC program development will be automated and the time for debugging the PLC program on the shop floor will be drastically reduced, due to the new mathematically based verification process.

The most important result is the modeling of the control of a cell. It gives a possibility for standardized representation of the control information, which enables information reuse and simplifies the automation of the verification and program generation processes. Since the modeling of the control separates information about coordination and execution of operations the complexity of the information needed for verification of the coordination is essentially reduced. This will significantly reduce the number of states in the verification process and thereby also reduce the effects of the combinatorial state-space explosion problem. The modeling of the control also impacts the handling of error situations in a cell. Since the interlocks are separated from the program code and represented by data, it is possible to design standardized software components that interpret the data and give improved operator support and resynchronization functionality.

The planned future work involves a continuation of the verification of the industrial ability of the method, development of algorithms for analysis of the result of the supervisor synthesis, an extension of the three-level control hierarchy to a multi-level control hierarchy, and development of algorithms for time optimization of the order of the execution of the operations.

The verification of the industrial ability of the method is to be done by a study of manufacturing cells from assembly plants within the automotive industry. The study will include mapping of control properties of a number of cells from different processes. In the analysis of the study, the control of the cells will be modeled according to the method.

In case the supervisor synthesis does not find a solution that fulfills the specification the result gives no indication of why the synthesis failed. Therefore it is important to develop a feedback mechanism identifying parts of the specification that might have caused the unsuccessful outcome.

The purpose of the extension of the three-level control hierarchy is to extend the scope of the proposed method from the control of the work inside a cell, as presented in this paper, to the control of the material flow between cells.

REFERENCES

[1] Ramadge, P. J. Wonham, W. M., "The control of discrete-event systems", Proc. Of the IEEE, vol. 77, no. 1, 1989.
[2] Chandra, V. Zhongdong Huang, Kumar, R. "Automated control synthesis for an assembly line using discrete event system control theory" Systems, Man and Cybernetics, Part C, IEEE Transactions on Volume 33, Issue 2, May 2003.
[3] Lauzon, S.C.; Ma, A.K.L.; Mills, J.K.; Benhabib, B.; "Application of discrete-event-system theory to flexible manufacturing", Control Systems Magazine, IEEE Volume 16, Issue 1, Feb. 1996.
[4] De Smet, O. Rossi, O."Verification of a controller for a flexible manufacturing line written in Ladder Diagram via model-checking", Proc. of the Americanagram via model-checking,l. in Proc. of the American Control Conference, Anchorage, 2002.
[5] Rausch, M.; Krogh "Formal verification of PLC programs" , B.H.; American Control Conference, 1998. Proceedings of the 1998 Volume 1, June 1998.
[6] Richardsson, J. Fabian, M. "Modeling the Control of a Flexible Manufacturing Cell for Automatic Verification and Control Program Generation", Journal of Flexible Service and Manufacturing, Volume 18, Number 3, September 2006.
[7] Richardsson, J. Andersson, K. Fabian, M. "Reliable Control of Complex Manufacturing Cells" Proceedings of the 2007 IEEE International Symposium on Assembly and Manufacturing, Ann Arbor, USA • July 2007.
[8] Flordal, H. Malik, R. "Supervision Equivalence", Proceedings of the 4th Workshop on Discrete Event Systems, WODES'06 Ann Arbor, MI, USA , 2006.
[9] Feng, L. Wonham,W.M. "Computationally Efficient Supervisor Design: Abstraction and Modularity", Proceedings of the 4th Workshop on Discrete Event Systems, WODES'06 Ann Arbor, MI, USA , 2006.
[10] Danielsson, K. Richardsson, J. Lennartson, B. and Fabian, M. "Automatic Scheduling and Verification of the Control Function of Flexible Assembly Cells in an Information Reuse Environment ", 6th IEEE International Symposium on Assembly and Task Planning, Montreal, Canada, July 2005
[11] Åkesson, K. Fabian, M. Flordal, H. Malik, R. "Supremica – An integrated environment for verification, synthesis and simulation of discrete event systems", Proceedings of the 4th Workshop on Discrete Event Systems, WODES'06 Ann Arbor, MI, USA , 2006.
[12] Sharique, F. Gore, G. Richardsson, J. Fabian, M. "Verification of a Novel Approach in Control of Flexible Manufacturing Cells", Proceedings of the Intl. Conference on Flexible Automation and Intelligent Manufacturing, Toronto, Canada, July 2004
[13] Ljungkrantz, O. Åkesson, K. Richardsson, J. Andersson, K. "Implementing a Control System Framework for Automatic Generation of Manufacturing Cell Controllers" Proceedings of the 2007 IEEE Intl. Conference on Robotics and Automation, Rome, Italy, April 2007.
[14] Shlaer S. Mellor S. "Object Lifecycles", Yourdon Press, 1992.
[15] Kollura, R., Smith, S., Meredith, P., Loganautharaj, R.; Chambers, T.; Seetharamau, G.; D'Souza, T., "A framework for the development of agile manufacturing enterprises" Proceedings of the IEEE Int. Conference on Robotics and Automation, 2000.
[16] Laplante, Phillip A., "Real-time systems design and analysis", page 173, IEEE Press, cop. 1997.
[17] Richardsson, J. Danielsson, K. Fabian, M. "Design of Control Programs for Efficient Handling of Errors in Flexible Manufacturing Cells" Proceedings of the 2004 IEEE Intl. Conference on Robotics and Automation, New Orleans, USA, April 2004.

Paper 2

Modeling the Control of a Flexible Manufacturing Cell for Automatic Verification and Control Program Generation

Johan Richardsson and Martin Fabian

Journal of Flexible Service and Manufacturing, Volume 18, Number 3, September 2006

(this journal was previously named
International Journal of Flexible Manufacturing Systems)

(The paper is reformatted for readability)

Modeling the Control of a Flexible Manufacturing Cell for Automatic Verification and Control Program Generation

Johan Richardsson[1] · Martin Fabian

Abstract – Shortened product life-cycles decrease the output rate of manufacturing systems. Offline verification of the control systems promises to increase the output. However, to make offline verification possible some major improvements of the current development process of manufacturing systems are needed. Information handling and development of control programs based on information reuse are two of the most important improvement areas. This paper presents the results of the modeling of a real manufacturing cell according to a previously presented method for offline verification and program generation based on information reuse.

1 Introduction

Consumer product life-cycles are constantly shortening; the automotive industry is an illustrative example. As a consequence, the introduction of new products into the manufacturing system necessarily becomes more frequent. Inherently, this brings a performance reduction for the manufacturing system. The reduced performance is caused by a down-time and a ramp-up-time. During the down-time the mechanical equipment is rebuilt and the new control programs are debugged. During ramp-up there are a large number of errors mainly caused by mechanical devices not being properly adjusted, bugs in the control programs and operators not used to new procedures. Thus, in order to maintain the productivity level and to achieve full cost-efficiency both the down-time and the ramp-up time must be reduced. One way to reduce these lead times is to verify the control programs in offline mode. However, efficient and reliable offline verification requires some major improvements of the current development process of manufacturing systems. Information handling and development of control programs based on information reuse are the two most important improvement areas.

The need for information reuse is illustrated with an example from the automotive industry, the development of control programs for assembly cells. First a mechanical engineer

Johan Richardsson
Volvo Car Corporation, Sweden
jrich103@volvocars.com

Martin Fabian
Chalmers University of Technology, Sweden
fabian@chalmers.se

designs the cell and its control function. Then the same control function is implemented, typically manually, in a robot simulation tool where robot paths are added. After that the mechanical design is verified by a 3D-simulation. Finally the same control function is implemented a third time, in the PLC (programmable logic controller) program. One reason for the multiple implementation of the control function is that contemporary manufacturing systems development largely consists of isolated sub-processes that generate proprietary information; CAD drawings, different models for simulation, mechanical timing diagrams, electrical schemas etc. Efficient information handling is essential in improving this situation.

In a previously presented method, (Richardsson and Fabian 2003a, 2003b), and (Danielsson et al. 2005), information from different tools in the development process of a manufacturing system is reused and processed by tools for verification and optimization. Then the PLC (Programmable Logic Controller) programs are generated by combining the processed information with a library of standardized software components. This paper shows how the control of an existing cell from Volvo's Torslanda plant can be modeled according to the previously presented method. Thus we present a concretization of the rather abstract description given in (Richardsson and Fabian 2003a, 2003b) and (Danielsson et al. 2005).

Figure 1 shows an outline of the previously presented method. Different tools involved in the development of a manufacturing system exchanges information in a standardized format, (von Euler-Chelpin et al. 2004), through a data base, the Manufacturing system data. A certain part of this information represents the control function of the system, the Control Information. Since the information about the control of a manufacturing cell is stored according to a standardized data format, instead of today's practice of being embedded in a control program, it will be possible for new development tools to access that information. This paper concerns new tools for verification and optimization of the control function and generation of the PLC program.

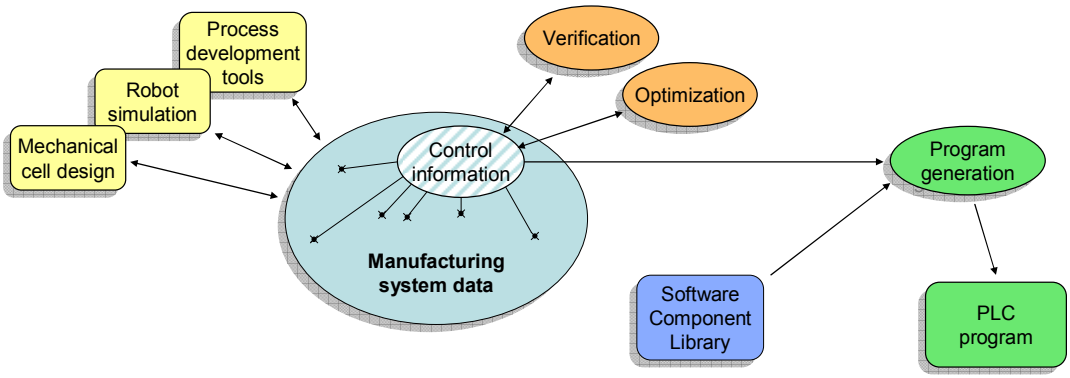

Figure 1 Overview of the method for automatic verification, optimization, and generation of PLC programs.

2 Related Work

Since the previously presented method covers several, sometimes overlapping, research areas this section is divided into three parts, one for each area.

2.1 Verification of Control Programs

Verification of control programs is a well established research area but industrial applications are still very few, especially for flexible manufacturing systems targeted by this work. This section briefly describes three methods for development of correct control programs and connected reasons for the low usage of the methods in control of manufacturing cells.

The first method is the *supervisory control theory* (SCT), (Ramadge and Wonham 1989). A model of the system and a specification of the behavior are processed together. The result is a logical model, typically a finite state machine, of the control of the cell that by construction fulfills the specified behavior. There are three assumed reasons for the low usage of SCT in industry. The first one is a theoretical problem; the combinatorial state-space explosion of the logical model when the control of a complete cell is processed together, generally results in unacceptable calculation times or even makes the calculations intractable. The second problem is the fact that the output is a logical model that has to be converted into program code. Several applications, for instance (Chandra et al. 2003) and (Lauzon et al. 1999) have shown that it is possible to automatically convert the output to program code but those applications concerned small systems and it is not obvious that it scales to systems of industrial sizes. A connected problem is that the applications usually concern only the automatic operation of a system which is about 10% of the control program. An important question is then if it is possible, in an efficient way, to represent the functions of a complete program by state machines. The third reason has to do with a need of increased resources for development of a control program. In addition to the work needed in traditional development a detailed logical model of the cell has to be made.

In the second method, *formal verification* ((Smet and Rossi 2002), (Rausch and Krogh 1998)), an existing control program is converted into a format that is interpretable by a verification tool. A model of the system to be controlled is developed and the verification tool will confirm or reject statements, typically expressed as temporal logic formulas. The reasons for the low usage of Formal Verification are similar to the ones of SCT. The two methods share the state-space explosion problem and formal verification requires even more additional work since the properties that are to be verified have to be stated individually. However, in contrast to development of one of a kind manufacturing cells the additional work for using formal verification is financially motivated in development of high volume products with complex control functions.

The third method is *simulation*, which in this context means that a control program is verified by having it controlling a 3D model of a cell, while a designer is watching that everything works as planned. Like in SCT and Formal Verification additional work for building a model is needed, but for the simulation of a cell the 3D model can typically be reused from the robot simulation.

A problem with simulation of manufacturing cells is that the efficiency depends on the type of control that is used in the cell, whether it is deterministic or not. In a deterministically controlled cell, when the operations of the machines are always performed in exactly the same way, simulation is a very good verification method. However, most cells are controlled in a nondeterministic way, which may result in a very large number of situations that have to be simulated individually for guaranteeing the desired behavior of a program. The main reason for having nondeterministic control is to minimize the cycle time of a cell. A deterministic control affects the cycle time negatively in two different ways. Firstly, in the case of deterministic control the programmer has to minimize the cycle time by manually scheduling the order of all operations. This is a difficult task considering the fact that a work cycle usually includes more than 50 operations with typically four to ten operations executed in parallel. In a nondeterministically controlled cell the machines operate individually and are synchronized only at specific instants, for instance to avoid collisions or to pass material between machines. The latter does not guarantee a lower cycle time but letting the machines execute as many operations as possible between the synchronization points is expected to result in a lower cycle time on average. Secondly, if the control of a cell is to be considered as deterministic, so it is possible to verify a control program by simulation of a single work cycle, it is necessary that every state change of the machines in a cell is synchronized.

Otherwise, every time a machine is delayed, the cell will be in a new state not included in the verification. A series of state changes of a machine describe an operation and since the duration of operations is different, the deterministic control will cause the machines to wait for each other at the synchronizations and may thereby increase the cycle time.

2.2 Development of Control Programs

Several papers have been written about development of control programs for automated manufacturing. These can be roughly divided into two types. The first type (Kanai et al. 2000), (Chan et al. 2000), (Kollura et al. 2000), could be classified as based on software engineering and the second type (Lauzon et al. 1999), (Park et al. 1999), as based on mathematical system representation and manipulation. The program structure proposed in this paper is based on software engineering principles and adapted to import information processed by formal mathematical methods.

There are four reasons why the previous research does not solve the problems connected to shortening the down- and ramp-up times. First, in (Kanai et al. 2000), (Chan et al. 2000), (Lauzon et al. 1999) and (Park et al. 1999) only the part of the program that involves the control function for automatic operation is concerned. This is typically 10% of a complete control program and the whole program must be considered in order to get a reliable verification and an efficient program development process. Secondly, information reuse is not considered. The logic is manually specified in a programming tool, (Kanai et al. 2000), and (Chan et al. 2000), by controlled-automata, (Lauzon et al. 1999), or by Petri nets, (Park et al. 1999). Thirdly, since the logic is manually specified it is not possible to guarantee that the specification is uncorrupted after conversion to a control program. Fourthly, (Kanai et al. 2000), (Chan et al. 2000), and (Lauzon et al. 1999) specifically deal with automated manufacturing cells, but they do not cover the implementation of the control programs of the individual devices. The reason for this is clear: the devices involved are mills, lathes and similar, and they all have their own control systems. In the processes considered in this paper, this is not the case.

2.3 Information Reuse

Information reuse or exchange of information between different tools based on standards is another well established research area but little previous research results have been found concerning information reuse for development of control programs. In those cases when research results about information reuse in development of control programs have been found the scope have been reusable software components, (Kovács et al. 1999), not the information that is to be converted into a control program, which is the meaning of information reuse in this paper. The academic research about development of control programs has so far not involved information reuse. Industry, on the other hand, has noticed the improvement potential. In recent years automation suppliers have launched new tools for more efficient development of control programs. The basic idea of these tools is to reuse information from the mechanical design of a cell and to convert that information to program code for a control system. In one survey, (Richardsson 2002), three of these tools were evaluated. Several general limitations were identified, of which two are important for the current discussion. Firstly, it is not possible to express the complete control function of a cell in the mechanical part of the development tool. Then manual work is needed to convert the specification to program-code, which leads to three disadvantages. The information reuse becomes a one time event since the manual work must be redone every time the mechanical information is changed. Also, when manual work is included in the conversion, it is no longer possible to guarantee an uncorrupted conversion of the specification to program code. Finally, the

verification of the operation of a cell is mainly by physically observing a simulation, a tedious and hence error-prone task.

3 The Proposed Method

An important aspect of the proposed method is to decompose the information about the control of a cell. Decomposition of the control information means that a complete schedule of a cell, a timing diagram from the mechanical design or a SOP, (sequence of operations), from the robot simulation is broken down into a set of operations, each accompanied with scheduling information. The decomposition gives the possibility to schedule the operations in a time optimized way and in a way that collisions between the resources are avoided and other requirements of the behaviour of the cell will be fulfilled. The information needed to describe the control of a cell is divided into six categories,

1. Relations of Operations, ROP
2. Execution of Operations, EOP
3. Sequence of Operations, SOP
4. Interlocks, IL
5. Mechanical Components and Functions, MCF
6. Function Blocks, FB

The ROP, *Relations of Operations,* holds information about when an operation can be executed and the EOP, *Execution of Operations,* holds information about the states a resource passes through while an operation is executed. Information about how a resource is to be controlled to go between different states is stored in FB, *Function Blocks*, as program code. IL, *Interlocks*, shows the conditions, in terms of states of the components in the cell that have to be fulfilled before a resource can execute an operation without causing any damage to the cell. ROP, EOP, and IL are used as input to the verification and optimization tools and the output from them is the SOP, *Sequence of Operations*. Details about the verification and optimization are found in (Danielsson et al. 2005). The SOP shows a time optimized order of the execution of the operations that guarantees that the IL will not be violated and that the ROP will be fulfilled. By the design of the verification algorithm the SOP will also be deadlock free. Items 1 to 5 are stored as data in *Control Information*, in Figure 1, while item 6 is stored as program code in the *Software Component Library*, also shown in Figure 1.

MCF, *Mechanical Components and Functions*, is a list of the mechanical components in a cell and a list of general functions, not directly connected to the mechanical equipment, needed to control a cell. The MCF is an input to the program generation process and it points out which FB's that are to be instantiated in the PLC program of a specific cell.

The main sources of information for generation of PLC programs ar the mechanical design of the cell and the robot simulation. From the mechanical design of a cell come MCF, ROP, EOP, and IL. From the robot simulation we also get scheduling information and control logic. The control logic is the same as the one from the mechanical design, as is the scheduling information except for one additional part, the collision *zones*, a part of the ROP. von Euler-Chelpin et al. (2004), presents an information model where the control information is represented according the ISO 10303 international standard.

The basic idea of the program generation is to "assemble" the control program out of objects representing the machines in a cell and basic cell functions. These objects are instantiated from a library of standardized software components, FB's. The SOP, that constitutes the "glue" that coordinates the objects, can then be loaded as data to the PLC and interpreted by a standardized software component, a general sequence function called Coordinator. In the same way the EOP's are loaded to FB's representing the machines in a cell. Those FB's are also equipped with a standardized software component that interprets an

EOP and control a machine accordingly. The execution of an EOP could be compared with the movement control part of the application program of an industrial robot. It contains a series of locations that is interpreted by the robot control system, which orders the robot to move along the locations. Details about the program generation are found in (Richardsson and Fabian 2003a), (Sharique et al. 2004) presents an implementation of the proposed method.

In the first step of the PLC program generation process a standardized framework is loaded in the program development tool. The framework is a set of predefined variables for communication between the FB's that are to be instantiated. In the second step the FB's are instantiated according to the MCF, the list of Mechanical Components and Functions of a cell. In the third and final step the SOP, EOP, and IL are loaded to the FB's.

The principle structure of the control of a cell is outlined in Figure 3. When a new product has entered the cell it is detected by the Product Data Handler which forwards information about the type of product to the Coordinator. The Coordinator loads the SOP for the current type of product and starts to distribute the operations to the resources according to the SOP. When a resource receives an order it loads the corresponding EOP and executes the operation accordingly. When finished the resource signals back to the Coordinator that the operation is finished and the Coordinator can move on in the SOP. This is repeated until all operations in the SOP are executed and the work cycle of the cell is finished.

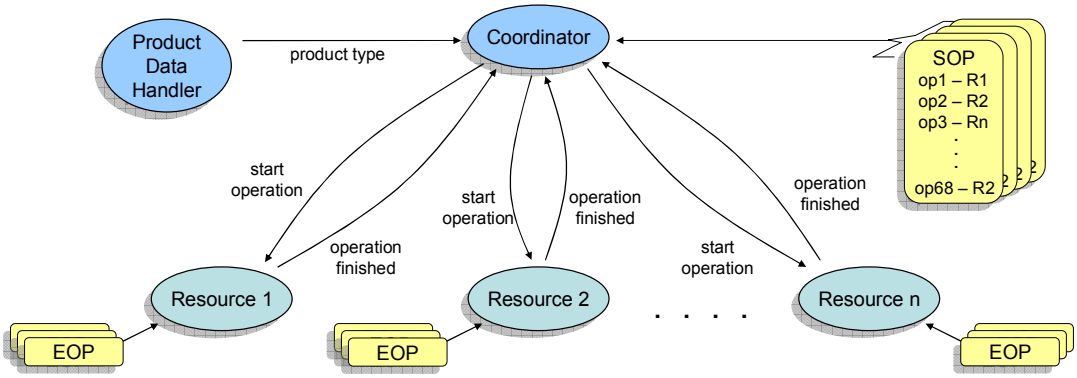

Figure 2 The principle structure of the control of a cell

4 The Sample Cell

The sample cell, Figure 2, has nine resources, four robots, two fixtures, two turn tables, and a conveyor. The task of the sample cell is to spot weld a plate to the side of the floor of a car. Additional spot welding on previously loaded parts is also done. Two products are processed in the cell, Volvo V70, and S80.

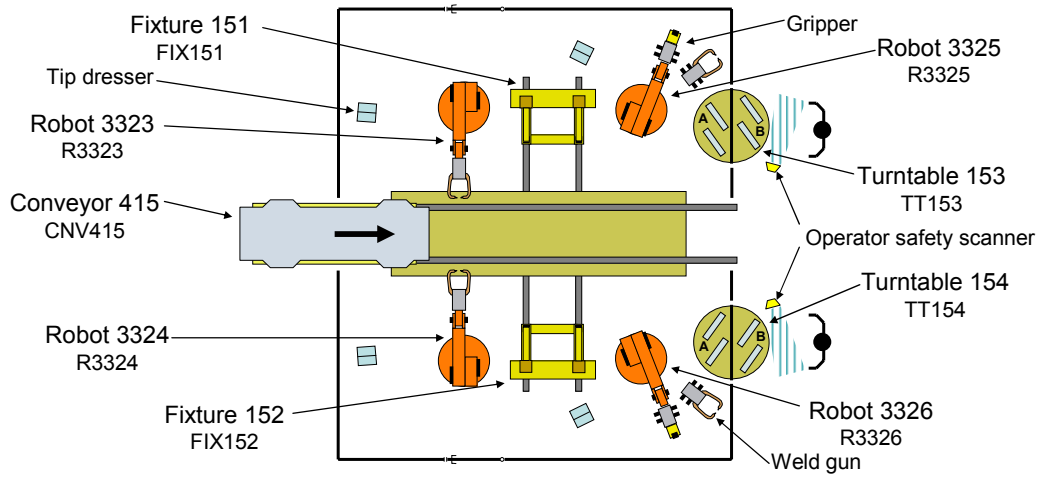

Figure 3 The mechanical components of the sample cell.

The work in the cell starts when the cell is informed of the type of body that is next going to enter the cell. Robot 3325 and 3326 pick parts from the turntables and put the parts in their respective fixture. The fixtures clamp the parts and the robots move to a tool change position where the gripper is exchanged for a weld gun. When the conveyor is in work position the fixtures move in and the robots start to weld. All four robots have three different weld jobs. In the first job the robots weld previously loaded parts. In the second job the robots weld the parts in the fixtures and the floor together. The third job starts when fixtures have left the floor and reach their home positions. The robots then weld parts of the plate previously blocked by the fixtures. After the third job the robots move out of the floor, the cell has finished its work and enables the conveyor to move the floor out of the station.

5 Modeling the Control of the Sample Cell

The aim of our modeling is to support information reuse and offline verification. A clear separation between the information concerning operation coordination, the SOPs, and the information concerning operation execution, the EOPs, together with a standardized representation of the information enables this. The information comes from the robot simulation and the mechanical design of the cell.

5.1 Defining Operations

An operation is a small part of the work cycle of a cell containing one or a series of actions of an actuator or a group of actuators. An actuator may belong to one group only and it is not possible to regroup actuators. To be able to handle handshaking between resources in the strictly hierarchical control structure, see Figure 2, some operations do not result in actual actions. An example from the sample cell of such an operation is when a robot is to put a part in a fixture. Before the robot can put the part in the fixture it must be sure that the fixture is in the right position for receiving the part. In the method presented here, it is not the robot that does this check, but the Coordinator. This check is represented by an operation only checking the expected state of the fixture. When the fixture receives the operation it checks its state and

if it is the same as the expected state the fixture acknowledges the operation. Then the Coordinator orders the robot to put the part in the fixture. The situation when Robot 3326 is to put a part in a Fixture 152 is modeled by two operations and the communication between the Coordinator and the resources is shown in Table 1.

Direction	Operation	Comment
From Coordinator to Fixture	Ready to receive part?	The Coordinator checks if the fixture is ready
From Fixture to Coordinator	Ready to receive part!	The fixture indicates ready
From Coordinator to Robot	Put part in fixture	When the Coordinator is notified that the fixture is ready it orders the robot to put the part in the fixture
From Robot to Coordinator	Put part in fixture	The robot indicates ready

Table 1 Example of communication between the Coordinator and two resources.

5.2 Classification of Operations

For scheduling and control purposes the operations have to be classified into one out of five different types. The five types of operations are,

1. Basic
2. Alternative
3. Parallel
4. Uncontrolled
5. Unscheduled

Type 1, a basic operation, is a straight forward operation executed by a single resource in the same way every time. Type 2, the alternative operation, is also executed by a single resource but the way it is executed may vary from time to time. There is one EOP defined for each way an alternative operation can be executed in. The third type, the parallel operation, is carried out simultaneously by two or more resources. An example of the fourth type is an operator putting a part in a fixture. The purpose of including uncontrolled operations in the specification of the control is to make it possible to optimize the whole work cycle of the cell and to be able to detect unexpected delays. An example of the fifth type is an operation trigged by an error and therefore not scheduled in a normal work cycle.

5.3 Defining the Relations of Operations, ROP

The ROP holds information for the verification process and for the optimization process. The information concerns the interaction between resources and the duration and frequency of the operations. In a ROP each operation is connected to the following nine types of information,

1. Type of operation.
2. Resource, the resource that is to execute the operation.
3. Predecessors, requirement of operations that have to be executed before the operation can be executed.
4. Product, the type of product the operation is to be executed on.
5. Work cycle, in case the work cycle of a cell is divided into several cycles this information indicates in which work cycle the operation is to be executed.
6. Zones, a zone represents a physical volume in the cell and to prevent collisions only one resource at a time may be in a specific zone.
7. Duration of the operation.
8. Frequency of the operation, the operation is executed every n-th work cycle.
9. Initial state, if the execution of the operation is dependent on a specific initial state of the cell, the identity of that state is entered here.

Table 2 shows the ROP for all operations of Fixture 152 and the ROP for some of the operations of Robot 3326. Duration, frequency, and initial state are not shown in the table due to space limitations.

id	Comment	type	resource	prede-cessors	product	work cycle	zones before	zones after
40	Cycle start condition	1	FIX152	-	all	1	-	-
41	Ready to receive part?	1	FIX152	-	all	2	-	-
42	Close clamps	1	FIX152	93	all	2	-	-
43	Move fixture to work position	1	FIX152	1	all	2	2,4,6,8,10	2,4,10
44	Move fixture to home position	1	FIX152	63,98	S80	2	2,4,10	-
45	Move fixture to home position	1	FIX152	64,99	V70	2	2,4,10	-
91	Change tool to gripper	1	R3326	-	all	2	-	-
92	Fetch part from turntable, new rack side A	2	R3326	91,121	all	2	12	-
92	Fetch part from turntable, old rack side A	2	R3326	91,121	all	2	12	-
92	Fetch part from turntable, new rack side B	2	R3326	91,121	all	2	12	-
92	Fetch part from turntable, old rack side B	2	R3326	91,121	all	2	12	-
93	Put part in fixture	1	R3326	92,41	all	2	10	-
98	weld job 2, weld part hold by fixture	1	R3326	96,43	S80	2	8	-
99	weld job 2, weld part hold by fixture	1	R3326	97,43	V70	2	8	-

Table 2. The table shows the ROP for a selection of operations from the sample cell.

As indicated in the *work cycle* column, the sample cell has two work cycles. In the first cycle the incoming product is identified and the initial state of the cell is checked. The third operation in Table 2, number 42, has operation 93 as predecessor, which means that operation 93 must be finished before operation 42 can start. In other words, Robot 3326 must put the part in Fixture 152 before the fixture can close its clamps. Operation 43, move fixture to work position, has operation 1 as predecessor; the floor must be ready for welding. To operation 43 there are also five zones connected. The zones in the *before* column means that Fixture 152 will occupy zone 2,4,6,8, and 10 during the execution of operation 43 and the zones in the *after* column means that the fixture will still occupy zone 2,4, and 8 when the operation is finished. Zone 6 and 8 will be used by robot 3324 and 3326, respectively when the part is welded to the floor. Figure 4 shows the location of the zones in the cell.

Most of the operations are executed on both products but the weld jobs of the robots are specific for each type of product. Consequently there must be two operations for moving the fixture back to home position since the predecessors will be different depending on the type of the product that is processed. The difference between operation 44 and 45 is the predecessors.

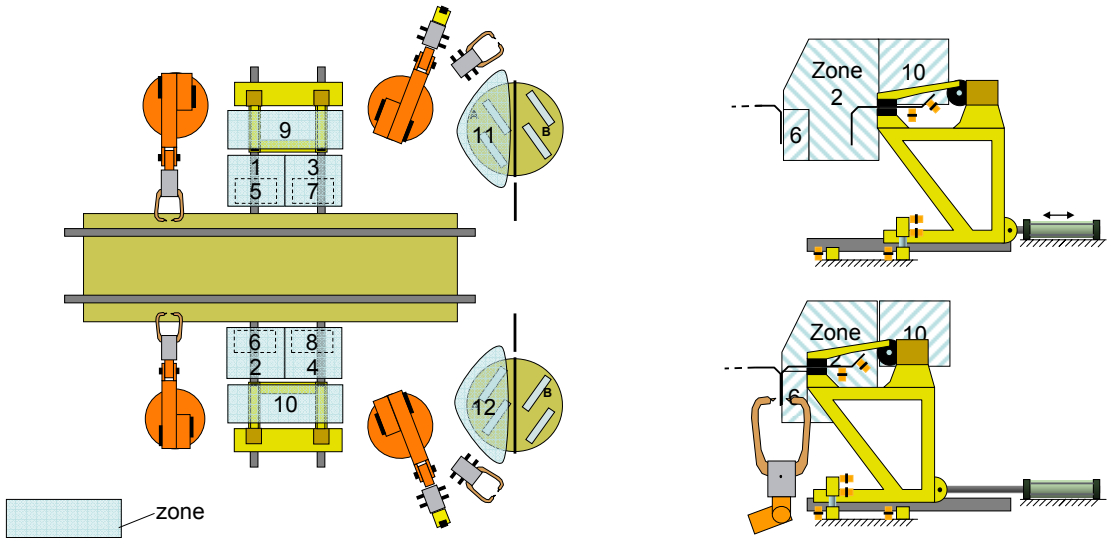

Figure 4 Zones in the cell in which two or more resources can move.

Operation 92, fetch part from turntable is of type 2, an alternative operation. This operation can be executed in four different ways. Two of the alternatives depend on which side of the turntable that is in the cell. Since the turntable is not exactly symmetric, the robot has one

program to fetch from side A of the turntable, and one for side B. The other two alternatives depend on whether the racks on the turntable are full or not. The robot counts the parts it picks and uses that information to calculate the position of the next part to pick. The robot then needs to know when the turntable has rotated and the racks are full. The alternative operation is modelled as four operations with the same number. In operation mode the robot will evaluate the initial-state-information, stored in the EOP of every operation, and then choose the operation with initial state matching the current state of the turntable.

The difference, with respect to zone allocation and duration, between the four alternatives of operation 92 is negligible and it is therefore suitable to model operation 92 as an alternative operation. If, on the other hand, the difference would have been considerable the possibilities for scheduling and optimization will be reduced since the operation will claim more zones and have an uncertain duration. In case of a big difference between alternatives it is better to model an operation as several basic operations each connected to different initial states of the cell, given that the necessary information is available before the work cycle is started. This can be viewed as a tuning parameter.

5.4 Defining the Execution of Operations, EOP

An EOP holds information about the states that a resource passes through while an operation is executed and it is used to control the resource. In order to reduce the complexity of an EOP, a resource is represented by its actuators and the sensors not connected to an actuator. For example, the states of sensors detecting if a clamp is open or closed is not included in the EOP, only the state of the clamp. The FB controlling the clamp must then be able to receive orders of which state the clamp is to be in and to give feedback about which state the clamp is in. Figure 5 shows how Fixture 152 is represented.

In order to reuse information from the mechanical design of the cell the hierarchy of the FB's controlling the fixture in Figure 5 has the same structure as the mechanical components of the fixture. Then the list of the mechanical components of the cell, the MCF, is used to point out which FB's that are to be instantiated in the PLC program. Besides the FB's representing the mechanical components of the resource, there is also a control FB. The control FB is a standardized FB that receives order from the Coordinator, loads the corresponding EOP and controls the resource by distributing states to the actuators according to the EOP.

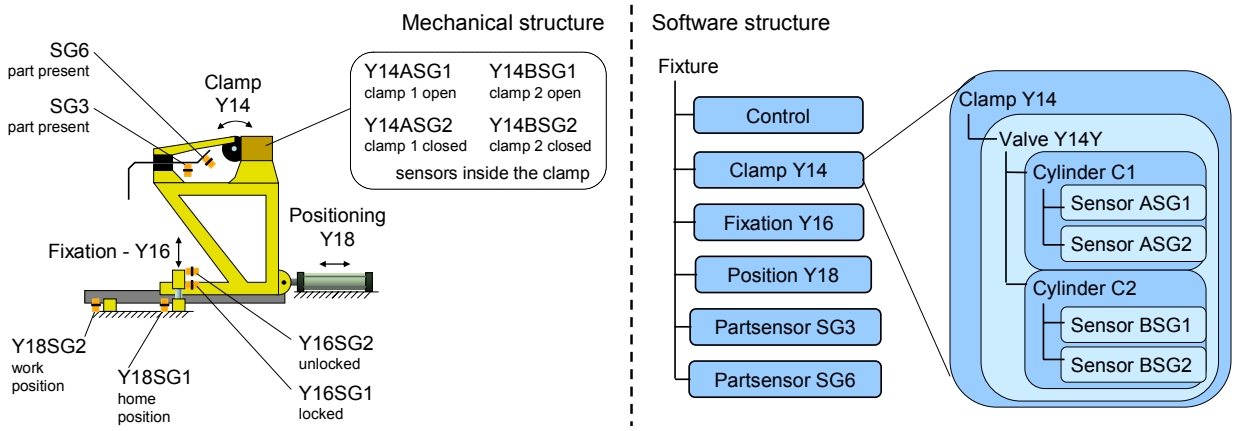

Figure 5 The mechanical structure of Fixture 152 and the corresponding software structure.

Table 3 shows the EOP for operation 43, move Fixture 152 to work position. The first row of the table is the supposed initial state of the components of the fixture when the operation is

to be executed. The initial states are compared with the actual states of the components, and if they do not match an alarm is set off and the resource is stopped. If the states match, the control FB will order the components to adopt the states of row two, Action 1. Action 1 also includes booking the zones 2,4,6,8, and 10. When the states of the components are the same as the ones in row two and the zone booking is confirmed, Action 1 is finished and the control FB will order the components to adopt the states of row three, Action 2. This procedure is repeated until all rows have been executed. Then the control FB sends a message back to the Coordinator to acknowledge that the operation is executed.

	internal components						initial state check	
	Y18	Y16	Y14	SG3	SG6	zones	alarm type	alarm delay
Initial state	home	locked	close	on	on	-	1	0
Action 1	home	unlocked	close	on	on	b(2,4,6,8,10)		
Action 2	work pos	unlocked	close	on	on	-		
Action 3	work pos	locked	close	on	on	u(6,8)		

Table 3 EOP for the basic operation 43, move Fixture 152 to work position.

Table 4 shows the EOP of the more complex, alternative operation 123, rotate Turntable 154. This operation can be carried out in three different ways, no rotation, rotation from side A to side B in the cell, and rotation from B to A. The outcome of no rotation is detected in two ways and the rotations in one way each, so there are four alternatives for operation 123. Table 4 shows two of them. The difference in duration and zone allocation between the alternatives are considerable and the operation should then be modeled as basic operations connected to different initial states of the cell, as mentioned in Section 3.3. However, in this case the turntable can rotate rather independently from the other resources of the cell and neither limits the scheduling possibilities nor the optimization.

Operation 123 Rotate Turntable 154 - no rotation, new rack

	internal components				ext.comp.	initial state check	
	Rotation M1	154F7 KM1-2	New rack in cell	zones	R3326 # parts>0	alarm type	alarm delay
Initial state	*	TRUE	TRUE	-	*	2	0

Operation 123 Rotate Turntable 154, A->B

	internal components				ext.comp.	initial state check	
	Rotation M1	154F7 KM1-2	New rack in cell	zones	R3326 # parts>0	alarm type	alarm delay
Initial state	A	TRUE	FALSE	-	FALSE	2	0
Action 1	A	TRUE	FALSE	b(12)			
Action 2	B	TRUE	FALSE	u(12)			

Table 4 EOP of two of the four alternatives for operation 123, rotate Turntable 154.

There are three internal components in the EOP of operation 123. One actuator, M1 which is an electrical motor handling the rotation of the table. One sensor, 154F7 KM1-2 which is an input to the PLC indicating that the operator safety scanner is not tripped, and one variable, *New rack in cell* indicating a full rack on the side of the table facing the cell. The variable is set true when the turntable is rotated and is set false when the robot has picked the first part. Due to the hierarchical control structure it is not possible for the robot to set the variable false. Instead this has to be done by a new operation, *fetch finished*, that has operation 92 as predecessor. The EOP of operation 123 also has an external component, *R3326 # parts>0*. The external component is a variable in Robot 3326 indicating that the number of parts on the side of the turntable facing the cell is greater than 0.

5.5 Defining the Interlocks, IL

Interlocks are defined for each action of the actuators and in some cases also for operations. The latter is typical for robots since they usually do not have any defined actions smaller than an operation. Table 5 shows the interlocks for moving Fixture 152 to its home position. The actuator doing that is 152Y18 and the action is *go to home position*. The four bottom rows in Table 5 represent four different conditions and at least one of them has to be fulfilled for executing the action *go to home position*. The conditions consist of states of the components of the fixture, booking of zones, states of operations, states of components of other resources, and the type of the product currently in the cell. The first condition is fulfilled if the fixation is unlocked and the part sensors are unaffected. If the part sensors are affected and the clamp closed as in the second condition, operation 63 and 98 may not be started for fulfillment of the condition. In operation 63 and 98 the plate in the fixture is welded to the floor.

Actuator / operation id			152Y18							
Action / operation comment			go to home position							
Resource			FIX152							
internal component states					collision zones		operations		ext.comp.st.	
Position Y18	fixation Y16	clamp Y14	part sensors SG3	part sensors SG6	before	after	not started	not ongoing	resource component	Product
*	unlocked	*	off	off	-	-	-	-	-	-
*	unlocked	closed	on	on	-	-	63,98	-	-	S80
*	unlocked	closed	on	on	-	-	64,99	-	-	V70
*	unlocked	open	*	*	-	-	-	-	-	-

Table 5 The interlocks for actuator 152Y18 of Fixture 152.

5.6 The Sequence of Operations, SOP

After defining the ROP, the EOP, and the IL, the sequence of operations, SOP, can be calculated, (Danielsson et al. 2005). Table 6 shows the beginning of an SOP calculated for the product type S80. When an S80 has entered the cell the Coordinator loads the corresponding SOP and distributes the first operations to the stated resources; all in parallel. The Coordinator waits until the specified operations are finished. Then the Coordinator distributes the second group of operations, and the same procedure is repeated until the whole SOP is processed.

Start operation	comment	resource	wait for operation to finish
21	Move in product	CNV415	
31	Ready to receive part?	FIX151	
41	Ready to receive part?	FIX152	
71	Change tool to gripper	R3325	
91	Change tool to gripper	R3326	
111	Part ready on turntable?	TT153	
121	Part ready on turntable?	TT154	
			71,91,111, 121
72	Fetch part from turntable	R3325	
92	Fetch part from turntable	R3326	
			31,41,72,92
73	Put part in fixture	R3325	
93	Put part in fixture	R3326	
112	Fetch finished	TT153	
122	Fetch finished	TT154	
			73,93
32	Close clamps	FIX151	
42	Close clamps	FIX152	
			32,42

Table 6 The beginning of an SOP for the second work cycle of the sample cell.

6 Conclusions and Future Work

Offline verification and information reuse promise to shorten development and reconfiguration times for control systems of flexible manufacturing cells. A clear separation between the information about coordination and the information about execution of operations, together with a standardized representation for this information enables information reuse and simplifies the automation of the verification and control program generation processes. The information comes from the mechanical design of the cell and the robot simulation, two typical tasks in contemporary manufacturing cell design. The standardized representation then permits automatic verification, optimization and conversion to PLC programs, three non-typical tasks in contemporary manufacturing cell design. The focus of this paper has been on applying the modeling approach to a practical manufacturing cell from the automotive industry.

The modeling of the control of the sample cell resulted in 57 operations. Among the operations there were 47 basic ones, 10 alternative ones and no parallel, uncontrolled, or unscheduled operations. The corresponding part of the existing PLC program consists of six sequences, IEC 61131 SFC, with a total number of 39 steps. The number of operations is higher than the number of steps but acceptable and in line with what could be expected since each step of a sequence may contain several actions.

Modeling the sample cell has shown that some parts of the described control information are not normally considered in the mechanical design phase. An example is operation 122, fetch finished, the purpose of which is to update a variable of Turntable 154. In the current workflow the need of updating the variable is detected by the PLC programmer, who also implements the function. In order to include also none-mechanical operations in the control information, the work method of the mechanical design has to be adapted to the modeling method described here. As an alternative, a control engineer could be involved in defining the operations.

The part of the PLC program that have been the concern of this paper is the one that handles the automatic control, which is considered to be a small part, about 10% of a complete PLC program. However, it is this 10% that is the unique part of a PLC program when the programs of a number of cells are compared. Thus, it can be expected that the remaining 90% of a PLC program could be built out of a set of standardized software components.

The modeling of the control also impacts the handling of error situations in a cell. Since the interlocks are separated from the program code and represented by data, it is possible to design standardized software components that interpret the data and give improved operator support and resynchronization functionality.

Ongoing and planned future work involves a verification of the industrial ability of the method and extension of the method to multi-product capability. The verification of the industrial ability of the method is to be done by a study of manufacturing cells from assembly plants within the automotive industry. The study will include mapping of control properties of a number of cells from different processes. In the analysis of the study the cells will be modeled according to the method and the result will be a proven industrial ability of the method, or a need for further development of the method.

In order to increase the general usability of the method a multi-product capability has to be introduced. The consequences for the proposed method of processing several products at the same time in one cell are that the SOP has to be divided and executed by several Coordinators. Then a technique must be developed that calculates when it is allowed to change SOP at different Coordinators, which is necessary when a new product enters the cell. Work on this is ongoing; see Andersson et al. (2006).

References

Andersson K, Richardsson J, Lennartson B, Fabian M (2006). "Synthesis of Hierarchical and Distributed Control Functions for Multi-Product Manufacturing Cells", submitted to IEEE Conference on Automation Science and Engineering, IEEE CASE 2006, Shanghai, China, October 8 – 10, 2006.

Chan FTS, Zhang J, Lau HCW and Ning A (2000) "Object-oriented architecture of control system for agile manufacturing cells" IEEE Conference on Management of Innovation and Technology, Page 863 -868.

Chandra V, Zhongdong H and Kumar R (2003) "Automated control synthesis for an assembly line using discrete event system control theory" IEEE Transactions on Systems, Man and Cybernetics, Part C, Volume 33, Issue 2, Page:284 – 289.

Danielsson K, Richardsson J, Lennartson B and Fabian M, (2005) "Automatic Scheduling and Verification of the Control Function of Flexible Assembly Cells in an Information Reuse Environment", 6th IEEE International Symposium on Assembly and Task Planning.

De Smet O and Rossi O (2002) "Verification of a controller for a flexible manufacturing line written in Ladder Diagram via model-checking", Proc. of the American Control Conference, pp. 4147-4152.

Kanai S, Kishinami T and Tomura T, (2000) "Object-oriented graphical specification and seamless design procedure for manufacturing cell control software development", IEEE Int. Conference on Robotics and Automation, Page 401 –407.

Kollura R, Smith S, Meredith P, Loganautharaj R, Chambers T, Seetharamau G, D'Souza T, (2000) "A framework for the development of agile manufacturing enterprises" IEEE Int. Conference on Robotics and Automation, Page 1132 -1137 vol.2

Kovács GL, Kopácsi S, Nacsa J, Haidegger G, Gruompos P (1999) " Application of software reuse and object-oriented methodologies for the modeling and control of manufacturing systems", Computers in Industry, Volume 39, page 177 – 189.

Lauzon SC, Ma AKL, Mills JK and Benhabib B (1996) "Application of discrete-event-system theory to flexible manufacturing", Control Systems Magazine, IEEE Volume 16, Issue 1. Page(s):41 – 48

Park E, Tilbury D, Khargonekar PP (1999)"Modular logic controllers for machining systems: formal representation and performance analysis using Petri nets", IEEE Transactions on Robotics and Automation, Volume: 15 Issue: 6.

Ramadge PJ and Wonham WM (1989)"The control of discrete-event systems", Proc. of the IEEE, vol. 77, no. 1, pp. 81-98.

Rausch M and Krogh BH (1998) "Formal verification of PLC programs", American Control Conference. Volume 1, 24-26 Page:234 – 238.

Richardsson J (2002) "A Survey of Tools and Methods for Design of Automated Production Plants", 33rd Int. Symposium on Robotics.

Richardsson J and Fabian M (2003a)" Automatic Generation of PLC programs for Control of Flexible Manufacturing Cells "IEEE Int. Conf. on Emerging Technologies and Factory Automation.

Richardsson J and Fabian M, (2003b) "Reuse of information as a Base for Development and Verification of Control Programs for Flexible Manufacturing Cells", IEEE/RSJ Int. Conf. on Intelligent Robots and Systems,.

Sharique F, Gore G, Richardsson J and Fabian M, (2004) "Verification of a Novel Approach in Control of Flexible Manufacturing Cells", 14th Int. Conference on Flexible Automation and Intelligent Manufacturing.

von Euler-Chelpin A, Holmström P and Richardsson J (2004) "A Neutral Representation of Process and Resource Information of an Assembly Cell - Supporting Control Code Development, Process Planning and Resource Life Cycle Management", 2nd International Seminar on Digital Enterprise Technology.

Paper 3

Design of Control Programs for Efficient Handling of Errors in Flexible Manufacturing Cells

Johan Richardsson, Kristin Andersson, and Martin Fabian

Proceedings of the 2004 IEEE Intl. Conference on Robotics and Automation
New Orleans, USA • April 2004

(The paper is reformatted for readability)

Design of Control Programs for Efficient Handling of Errors in Flexible Manufacturing Cells

Johan Richardsson
Control and Computer Systems
Advanced Equipment Engineering
Volvo Car Corporation
Göteborg, Sweden
jrich103@volvocars.com

Kristin Danielsson, Martin Fabian
Control and Automation Laboratory
Dept. of Signals and Systems
Chalmers University of Technology
Göteborg, Sweden
{kickid,fabian}@s2.chalmers.se

Abstract - Insufficient indication of errors is a problem in many manufacturing systems. Lack of support for resynchronization of the cell and its control system is another, less obvious problem. A third problem connected to errors in manufacturing cells is lack of support for manual control. In order to resolve an error situation manual control of the cell is often required. The problem is that some of the manual operations may be blocked due to machine protection. When an operator is to execute a blocked operation the only response is that nothing happens. This paper proposes a method where control programs with integrated functions for error detection, resynchronization, and support for manual control are generated out of information that already exists in the development process of a manufacturing system.

I. INTRODUCTION

A problem in many manufacturing systems is that, due to a malfunction, the work in a cell stops without any indication of the cause of the stop. It is then up to the knowledge and experience of the operator or the maintenance personnel to identify the problem. In the automotive industry, which is the concern of this paper, and other processes with high volume manufacturing systems it is not possible to spend several hours to find the cause of a stop since the drop in manufactured quantity would be to costly. Therefore, in some cases it is more cost-efficient, but not optimal, to scrap the material and reinitialize and restart the cell. Besides the down time of the manufacturing system this type of problem also causes disturbances in the material logistics and additional costs for new material.

Since it is hard to find the reason why cells stop without indicating errors we assume that there are two different reasons. The first one is that the detection of errors is insufficient and the second one is that the control system and the cell have lost their synchronization. The latter means that the control system expects the cell to be in a specific state while the actual state of the cell is another. In such situations the cell usually gets into a locked state without indications of errors.

Regarding insufficient error detection it is clear that in contemporary manufacturing systems of the automotive industry only a small part of all possible errors are monitored. The reason is that the programming effort would be too big if all errors were to be monitored. The programming effort is further increased by a common trend in automotive industry, shortened product life cycles. A consequence is more frequent introductions of new products into the manufacturing system, and consequently more frequent rebuilds of the cells and reprogramming of the error monitoring functions.

Error recovery in manufacturing systems has been treated by several researchers. An overview of different techniques can be found in [1]. In [2], the authors propose a mathematical framework to deal with error monitoring and error recovery in manufacturing systems. In [3], a method for detection and prediction of errors is presented. A template

monitoring method is used for fault detection, and the results from the detection are further processed by statistical process control (SPC) objects. The SPC objects evaluate the system behavior over time, and may issue fault prediction warnings. In [4], [5] and [6], the authors develop a method for diagnosis in discrete event systems using a diagnoser, which is a refined form of an observer. They also discuss diagnosability of discrete event systems. The concept of a diagnoser is extended to timed discrete event systems in [7]. A different approach to diagnostics is used in [8], where fuzzy reasoning and $\overline{X}$ control charts are used. Cause-symptom relations are represented in a fuzzy relation matrix, which is constructed from expert's knowledge or a historical data set. When a fault has been diagnosed, methods for recovery from the error must also be applied to the system. In [9], two methods are developed and combined to implement the whole fault reaction loop from detection to the proposal of a new organization of the manufacturing system. The method used for diagnostic and prognostic procedures is based on a model called a functional graph, which provides a graphical description of logical conditions. In the recovery procedure, a method based on an operational accessibility graph (OAG) is used. The OAG models the operations of a manufacturing system. It is used in the recovery procedure to provide knowledge about the sequences of operations that allow applying a reconfiguration decision. [10] and [11] present a development method for control programs with integrated error handling. When an error occurs, it is handled by the control system itself, but in a separate control mode. The machine is controlled in an alternative way and the error is resolved. [12] presents two diagnosis models for control systems for manufacturing systems, one model for combinatorial logic and one for sequential logic. [12] also presents an example of a diagnosis method from the fault-tree analysis field. The propagation process of the system faults is described in the form of a tree, and different diagnosis models are applied on different levels of the fault tree. A rough position where there is a fault can be located using the state signals in the PLC. However, further diagnosis must be carried out by expert experience. The fault must be analyzed layer by layer till the final cause of the fault is found. An other popular approach to error recovery is the use of artificial intelligence (AI). An example of this can be found in [14]. An intelligent system platform which integrates fault diagnosis and analysis is developed. The diagnosis assists the operators in finding the cause of the error by providing suggestions concerning the state of the components. In [15] the issue of restart after the occurrence of an error is discussed. A control system architecture (CHAMP) is presented, which enables the fast restart of a machining cell through the use of restart points. After the resolution of an error, the prediction of the actual production state by the control system may be difficult, especially if the error has been resolved by human intervention. Depending on where in the process a fault has occurred, the system is restarted from different restart points.

In the manufacturing industry, shortened product life-cycles and increasing demands for flexibility increases the need for automatically generated control programs, see e.g. [17]. Many of the existing methods for diagnosis and restart need additional programming, which may induce a significant cost. This is especially true if implementation aspects have not been a major concern in the theoretical development of the methods. There are also few methods that include the whole error recovery loop, from detection of a fault to resynchronization of the system. Often the system has to be manually resynchronized after the fault diagnosis, without any further support from the control system. Finally, since the manufacturing industry often works with continuous improvements, the errors that occur are often previously unknown errors that do not follow any pre-known statistical pattern.

In the method for development of control programs proposed in this paper, functions for an extensive detection of errors, resynchronization, and support for manual control is integrated and obtained without additional programming.

II. THE PROCESS

The process that is the concern of the proposed method is the body shops in the automotive industry. The body shop of a car factory is usually highly automated and consists of a large number of cells, usually organised in lines. Figure 1 shows a 3D-model of a cell with three robots assembling a part of a floor of a car. Inside a cell there are a number of different machines, of which the most common are robots, fixtures, racks, and pallets. The robots are equipped with tools for different types of welding, screwing, material handling, and application of glue and sealing material. The task of a rack is to hold a body part in a predetermined position, with good precision, and thereby make it possible for the robot to pick the part in a reliable way. The fixture is a robust machine that clamps a body-part and moves it between two or more fixed positions. When a part is to be welded to the body a robot picks the part from a rack, puts it in a fixture which positions the part against the body, and finally a robot welds the part to the body. The positioning cannot be done by a robot since they do not have the required precision. And finally, the pallet is a kind of fixture but for larger parts like a side or a floor. A pallet also moves the part from cell to cell while a fixture only moves within a cell.

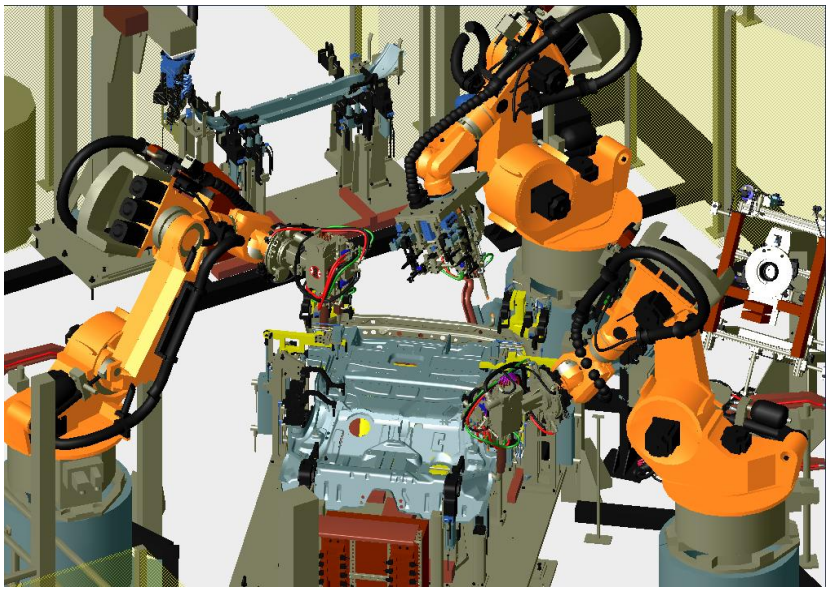

Figure 1. Cell with three robots

A manufacturing cell includes a number of control systems. Each robot has its own, and there is one that coordinates the work in the station and controls other devices, such as fixtures, racks and pallets. The control systems for the robots are called robot controllers and the coordinator is typically a PLC (Programmable Logic Controller). A PLC is a robust computer, specially designed to control manufacturing devices.

This paper describes only the development of the program for the PLC.

III. THE PROPOSED METHOD

The method presented in this paper builds on a previously presented method [16], [17]. The main purpose of the previously presented method is to make it possible to generate control programs for a cell out of information that already exists in the development process of the product and its manufacturing system. In this paper functions for automatic resynchronization and operator support are added and it is explained how the previous method supports the scope of this paper, to eliminate situations when the work in a cell stops

without any indication of the cause of the stop. The parts of the previous method that are interesting for the scope of this paper are error detection and how the program structure prevents the cell and its control system to lose synchronization.

The basic idea of the program generation is to "assemble" the control program out of objects representing the devices in a cell and basic cell functions. These objects are instantiated from a library of standardised software components. The control function, that constitutes the "glue" that coordinates the objects, can then be loaded as data to the PLC and interpreted by a standardised software component, a general sequence function. The control function consists of a set of small operations and a list that defines the order of the execution of the operations. An operation represents a part of the work cycle of cell and it consists of two parts. One part is the control logic and the other part is information about how and when the operation is to be executed. The logic part of an operation holds all sensor and actuator states a machine passes through while an operation is executed. The second part of an operation is used to calculate a collision-free, deadlock-free and time-optimised scheduling of the operations.

The use of standardised software components will improve the performance of the system, since it is then worth investing some extra effort in the implementation of a standard component. Improved error detection, extended state information and configuration possibilities are examples of functions that are normally not implemented due to high initial costs. Reusing such an implementation, on the other hand, will spread those costs over several implementations, and thus the cost can be motivated.

A. Interlocks in manual mode

A common situation connected to errors is that an operator temporarily takes over the control of the cell. A reason could be to drive a machine to a specific position to make it possible to replace a broken component. After such an operation the synchronisation between the cell and the PLC is lost and it could be a difficult and time consuming task to manually put the cell in exactly the same position as before the manual intervention.

One reason for the difficulties connected to manual operation is the interlocks. Interlocks are used to ensure a basic rule of manual operation, that an operator shall not be able to damage the cell no matter how the control is performed. The difficulties arise when the operator orders a machine to execute an operation and the order is blocked by the control system due to an interlock that is not fulfilled. The problem is the lack of response from the control system when an order is blocked. The only feedback on operator gets when trying to execute a blocked order is that nothing happens. The reason is the same as the one for not monitoring all the possible errors, as mentioned in the introduction, the programming effort would be too big.

In the method proposed here the interlocks are separated from the program code and represented by data in a neutral format. Thereby the interlock function can be handled by a standardised software component that interprets the data. No additional programming will be needed to get information about which operations are allowed to be executed and reasons why other operations are not allowed. The information about the availability of the operations will be used both by operators and by the resynchronization function.

The fixture in Figure 2 serves as an example when the interlocks for manual mode are described. The fixture is equipped with 3 actuators, A, B, and C, and one separate sensor D. A is a cylinder that moves the fixture between three different positions. R is the home position where a robot puts a plate in the fixture, which clamps the plate with actuator C. The clamp C has two positions. Open is represented by R and closed by S. Actuator B is a conical fixation pin that ensures an exact positioning of the fixture. R means that the fixation is

unlocked and S means that the fixation is locked. When the fixture has clamped a part it moves to a work position, S1 or S2 depending on the current type of product. D is a sensor that detects the presents of the plate. The actuators also have sensors but they are, in an object oriented manner, integrated in the actuators. The reason why the part sensor is treated in a different way is that it is said to be uncontrollable. It is not possible to control the state of the part sensor by any of the actuators of the fixture.

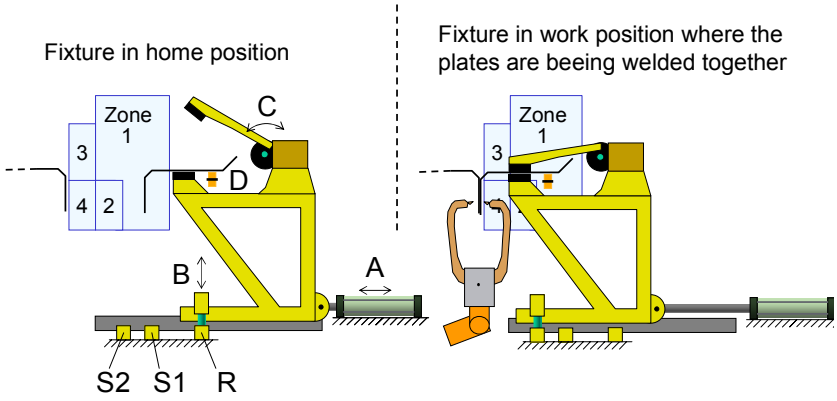

Figure 2. Fixture positioning a plate.

The interlocks are divided into internal and external interlocks. Internal interlocks handle the safety when components of a machine interact and the external interlocks handle safety when different machines interact. An example of an internal interlock is that the fixture is not allowed to move if the fixation is locked. The main part of the interlocks between machines is handled by a zone booking system. A zone represents a defined volume in the cell that a machine must book if it is to be allowed to move into the zone. Since only one machine at the time can book a zone collisions are avoided. Besides the zone booking system, the safety when machines are interacting is also handled by keeping track of which operations are started and which ones are finished. This information is available at the Coordinator.

Table 1 shows some of the interlocks in manual mode for the fixture in Figure 2. The three rows in Table 1 shows the interlocks for executing operation S2 of actuator A, which means to move the fixture to position S2. If it is to be allowed to move component A to state S2 the components of the fixture must have one of the three displayed combinations of states. For instance, it is allowed to move A to state S2 if B and C both are in state R, according to row one. Also, the external interlocks must be fulfilled and this means, for row one, that zone 1 and 2 must be free and operation 7 must not be started or already finished. Operation 7 means that a robot puts a part in the fixture and as long as that interaction is ongoing it is not allowed to move the fixture to position S2. The difference between before and after in the zone-column is that some zones can be cancelled after the operation is done when the machine are just passing through those zones. The meaning of the third row is that it is allowed to move the fixture to position S2 if the fixation is unlocked, the clamp is closed, a part is present, operation 14 has not started and operation 7 is not ongoing. Operation 14 is the irreversible operation of welding the plate on the fixture to another one. In this case it means that the plate has already been welded on the product currently in the cell and if the fixture moves to S2 with another plate it will result in a collision.

Internal interlocks component state				External interlocks zone		operations not started	operations not ongoing	external machine state
A	B	C	D	before	after			
*	R	R	*	1,2	1,2	-	7	-
*	R	S	off	1,2	1,2	-	7	-
*	R	S	on	1,2,3,4	1,2,3	14	7	-

Table 1. Interlocks for operation →S2 of component A.

B. The Program Structure

The program structure is object oriented [19] but has a hierarchical control structure. Modern PLC programming languages do not fully support object oriented principles. At best, the only construct available for implementation reuse is containment rather than inheritance. The unique parts of a control program, in this case the interlocks, are separated from the program and thereby it is possible to instantiate the program out of a set of standardised software components. Information about which software components that is to be instantiated is imported from the mechanical design of a cell. Likewise are the interlocks defined by a mechanical designer but they are loaded to the PLC as data that is interpreted by a standard component in the PLC, the *Manual Control* object.

C. Error detection

The following three types of alarms will detect the errors in all controlled actions. The first type is used when a component is to execute an order. Before an order can be executed the mechanical conditions, the *interlocks*, must be checked, and only if they are fulfilled will the object execute the order. Otherwise, if the interlocks are not fulfilled, an alarm will be generated. Since the state of all actuators and sensors are stored in an operation it will be possible to detect errors by comparing the current state of the cell with the states of the stored operations. The second kind of alarm concerns the execution of an operation, and is activated if the component fails to carry out the order. For instance, whether the operation is done too quickly or is not finished within the expected time. The third kind of alarm is used to check that a component remains in its position. If a component loses its position due to a mechanical fault or a manual intervention then an alarm is generated. Details about the error detection are found in [16].

D. Preventing loss of synchronization

The control program of a manufacturing cell is usually realized by a set of sequences. A sequence represents a series of actions with a transition condition between each action. When an action is finished and the following transition condition is fulfilled, then the next action is allowed to start. It is assumed that loss of synchronization can be caused by an incorrectly trigged transition, due to a transient error, an unexpected signal, or a program bug.

In the program structure of the method presented in [16] there are no explicit sequences. Instead there is a Coordinator with a sequencing function. The purpose of the Coordinator is to coordinate the machines in the cell. When a new product has entered the cell, the Coordinator loads the corresponding operation list and starts to, via messages, distribute the orders to the machines which activate their operations. The only possibility for the Coordinator to lose track of the cell is if a machine sends an acknowledgement of a finished operation before the operation is finished. This could happen, as a consequence of a transient error but then the machine would set off an alarm, according to section C. The cell would then stop and it would not be possible to start it in auto mode until the error is fixed. Then it is assumed that the only way the system can be unsynchronized is by a manual intervention.

E. A Machine Object

Figure 3 shows the internal objects of an object that represents a machine in a cell, e.g. the fixture in Figure 2. The machine object consists of a number of standard objects, identical in all machine objects. Then there are a set of specific objects, actuators and sensors that adapt the machine object to the mechanical composition of a specific machine. Finally there are some specific data, operations that represents the behaviour of a specific machine and interlocks that describes when operations are allowed to be executed in manual and in resynchronisation mode. The objects and data that are of interest for the scope of this paper are *Message Handler, Manual Control, Mail Box, Resynchronisation Control, Machine Model, Operations, Interlocks, and an object outside the machine object, the Coordinator*. Information about the remainder of the objects inside a machine object and the objects at the top level of a PLC program is found in [16].

In the machine object we have, from the left, the standardised interface with four connections, which handle all communication between an object and its surrounding environment. The *Message Handler* manages the message exchange with the *Coordinator* and in some cases also with other machines. When the machine is to execute an operation, a message comes in from the *Coordinator* to the *Message Handler* which forwards it to the *Automatic Control* object. The operation is loaded by the *Automatic Control* object, which then starts to control the actuator objects according to the states in the operation. In manual mode the *Manual Control* object evaluates the interlock information and indicates the operations that currently are allowed to be executed. For additional operator support the *Manual Control* object is also able to show why some operations are not allowed to be executed.

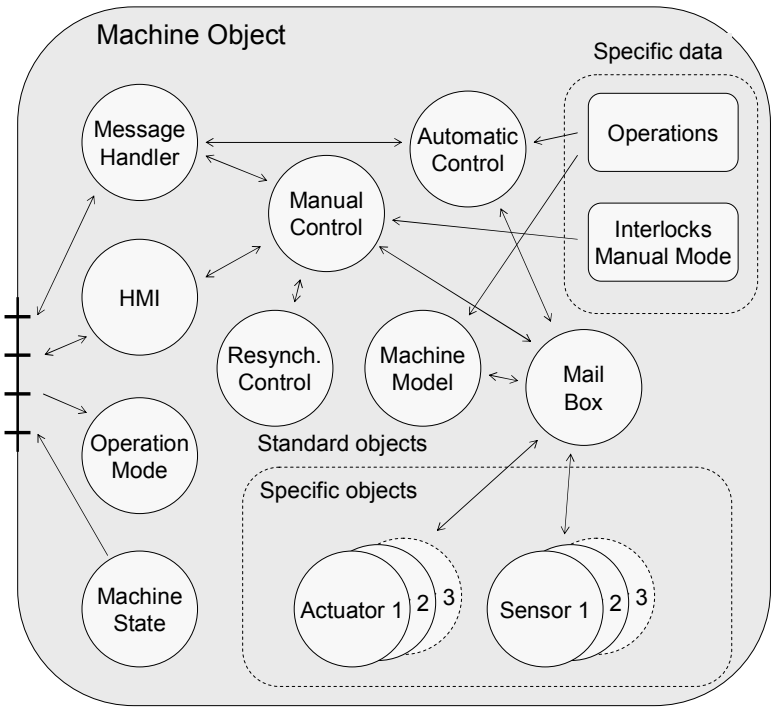

Figure 3 Objects inside a machine object

During the calculation of a resynchronisation schedule the *Machine Model* is used to hold different states of the machine and make it possible to apply the interlocks and detect which operations are possible to execute at certain machine states. The *Machine Model* consists of logic models of the actuators and the separate sensors of a machine. The actuators simply change state when they are ordered to but the separate sensors are more difficult to control

since they are not directly controlled by the actuators of the machine. However, information about when the separate sensors are supposed to change state is stored in the operations for automatic control. When the *Machine Model* changes the state of an actuator it checks the operations for any connected state change of separate sensors and update the sensors accordingly. The task of the *Resynchronisation Control* object is to create a schedule of operations that takes the machine from its current state to the state when manual mode was entered. The last object to describe is the *Mail Box*, a mutual message queue used to pass data between the objects. As described in [21], a mail box is possible to implement in any PLC.

F. Automatic resynchronisation

The principle of the proposed resynchronisation function goes as follows. The state of a machine is stored when auto mode is interrupted and manual mode entered. When the manual operation is finished and the state of the cell is different from the state of the PLC the resynchronisation function can be activated. The machine, or machines are then automatically put in the positions they had when manual mode was entered and it is possible to restart the cell in auto mode.

The realization of the resynchronisation function consists of three parts. First some restrictions of the manual control have to be introduced. Secondly, conditions for interlocks in manual mode have to be defined and finally the synchronisation algorithm.

The following algorithm executes the resynchronisation.

1. When manual mode is entered; the current state of the machine, the resynchronisation state, is copied to the *Machine Model* and to the *Resynchronisation Control* object.
2. When the resynchronisation function is active; the *Manual Control* object evaluates the interlocks according to the state of the *Machine Model* and if specified in the interlocks, also according to the states of other machines and the *Coordinator*. The result of the evaluation, information about which operations are allowed to be executed, is sent to the *Resynchronisation Control* object together with the current state of the *Machine Model*.
3. If the current state of the Machine Model equals the resynchronisation state, go to step 7.
4. The Resynchronisation Control object selects one of the allowed operations according to the rules presented after the algorithm. The Manual Control object is ordered to execute the selected operation. The selected operation is stored in a schedule.
5. The *Manual Control* object orders the *Machine Model* to adapt to the new state. The *Machine Model* updates its actuators and separate sensor.
6. Repeat from step 2.
7. The *Resynchronisation Control* object orders the *Manual Control* object to execute operations according to the calculated resynchronisation schedule.

The following rules are used by the *Resynchronisation Control* object to select operations.

1. Operations that decrease the difference between the resynchronisation state and the current state of the *Machine Model* are prioritised.
2. Among equally prioritised operations are operations that lead to a state not previously stored in the schedule prioritised in comparison to operations that lead to a state already stored.
3. It is not allowed to select an operation that results in a state that already has been stored if the state previous to the already stored one was the same as the current state of the *Machine Model*.

Resynchronisation of a single machine, as described above, is a simple function that can be implemented in any PLC. When a resynchronisation involves several machines the complexity grows since the machines may block each other and the may end up in a deadlock.

In such situations the sequence of operations is calculated for each machine as described above, without respect to the zones. Then all the sequences together with information about which zones each operation needs for its execution are exported to an external and more powerful computer than the PLC. Then, each list is processed for a collision-free and deadlock-free scheduling of the operations. Algorithms for the calculation are presented in [18], and a tool, developed at Chalmers, able to perform the calculations is presented in [20]. The output from the calculation is a list that shows the execution order of the operations.

It is not guaranteed that the above algorithm gives an optimal resynchronisation schedule. The main purpose at this time is to show that it is possible to automatically generate a resynchronisation function for PLC.

A limitation of the proposed method is that it is not, in manual mode, allowed to execute irreversible operations nor operations where several machines are interacting. However, a system where it is not possible to manually execute all operations would not be feasible. If an operator chooses an irreversible operation he or she should get a warning indicating it is not possible that the cell and the PLC can be automatically resynchronized after the chosen operation is started.

IV. CONCLUSION

The program structure proposed in this paper provides three results that contribute to a more efficient handling of errors in flexible manufacturing cells.

Firstly, the robustness of the proposed program structure prevents loss of synchronisation between the cell and its control system. In the only situation a loss of synchronisation is expected, after a manual interaction, an automatic resynchronisation function is offered by the proposed program structure.

Secondly, since the interlocks are separated from the program code and represented by data, interpreted by a standardised software component, it is possible to improve operator support. In manual mode the operations that are allowed to be executed and reasons why other operations are not allowed can be presented.

Thirdly, since the automatic control of a cell is based on operations that hold the states of all actuators and sensors it will be possible to detect errors by comparing the current state of the cell with the states of the stored operations.

Moreover, the fact that it is possible to separate the interlocks from the program code is important for further research. Information about the constraints of a manufacturing cell is available to tools for mathematical manipulation of the control function of a cell.

Finally, the new functions for an extensive detection of errors, resynchronization, and support for manual control are obtained without additional programming. This is integrated in a control program that is generated out of information that already exists in the development process of a manufacturing system.

V. REFERENCES

[1] P. Loborg, “Error Recovery in Automation – An Overview, “ presented at AAAI-94 Spring Symposium on Detecting and Resolving Errors in Manufacturing Systems, Stanford, Ca, USA

[2] A. I. Kokkinaki and K. P. Valavanis, “Error Specification, Monitoring and Recovery in Computer Integrated Manufacturing: An Analytic Approach,” IEE Proceedings – Control Theory Applications, vol. 143, no. 6, 499-508, 1996

[3] H. K. Fadel and L. E. Holloway, “Using SPC and Template Monitoring Method for Fault Detection and Prediction in Discrete Event Manufacturing Systems,” Proceedings of the 1999 IEEE International Symposium on Intelligent Control/Intelligent Systems and Semiotics, 1999

[4] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen and D. C. Teneketzis, "Failure Diagnosis Using Discrete Event Models," IEEE Transactions on Control Systems Technology, vol. 4, no. 2, 105-124, 1996

[5] M. Sampath and S. Lafortune, "Active Diagnosis of Discrete-Event Systems," IEEE Transactions on Automatic Control, vol. 43, no. 7, 908-929, 1998

[6] S. Lafortune, D. Teneketzis, M. Sampath, R. Sengupta and K. Sinnamohideen, "Failure Diagnosis of Dynamic Systems: An Approach Based on Discrete Event Systems," Proceedings of the 2001 American Control Conference, vol. 3, 2001

[7] Y-L. Chen and G. Provan, "Modeling and Diagnosis of Timed Discrete Event Systems – A Factory Automation Example," Proceedings of the 1997 American Control Conference, vol.1, 31-36, 1997

[8] H-M. Hsu and Y-K. Chen, "A Fuzzy Reasoning Based Diagnosis System for Control Charts," Journal of Intelligent Manufacturing, vol 12, no. 1, 57-64, 2001

[9] A. K. A. Toguyéni, P. Berruet, E. Craye, "Models and Algorithms for Failure Diagnosis and Recovery in FMSs," International Journal of Flexible Manufacturing Systems, vol. 15, no. 1, 57-85, 2003

[10] E. Park, D. M. Tilbury, P. P. Khargonekar, "A Modeling and Analysis Methodology for Modular Logic Controllers of Machining Systems Using Petri Net Formalism", IEEE Transactions on Systems, Man and Cybernetics, Part C, vol. 31, no. 2, 2001

[11] S. S. Shah, E. W. Endsley, M. R. Lucas and D. M. Tilbury, "Reconfigurable Logic Control using Modular FSMs: Design, Verification, Implementation, and Integrated Error Handling," Proceedings of the 2002 American Control Conference, vol. 5, 4153-4158, 2002

[12] W. Hu, A. G. Starr and A. Y. T. Leung, "Operational Fault Diagnosis of Manufacturing Systems," Journal of Materials Processing Technology, vol. 133, 108-117, 2003

[14] M. Rao, X. Sun and J. Feng, "Intelligent System Architecture for Process Operation Support," Expert Systems with Applications, vol. 19, no. 4, 279-288, 2000

[15] M. Tittus, S-A. Andréasson, A. Adlemo, J-E. Frey, "Fast Restart of Manufacturing Cells Using Restart Points," presented at 4th World Automation Congress, Wailes, Maui, June 11-16, 2000

[16] Richardsson, J. Fabian, M. "Automatic Generation of PLC programs for Control of Flexible Manufacturing Cells", Int. Conf. on Emerging Technologies and Factory Automation, 2003

[17] Richardsson, J. Fabian, M. "Reuse of Information as a Base for Development and Verification of Control Programs for Flexible Manufacturing Cells" IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2003.

[18] Liljenvall, T., "Scheduling for Production Systems"; Lic. Thesis, Chalmers University of Technology, 1998.

[19] Shlaer S. Mellor S. "Object Lifecycles", Yourdon Press, 1992.

[20] Åkesson, K. "Supremica - A Tool for Verification and Synthesis of Discrete Event Supervisors" 11th Mediterranean Conference on Control and Automation, 2003. Also: http://www.supremica.org

[21] Laplante, Phillip A., "Real-time systems design and analysis", page 173, IEEE Press, cop. 1997

Paper 4

Reliable Control of Complex Manufacturing Cells

Johan Richardsson, Kristin Andersson, and Martin Fabian

Proceedings of the 2007 IEEE International Symposium on Assembly and Manufacturing
Ann Arbor, USA • July 2007

(The paper is reformatted for readability)

Reliable Control of Complex Manufacturing Cells

Johan Richardsson, Kristin Andersson, and Martin Fabian

***Abstract*— Lack of tools for analyzing the control of complex manufacturing cells is a problem which consequences are functional errors and in some cases a lower throughput than expected. A typical property of complex cells is processing of several products at the same time. This paper presents a model for automatic generation of control programs based on the output of a previously presented algorithm for calculation of the control of multi product cells.**

I. Introduction

DEVELOPING control programs for flexible manufacturing cells is usually not considered as an exceptionally difficult task. Eventually the programmer manages to write a program that puts the cell in operation. However, there are cells where developing the control program is a difficult task. Typical properties of those cells are processing of several products at the same time and that it is not possible to separate the cell into smaller functional cells processing one product at the time, not without reducing the throughput. An example of a difficult task is to manually schedule the work of a cell containing eight robots which process between one and six products at the same time. Even here the programmer will eventually make a program that works but it is not guaranteed that the programmer will find the optimal way to control the cell.

Another problem in development of control programs, and in most cases of software development in general, is that the program development process includes a test phase, where the program is verified. In the case errors are found, the program is changed to avoid the errors, again tested and the loop is repeated until the program passes the test. When flexible manufacturing cells are concerned the test phase of the program development process can be very costly since it usually disturbs ongoing production. A typical situation is when one product type replaces another in a manufacturing system producing several different types of products.

Reducing disturbances in manufacturing systems connected to introductions of new product types has been the incentive of the authors' previous research in this area, which has resulted in a method for automatic generation of mathematically verified control programs, [1], [2], [3]. An algorithm extending the previously presented method to include cells processing several products at the same time is presented in [4]. Besides increasing the number of cells to which the method is applicable the new multi product capability also widens the functional scope of the method; from automatic verification and generation of control programs to automatic calculation of complex control functions. This paper presents a model for implementation of the output of the new algorithm, [4], for control of multi product cells.

The main idea of the previously presented method, [1], [2], and [3] is to reuse information from different tools in the development process of a manufacturing system, to process that information in tools for verification and optimization and finally to convert the reused and processed information into PLC-programs.

Several papers have been written about development of control programs for automated

Johan Richardsson is with Volvo Car Corporation, Göteborg, Sweden(phone: +46 31 325 73 15; e-mail: jrich103@volvocars.com).
Kristin Andersson is with the Department of Signals and Systems, Chalmers University of Technology, Göteborg, Sweden (e-mail: kickid@chalmers.se).
Martin Fabian is with the Department of Signals and Systems, Chalmers University of Technology, Göteborg, Sweden (e-mail: fabian@chalmers.se).

manufacturing. These can be roughly divided into two types. The first type [5], [6], [7], could be classified as based on software engineering and the second type [8], [9], as based on mathematical system representation and manipulation. The implementation model presented in this paper is based on software engineering principles and adapted to import information processed by formal mathematical methods.

There are four reasons why the previous research does not solve the problems connected to automatic verification and generation of control programs. First, in [5], [6], [8] and [9] only the part of the program that involves the control function for automatic operation is concerned. This is typically 10% of a complete control program and it is necessary to include parts about manual interaction and mechanical interlocks for reliable offline verification. A description of the remaining 90% is found in [2]. Secondly, information reuse is not considered. The logic is manually specified in a programming tool, [5], [6], by controlled-automata, [8], or by Petri nets, [9]. Thirdly, since the logic is manually specified it is not possible to guarantee that the specification is uncorrupted after conversion to a control program. Fourthly, [5], [6], and [8] specifically deal with automated manufacturing cells, but they do not cover the implementation of the control programs of the individual devices. The reason for this is clear: the devices involved are mills, lathes and similar, and they all have their own control systems. In the processes considered in this paper, this is not the case.

II. The Example Cell

Figure 1 shows a cell that serves as an example throughout the paper. The cell has no physical counterpart but is composed to be simple and still show increased complexity in a clear way when then number of simultaneously processed products increase from one to two.

The example cell, Figure 1, has three basic behaviors. In the first one, normal operation, two parts are welded together and transported out of the cell. In the second one the resulting part is put on a trolley to be taken out of the cell for manual inspection. In the third one an inspected and approved part is put back into the cell.

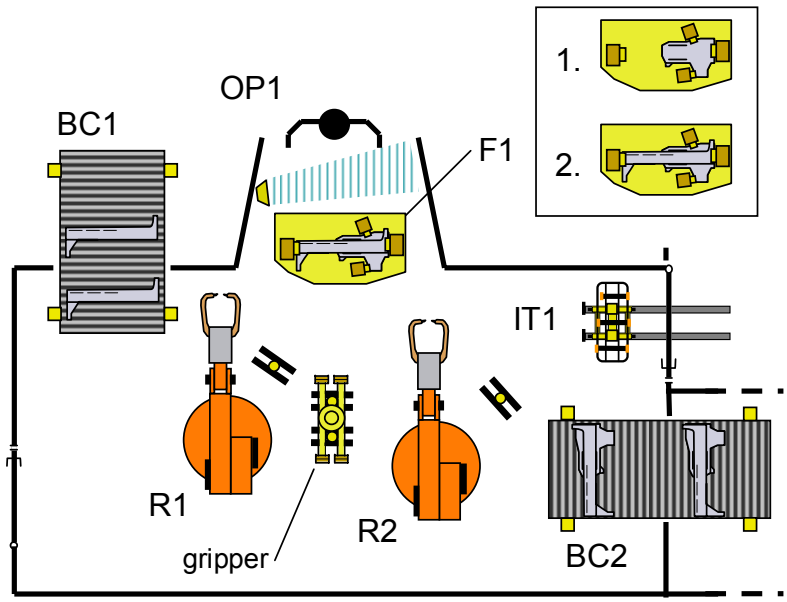

Figure 1 An assembly cell with two robots

In normal operation an operator, OP1, puts part one in the fixture, F1, and belt conveyor 1, BC1, drives part two to a predefined position where it will be picked up by robot 1, R1. After replacing its weld gun for the gripper, R1 picks part two from BC1 and puts it in F1. The upper right part of

Figure 1 shows how part one and two are loaded into F1. When R1 has released part two and moved out of F1 robot 2, R2, starts to weld the two parts together. R1 leaves the gripper, attaches its weld gun and joins R2 in welding the parts together. When R2 is finished welding it replaces its weld gun for the gripper and waits for R1 to finish welding. R1 finishes welding and moves out of F1. R2 then picks the new part from F1 and puts it on belt conveyor 2, BC2, which drives the part to the next cell.

The second basic behavior of the cell is the same as normal operation except for the last part. R2 does not put the welded part on BC2 instead it puts the part on the inspection trolley, IT1. In the third basic behavior R2 picks a checked and approved part from IT1 and puts it on BC2.

III. The Previously Presented Method

This section briefly describes three parts of the previously presented method for automatic generation of error free control programs, [1], [2], and [3].

A. Information for controlling a cell

An important aspect of the method is to decompose the information about the control of a cell. Decomposition of the control information means that a complete schedule of a cell, a timing diagram from the mechanical design or a SOP (sequence of operations), from the robot simulation is broken down into a set of operations, each accompanied with scheduling information. The decomposition gives the possibility to schedule the operations in a time optimized way and so that collisions between the resources are avoided and that other requirements of the behavior of the cell will be fulfilled. The information needed to describe the control of a cell is divided into eight categories,

1. Declaration of Operations, DOP
2. Relations of Operations, ROP
3. Execution of Operations, EOP
4. Cycle Start Condition, CSC
5. Interlocks, IL
6. Coordinated Operations, COP
7. Mechanical Components and Functions, MCF
8. Function Blocks, FB

The DOP (Declaration of Operations) and the ROP (Relations of Operations) hold information about when an operation can be executed while the EOP (Execution of Operations) holds information about the states a resource passes through while an operation is executed. Information about how a resource is to be controlled to go between different states is stored in FB (Function Blocks) as program code. The CSC (Cycle Start Condition) is used sort operations into different work cycles, which are triggered by different combinations of states of a set of the components of the cell. The three basic behaviors of the example cell are represented as three CSC, A, B, and C, where CSC A is normal operation of the cell. IL (Interlocks) describe the conditions, in terms of states of the components in the cell, that have to be fulfilled before a resource can execute an operation without causing any damage to the cell.

A part of the DOP and the ROP of the example cell are shown below. Table 1 shows all operations for the CSC A, normal operation, but not all of the operation attributes. Examples of attributes not shown in Table 1, but important for the discussion, are the duration of the operations and the collision zones. A collision zone represents a physical volume in the cell where only one resource may be at a time to avoid collisions.

id	comment	resource	id	comment	resource
10	Initial state check	BC1	41	Put part 1 in F1	OP1
11	Drive part to pick up position	BC1	42	Part picked by R2	OP1
12	Part picked by R1	BC1	50	Initial state check	R2
20	Initial state check	R1	51	Weld parts	R2
21	Get gripper	R1	52	Get gripper	R2
22	Pick part from BC1	R1	53	Clamp part in F1	R2
23	Put part in F1	R1	54	Pick part from F1	R2
24	Release part in F1	R1	55	Put part on BC2	R2
25	Get weld gun	R1	56	Leave gripper	R2
26	Weld parts	R1	57	Get weld gun	R2
30	Initial state check	F1	70	Initial state check	BC2
31	Close clamps part 1	F1	71	Ready to receive part from R2	BC2
32	Close clamps part 2	F1	72	Drive part from robot drop position	BC2
33	Open all clamps	F1	89	No action	IT1
34	Part picked by R2	F1	72	Drive prt from robot drop position	BC2
40	Initial state check	OP1	89	No action	IT1

Table 1. A part of the DOP for CSC A of the example cell.

Figure 2 shows a graphical representation of the ROP for CSC A of the example cell. The order of the operations for each resource and some required interaction between the resources are defined in the ROP. The arrows between the operations indicate that an operation must be finished before another one can be started. For instance, operation 33 (open the clamps of the fixture) must be finished before operation 54 (the robot takes the part out of the fixture) can start.

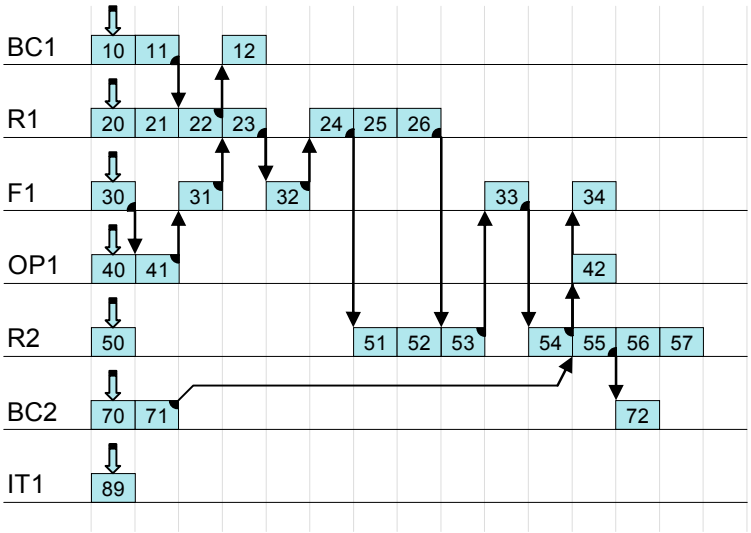

Figure 2. The complete ROP for CSC A of the example cell

DOP, ROP, EOP, CSC, and IL are used as input to the verification tool and the output is the COP (Coordinated Operations). For each resource one COP is calculated per CSC. The COP describes an order of the execution of the operations that guarantees that the IL will not be violated and that the DOP and the ROP will be fulfilled. The structure of the COP is the same as that of a ROP, a sequence of operations where the condition to start an operation is that one or several other operations are finished.

MCF (Mechanical Components and Functions) is a list of the mechanical components in a cell and a list of general functions, not directly connected to the mechanical equipment, needed to control a cell. The MCF is an input to the program generation process and it points out which FB's that are to be instantiated in the PLC program of a specific cell.

For illustrating the multi product capability a description of a part of the DOP, the ROP, and the COP are sufficient. Details about the other information types are found in [12] and XML-schemas representing the control information are described in [11].

B. Verification of the Control Information

The verification method is based on the Supervisory Control Theory, [10]. Logical models of the control information of the cell and rules restricting the behavior of the cell are processed. The result is a logical model, a state machine called *supervisor*, of the control of the cell that fulfills the rules. If it is possible to find a supervisor that fulfills all requirements, a set of COPs are extracted from the supervisor. For each resource one COP is extracted per CSC. The difference between a ROP and a COP is that the conditions to start an operation in a COP may require additional operations to be finished. By the design of the verification algorithm, the COP will also be deadlock free. A detailed description of the verification is found in [3].

C. The Control Program Model

The principle structure of the control program model is outlined in Figure 3. When a new product has entered the cell it is detected by the *Product Data Handler* which forwards information about the type of product to the *Coordinators*.

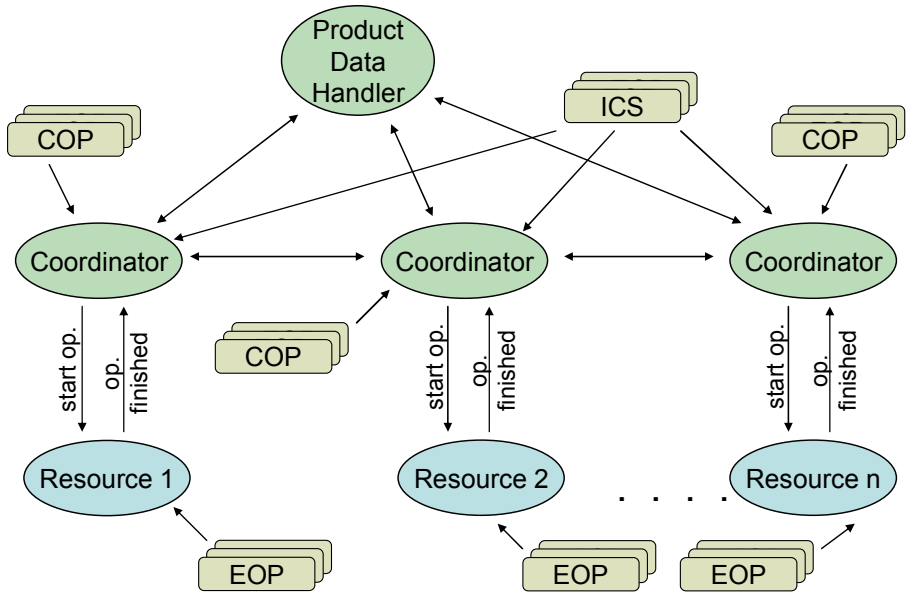

Figure 3 Structure of the control program model

Each Coordinator loads the COP for the current type of product and starts to distribute the operations to its resource according to the COP. When a resource receives an order to carry out a certain operation, it loads the corresponding EOP and executes the operation accordingly. When finished the resource signals back to the Coordinator that the operation is finished and the Coordinator can move on in the COP. This is repeated until all operations in the COPs are executed and the work cycle of the cell is finished.

The Product Data Handler and the Coordinators are standardized FBs while the resources are unique FBs built out of a set of standardized FBs representing the mechanical components of the physical resources. A detailed description of the control program model is found in [2].

IV. Multi Product Capability

In this paper the meaning of multiple products is the capability of handling several product instances at the same time. When one machine is finished processing product *a,* it starts a new cycle before all other machines in the cell are finished with product *a*. Thus, the two parts welded together in the example cell are not considered as multiple products, since they are handled in the same work cycle. The extension of the previously presented method to include multi product capability affects two areas, the verification process and the control program model.

A. Verification of Multi Product Cells

The verification is done in a similar way as described in section III.B. The difference is that one supervisor is calculated for each possible combination of number of products in the cell and CSC. Then one COP per product is extracted from each supervisor. The structure of a COP is also different for multi product cells. Conditions to start an operation are still that one or several other operations are finished but now a product instance number is added to the operations used as conditions. The product instances in a COP are referred to as P1, P2, P3 … where the first product that entered the cell is P1. Since a COP is static, variables representing the actual product in the cell must be relabeled from Pn to Pn-1 when a product leaves the cell. The operations and their order are always the same in the COPs belonging to a specific resource and a specific CSC. The predecessors, on the other hand, may vary and depend on the current number of products in the cell and their CSC.

Given that the example cell can hold at most two products and only CSC A is concerned, the verification process will result in two supervisors and three COP per resource. From the supervisor representing one product in the cell one COP per resource will be extracted and from the supervisor representing two products one COP will be extracted for P1 and one for P2, for each resource respectively.

B. The Control Program Model

Figure 4 shows how the control program model in Figure 3 is adapted to multi product mode. To handle multiple products a new FB is needed, the *Product Instance Handler, PIH*.

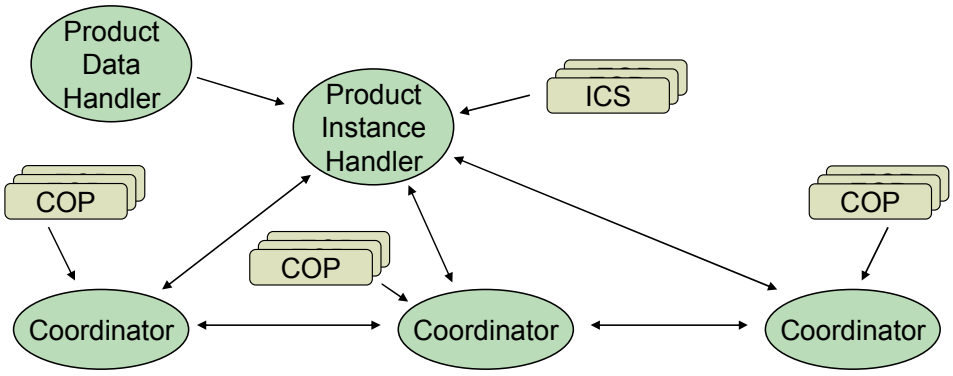

Figure 4 A part of the control program model for multi product cells

When a new product is about to enter the cell the Product Instance Handler, PIH, checks that the number of products in the cell is less than the maximum and if so the PIH evaluates the CSC and selects the one that matches the type of the new product and, if defined, also matches the current state of a set of the components in the cell. Information about the new product is then sent to the Coordinators which load the corresponding COPs and execute them as previously described.

To keep track of the products in the cell and to detect when a product leaves the cell a FIFO register is introduced, Table 2. The cell level product register is managed by the Product

Instance Handler, and the resource level product register, Table 3, is instantiated in all Coordinators and act as a slave to the former register.

product instance	CSC	ID	resource progress state						
			BC1	R1	F1	OP1	R2	BC2	IT1
P2	A	2	done						done
P1	A	1	done	done					done

Table 2. The cell level product register

Each row in the product register represents a product and the meaning of the columns is as follows. The *product instance* is the instance number of a product. The number of a new product will be Pn+1 where n is the highest instance number of the products currently in the cell. The *CSC* is the CSC that was valid when the product entered the cell. *ID* is an indexed local identity number used to keep track of which of the products in the cell the resources have processed. Since the instance number of a product will change while the product goes through the cell the none-changing ID is introduced. The last column, *resource progress state*, shows the current state of the resources that have finished their work on the different products. The column is updated by the coordinators that send a message to PIH every time they complete a COP.

The resource level product register, Table 3, contains the same information as its master with exception for the *resource progress state* that is replaced with the column *finished operations*. Since the conditions for starting an operation are that one or several other operations are finished a record of finished operations must be kept as long as the corresponding product is in the cell.

product instance	CSC	ID	finished operations
P2	A	2	20
P1	A	1	20, 21, 22, 23, 24, 25, 26

Table 3. The resource level product register

When a Coordinator has processed all products in its product register a request is sent to the PIH for a new product. One or several Coordinators could be enabled to send new product requests but usually only the one controlling the resource that brings in new products sends the requests. If it is allowed to bring in a new product in the cell the PIH responds on the request by inserting a new product into its product register. A product is removed from the product register when all resources are finished with their work on the specific product. Every time a product is added or removed from the product register the new values of *product instance*, *CSC,* and *ID* are sent to all coordinators.

V. The Control of the Example Cell

Figure 5 shows the order of the operations of the resources when two products with CSC A are processed and the COPs are selected according to the control program model of Section IV.B. The action is started when BC1 requests a new product. In the example cell only BC1 is allowed to request new products. The PIH responds by inserting a new product in its product register, as shown in the bottom left corner of Figure 5. The changed number of products in the register causes an update of the product registers of the coordinators and all resources start

to execute their COP. Table 4 shows how the selection of COP is depending on the number of products in the cell, their CSC and which products a resource previously has processed. Initially all resources select COP number 1, respectively, according to Table 4.

number of products and their CSC	product to be processed	COP
A	P1	1
AA	P1	2
AA	P2	3

Table 4 Selection of COP depending on cell state

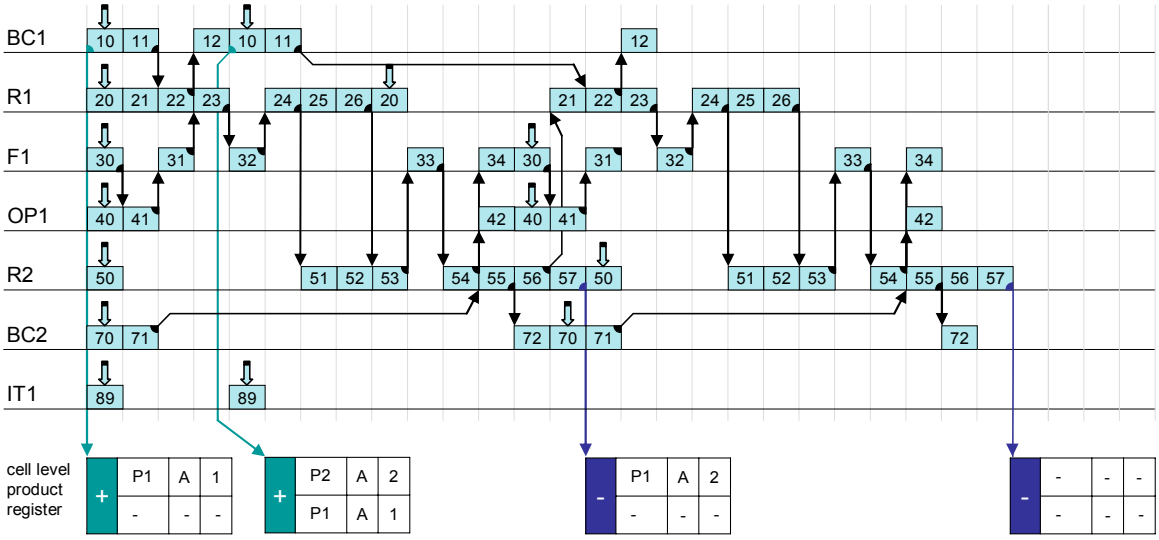

Figure 5. COPs for processing two products with CSC A

When the first COP of BC1 is completed a new product is requested and the product registers of the coordinators are again updated. Now the coordinators of BC1 and IT1 will select and start COP number three according to Table 4. The other resources which still are executing their first COP have to change COPs from number one to two. Since these resources continue to work on the same product the operations and their order will be the same in COP one and two, as mentioned in Section IV.A. Consequently, the physical work in the cell will not be disturbed by the change of COPs. However, COP one and two may be different in the conditions for starting the operations.

An example of when the conditions for starting operations are different for two COPs belonging to the same CSC is COP two and three of R1. Figure 5 shows that COP number three of R1 has a condition for starting operation 21 that is not present in the first COP. The condition is that operation 56 on P1 must be finished before operation 21 on P2 is started. This condition is not defined in the ROP but calculated in the verification process and prevents a prospective deadlock. If R2 is delayed in welding the parts (operation 51), R1 could get the gripper and pick a new part from BC1 but it will not possible for R1 to put the part in F1 since R2 has not removed the old part in F1. R2 will not be able to remove the part from F1 as long as R1 has the gripper and the cell is in deadlock.

The additional condition for starting operation 21 is also an example of increased complexity of the control function when the number of products in the cell increases.

The last resource to finish the first COP is R2. When operation 57 is complete a message is sent to the PIH which removes the product with ID = 1 from its product register, re-labels P2 to P1 and distributes the updated product register to the coordinators. Again the coordinators

have to change COPs. This time all of them will choose COP number one, according to Table 4. When P1 has left the cell all finished operations connected to P1 has to be deleted and those connected to P2 has to be re-labeled to P1. This is done automatically since the record of finished operations is connected to a product ID, Table 3.

The Inspection Trolley, IT1, is not physically involved in CSC A. The reason that IT1 has a COP for CSC A is to ensure that the products in the cell keep their relative position while they go through the cell, a requirement of the verification algorithm.

VI. Future Work

Ongoing work concerns two areas, verification of the industrial usability of the method and development of a time optimization function of the COPs. The industrial usability is verified by mapping existing cell in the automotive industry and representing the control of the cells according to the presented method. The optimization is needed since COPs can include arbitrary order of some operations. An example is when two resources use the same collision zone. In this case it is possible that the COPs can allow the first *or* the second resource to start using the zone. The arbitrary order can be removed by adding additional requirements in the ROP, but a better way is of course to calculate the time optimal order of which resource that is to start using the zone.

VII. Conclusions

One reason for the lack of tools for analyzing the control of complex manufacturing cells is the limitation of Gantt charts and timing diagrams to express variants. A Gantt chart or a timing diagram describing all combinations of the order of operations of a cell processing several products of different types at the same time would be incomprehensible. However, there is no actual need to have descriptions of the order of the operations for all possible combinations of number of products in the cell and their corresponding CSC. The functions that are needed are predictions of the throughput of the cell for the different combinations and a control program able to handle all combinations in a reliable way. Solutions for both functions are provided by the method presented in this paper, without describing all combinations of the order of operations. Since the duration of an operation is included in the DOP it is possible to calculate the cycle time and the throughput for any given combination of number of products and CSC, such as the one in F5.

The implementation model presented in this paper manages to handle the output of the verification process by a set of standardized functions controlling the cell by interpreting data from the verification process. Since the previously presented method for program generation is based on standardized functions the new implementation model will fit into the existing system. Thus the previously presented method can now include automatic verification and generation of control programs for reliable control of complex manufacturing cells.

REFERENCES

1. Richardsson, J. Fabian, M. “Reuse of Information as a Base for Development and Verification of Control Programs for Flexible Manufacturing Cells”, Proceedings of the 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, Las Vegas, USA
2. Richardsson, J. Fabian, M. “Automatic Generation of PLC programs for Control of Flexible Manufacturing Cells”, Proceedings of the, IEEE Intl. Conf. on Emerging Technologies and Factory Automation, Lisbon, Portugal, September 2003
3. Danielsson, K. Richardsson, J. Lennartson, B. and Fabian, M. "Automatic Scheduling and Verification of the Control Function of Flexible Assembly Cells in an Information Reuse Environment ", 6th IEEE International Symposium on Assembly and Task Planning, Montreal, Canada, July 2005.
4. Andersson K, Richardsson J, Lennartson B, Fabian M (2006). "Synthesis of Hierarchical and Distributed Control Functions for Multi-Product Manufacturing Cells", IEEE Conference on Automation Science and Engineering, IEEE CASE 2006, Shanghai, China, October 8 – 10, 2006
5. Kanai, S. Kishinami, T. Tomura, T. “Object-oriented graphical specification and seamless design procedure for manufacturing cell control software development”, IEEE Int. Conference on Robotics and Automation, 2000, Page 401 –407.
6. Chan, F.T.S. Jie Zhang; Lau, H.C.W.; Ning, A. “Object-oriented architecture of control system for agile manufacturing cells” IEEE Conference on Management of Innovation and Technology, 2000, Page 863 - 868.
7. Kollura, R., Smith, S., Meredith, P., Loganautharaj, R.; Chambers, T.; Seetharamau, G.; D'Souza, T., “A framework for the development of agile manufacturing enterprises” IEEE Int. Conference on Robotics and Automation, 2000. Page 1132 -1137 vol.2
8. Lauzon, S.C.; Ma, A.K.L.; Mills, J.K.; Benhabib, B., “Application of discrete-event-system theory to flexible manufacturing” IEEE Control Systems Magazine , Volume: 16 Issue: 1 , Feb. 1996 Page 41 –48
9. Park, E.; Tilbury, D. M.; Khargonekar, P. P. "Modular logic controllers for machining systems: formal representation and performance analysis using Petri nets”, IEEE Transactions on Robotics and Automation, Volume: 15 Issue: 6, Dec. 1999.
10. P. J. Ramadge and W. M. Wonham, “The control of discrete eventsystems,” Proc. of IEEE, vol. 77, no. 1, pp. 81–98, 1989.
11. Ljungkrantz, O. Åkesson, K. Richardsson, J. Andersson, K. "Implementing a Control System Framework for Automatic Generation of Manufacturing Cell Controllers" Proceedings of the 2007 IEEE Intl. Conference on Robotics and Automation, Rome, Italy.
12. Richardsson, J. Fabian, M. "Modeling the Control of a Flexible Manufacturing Cell for Automatic Verification and Control Program Generation", International Journal of Flexible Manufacturing Systems, in printing.