Rapid Virtual Prototyping and Operational Monitoring of PLC-Based Control System

Kwan Hee Han, Sang Hyun Choi, Jun Woo Park and Jun Woo Lee

***Abstract*—**As business environments are rapidly changing, the manufacturing system must be reconfigured to adapt to various customer needs. In order to cope with this challenge, it is quintessential to test industrial control logic rapidly and easily in the design-time, and monitor operational behavior correctly in the run-time of automated manufacturing system. Proposed integrated model for virtual prototyping and operational monitoring of industrial control logic is to improve limitations of current ladder programming practices and general discrete event simulation method. Each plant layout model using HMI package and object-oriented control logic model is designed independently and is executed simultaneously in integrated manner to reflect design practices of automation system in the design time. Control logic is designed and executed using UML activity diagram without considering complicated control behavior to deal with current trend of reconfigurable manufacturing. After the physical installation, layout model of virtual prototype constructed in the design time is reused for operational monitoring of system behavior during run time with slight modifications.

***Keywords*—**Automated Manufacturing System, Object-Oriented, PLC, Simulation, Virtual Prototyping, Reconfigurable

I. Introduction

The unpredictability of market changes, the growing product complexity and continuous pressure on costs force enterprises to develop the ability to respond and adapt to change quickly and effectively. Among their endeavor to overcome these obstacles, one of the frequently prescribed remedies is the automation of factories. Therefore, most enterprises are installing automated manufacturing systems (AMSs) for their competitive advantages to survive the global business environment. As the level of automation increases, material flows and process control methods of the shop floor become more complicated. Currently, programmable logic controllers (PLC) are mostly adopted as controllers of automated manufacturing systems, and the control logic of PLC is usually programmed using a ladder diagram (LD).

More recently, manufacturing trends such as flexible manufacturing facilities and shorter product life cycles have led to a heightened demand for reconfigurable control systems [7, 18]. Therefore, the control system as a brain of AMSs must be reconfigured timely and correctly. To meet these requirements, it is quintessential to design and execute industrial control logic rapidly and easily during the life cycle of manufacturing system.

Currently, in the automation project, it takes long time to execute the control specification. Logic verification is usually conducted during a trial run stage of a project. As a result, it is difficult to timely reflect the control logic changes occurred in the design and implementation stages. Moreover, in order to verify the control logic, it is still necessary to write a program code by low-level language such as a ladder diagram. However, PLC ladder logic gives only microscopic view of the system processes, and lacks semantic and conceptual integrity. Due to this limitation, it is difficult for factory automation (FA) engineers to have overall perspectives about the interaction of system components intuitively.

For verifying and analyzing the design of control logic before its real implementation, quite a few methods based on mathematical formalism or computer simulation are adopted in manufacturing industries. Among these, computer simulation methods are widely used because mathematical formalisms have a problem of solution space explosion as the size of system increases. However, since current simulation methods have mainly focused on the overall performance evaluation of manufacturing systems such as factory layouts, resource utilization, and throughput time, they have limitations with regard to the modeling capabilities of detail logic for the input/output signal level control.

Another problem is that plant layout model and control model are closely coupled in the existing discrete event simulation software. In the shop floor, mechanical engineers and control engineers conduct their own work independently. Therefore, it is needed that plant layout modeling and control logic modeling must be separated.

Moreover, simulation model for verification constructed at the design stage is discarded after the real implementation of AMSs. It is no more used in the operational stage. Reuse of simulation model for operational monitoring is needed.

The new requirements for the design, verification and monitoring of control logic are as follows: 1) it is necessary to have the functionality of modeling the interactions between a system controller and a plant for describing detail control logic

Kwan Hee Han is with the Department of Industrial & Systems Engineering, Engineering Research Institute, Gyeongsang National University, Korea. (e-mail: hankh@gnu.ac.kr)
Sang Hyun Choi (corresponding author) is with the Department of Industrial & Systems Engineering, Engineering Research Institute, Gyeongsang National University, Korea (email: chois@gnu.ac.kr)
Jun Woo Park is with VMS Solutions Corp., Korea (e-mail: junwoo@vms-solutions.com)
Jun Woo Lee is a graduate student in Department of Industrial & Systems Engineering of Gyeongsang National University, Korea (e-mail: junu31@nate.com)

on the shop floor. However, as current simulation methods adopt the 'process view' of work pieces or the 'activity view' of system resources, it is difficult to represent the interactions of system components in detail. 2) For the verification of control logic in an earlier stage of automation project, control logic must be modeled by high-level language which is intuitively understandable. But, current industrial control logic programming is written using low-level language such as ladder diagram. 3) For the simultaneous processing of plant layout modeling and control logic modeling by different disciplines, these two models must be built independently. After construction of each model, control logic verification is conducted by integrating and investigating two models concurrently. 4) In order to support the concept of virtual manufacturing, it is necessary that virtual prototype constructed at the design stage is reused to monitor the system behavior remotely after the real implementation of AMS.

The main objective of this paper is to propose an integrated model for virtual prototyping and operational monitoring of industrial control logic to improve the above mentioned limitation of the current general discrete event simulation method and ladder logic programming. A proposed integrated model during the entire life cycle of AMS in this paper is depicted in Figure 1. First of all, at the design time, control engineers design control logic using high-level language called UML (Unified Modeling Language) based on the system control requirements. Concurrently, mechanical engineers design plant layout model using HMI (Human Machine Interface) software. After that, two models are interfaced, and control logic is executed in the form of UML activity diagram and is animated at the plant model. During the concurrent execution of two models, FA engineers can evaluate and verify the PLC control logic easily and rapidly. During the run time, HMI layout model constructed at the design time is reused for operational monitoring with slight modifications.

The rest of the paper is organized as follows. Section 2 reviews related works. Section 3 describes a proposed O-O control logic design and execution preparation method. Section 4 presents virtual prototyping and operational monitoring of control logic using illustrative example. Finally, the last section summarizes results and suggests directions for future research.

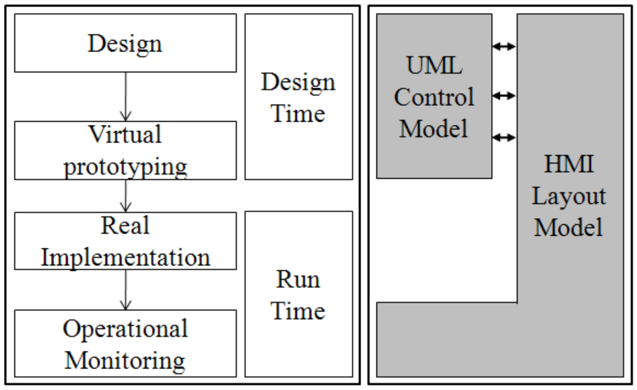


Fig. 1 Integrated model for virtual prototyping and operational monitoring of control logic

II. Related Works

Several researches were made regarding the manufacturing system modeling methods: Calvo *et al*. proposed an O-O method for the design of automation system, but they only showed the static structure comprised of a class diagram and a use case diagram [4]. Young *et al*. proposed UML modeling of AMS and its transformation into PLC code, but they didn't present the method of PLC code generation [26]. Bruccoleri and Diega presented UML modeling of FMS (Flexible Manufacturing System) and its simulation implementation, but they restricted the control level to the supervisory control level [2]. Bruccoleri compared ladder diagram based-method to O-O modeling for the development of control software [3]. Choi *et al*. proposed a virtual factory simulator framework as a 3D solid-based factory to be used as a line prototyping tool for an AMS [5]. Jalilvand *et al*. proposed hybrid modeling and simulation method for a robotic manufacturing system using timed Petri nets [14].

Among researches about design and validation tools for the PLC control logic, Folch proposed object oriented framework for PLC software development [8]. Spath and Osmers proposed a simulation method integrating plant layout sub-model and control sub-model, and also a PLC code generation from simulation result, but they omitted details of generation procedure [23]. Baresi *et al*. presented the procedure of control logic design using IEC function block diagram (FBD), its transformation into Petri net, and the validation of control logic using SIMULINK simulation system, and C code generation [1]. But, they confined their modeling scope to simple control logic which can be represented by FBD.

Oulidis suggested the use of IEC61499 function block method for the control of automation system [19]. Ekberg and Krogh proposed the method of creating the control software by combining independent predefined control templates [6]. Schreyer and Tseng outlined a framework for reconfiguration design of PLC-based control systems based on axiomatic design theory, but they didn't implement their framework in the shop floor [22]. Authors of this paper proposed object-oriented design, simulation and automatic generation of ladder logic using UML [11, 12].

In the area of automatic ladder logic generation method, there exist mainly three approaches as follows: First approach is Petri net-based ([9], [15], [20], [24]). Second approach is finite state machine-based ([16], [17], [21]). Last approach is flow chart-like-based ([10], [13]).

III. O-O Control Logic Design and Execution Preparation

Virtual prototyping and operational monitoring procedure using the proposed integrated model, which is depicted in Figure 2, is as follows: 1) control model design in parallel with physical layout model design, 2) control rule generation for control logic execution, 3) verification of control logic using the integrated virtual prototype at the design-time, 4) operational monitoring at the run-time. The following sub-section A explains object-oriented control logic design using UML activity diagram. Sub-section B describes the decomposition method of activity diagram for the execution of designed control logic.

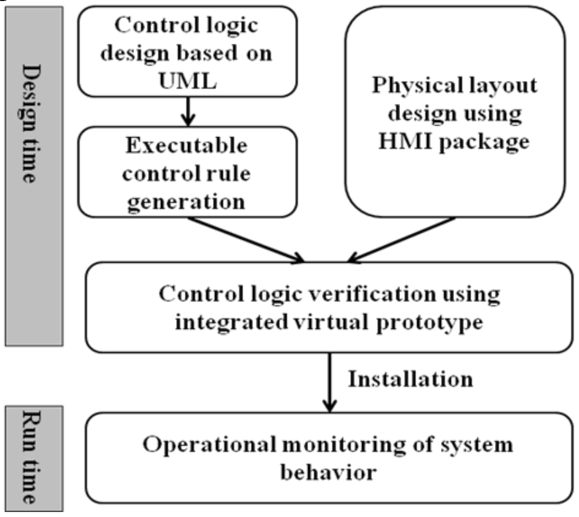

Design time
Control logic design based on UML
Executable control rule generation
Physical layout design using HMI package
Control logic verification using integrated virtual prototype
Installation
Run time
Operational monitoring of system behavior

Figure 2. Procedure of virtual prototyping and operational monitoring

A. O-O Control Logic Design

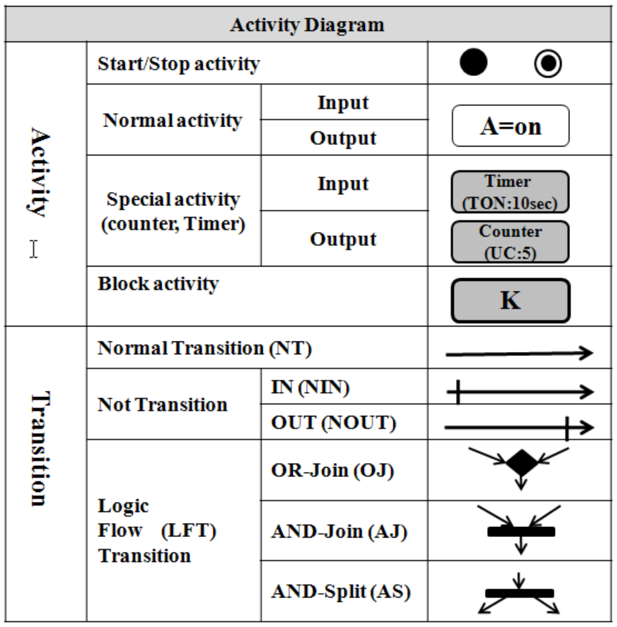
In order to support emerging requirements of manufacturing system design, significant efforts have been made in researches on O-O technologies in manufacturing systems. O-O modeling has been mainly used as a method for the analysis and design of general software system. Recently, it is presented that O-O modeling is also appropriate for real-time system design like an AMS as well as business process modeling. The most typical features of O-O modeling techniques include the interaction of objects, hierarchical composition of objects, and the reuse of objects.

Since the real FA system is operated by the signal sending and receipt among manufacturing equipments such as PLC, sensors, and actuators, it is essential to describe the interactions of FA system components in detail for the robust design of device level control. UML provides the activity diagram (AD), state diagram, sequence diagram, and communication diagram as a modeling tool for the dynamic system behaviors. Among these diagrams, the activity diagram is most suitable for control logic flow modeling because of the following features: 1) it can describe the dynamic behaviors of plant with regard to input/output events in sequential manner. 2) It can easily represent typical control logic flow routing types such as sequential, join, split, and iteration routing.

In order to design control logic written in ladder logic, modification and extension of standard UML elements are required to reflect the specific features of ladder logic. First of all, it should be tested whether activity diagram is suitable for the description of control logic flow, especially for the ladder logic flow. The basic control flow at the ladder logic is sequence, split and join. Especially, three types of split and join control flow must be provided for ladder logic: OR-join, AND-join, AND-split. UML activity diagram can model basic control flows of ladder logic well.

Basically, ladder diagram is a combination of input contact, output coil and AND/OR/NOT logic. Since 'NOT' (normally closed) logic flow in the ladder logic cannot be represented directly in standard activity diagram, new two transition symbols for representing normally closed contact and negated coil are added as normal arcs with left-side vertical bar (called NOT-IN Transition) or right-side vertical bar (called NOT-OUT transition) as depicted in Figure 3. In the extended UML activity diagram, logic and time sequence flow from the top to bottom of diagram.

Elements of extended UML activity diagram is classified into two groups as depicted in Figure 3: activity and transition. Activity group consists of start/stop activity, normal activity and special activity such as timer and counter. For representing control logic, transition group consists of normal transition, not transition and logic flow transition such as OR-join, AND-join and AND-split.



Activity Diagram			
Activity	Start/Stop activity		● ◉
	Normal activity	Input	A=on
		Output	
	Special activity (counter, Timer)	Input	Timer (TON:10sec)
		Output	Counter (UC:5)
	Block activity		K
Transition	Normal Transition (NT)		→
	Not Transition	IN (NIN)	
		OUT (NOUT)	
	Logic Flow (LFT) Transition	OR-Join (OJ)	
		AND-Join (AJ)	
		AND-Split (AS)	

Figure 3. Elements of extended activity diagram

B. Decomposition of UML Activity Diagram for Execution

In order to execute the control logic in the form of an activity diagram, it is needed that an activity diagram is decomposed into several logic units having input/output corresponding to ladder lung since basic ladder lung is a combination of input contact and output coil. This basic executable logic unit is called LU (Logic Unit) which is a 1:1 exchangeable unit to ladder lung. Consequently, a LU can be described in a form of condition-action rule. If certain condition is satisfied, related action is executed. For example, the activity diagram for industrial control logic depicted in Figure 4 can be decomposed into four LUs (LU1 ~ LU4).

The decomposition procedure is as follows: 1) after the creation of an activity diagram for the control logic graphically, store in the form of XML called AD-XML.

2) Decompose an activity diagram into several LUs, and store it in the form of two-dimensional table called LU-Table. LU-table has four columns such as index, input activity, transition, and output activity. Table 1 shows the LU table for the control logic of Figure 4.

3) Determine the logic pattern type for each identified LU. There are four LU pattern types of activity diagram from basic LU type to the concatenation of logic flow transition LU type. Figure 5 shows LU type and its corresponding if-then rule.

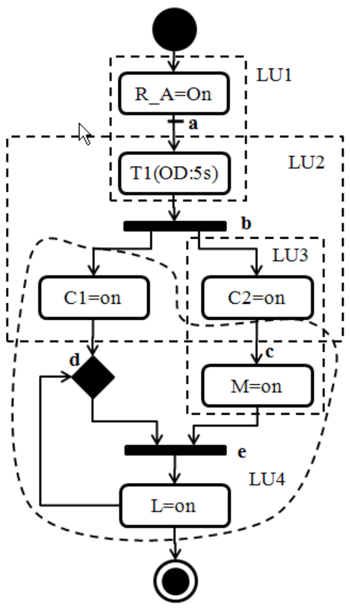
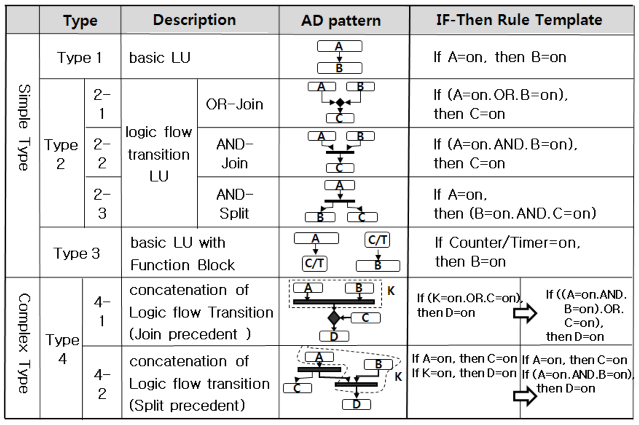


Figure 4. LU decomposition

Table 1. LU table for Figure 4

No.	Input Activity	Transition	Output Activity
LU1	R_A	A:NIN	T1
LU2	T1	b : AS	C1, C2
LU3	C2	c : NT	M
LU4	C1, L, M	d: OJ, e: AJ	L



	Type		Description		AD pattern	IF-Then Rule Template	
Simple Type	Type 1		basic LU			If A=on, then B=on	
	Type 2	2-1	logic flow transition LU	OR-Join		If (A=on.OR.B=on), then C=on	
		2-2		AND-Join		If (A=on.AND.B=on), then C=on	
		2-3		AND-Split		If A=on, then (B=on.AND.C=on)	
	Type 3		basic LU with Function Block			If Counter/Timer=on, then B=on	
Complex Type	Type 4	4-1	concatenation of Logic flow Transition (Join precedent)			If (K=on.OR.C=on), then D=on	If ((A=on.AND. B=on).OR. C=on), then D=on
		4-2	concatenation of Logic flow transition (Split precedent)			If A=on, then C=on If K=on, then D=on	If A=on, then C=on If (A=on.AND.B=on), then D=on

Figure 5. LU type and its corresponding If-Then rule

LU logic pattern type is classified to two types. One is simple type which compose basic if-then rule. The other is complex type that is a combination of simple types. Simple type is further classified to three types according to their corresponding if-then rule template: Type-1 (basic LU), Type-2 (logic flow transition LU: OR-join, AND-join, AND-split), and Type-3 (basic LU with function block). Since complex type is combination of several consecutive logic flow transitions, it has most sophisticated structure among 4 LU types. Complex type is further classified to two types: Type 4-1 (concatenation of logic flow transition: join precedent) and Type 4-2 (split-precedent). Classification criteria is whether 'join' logic flow transition is precedent to other logic flow transitions or 'split' transition is precedent to other elements. Complex LU type needs more operations such as sub-grouping and de-grouping for generating if-then rule.

4) Eventually, LU-Table is transformed to executable rule table. It is generated using LU table and connection information of AD-XML. Rule table has four columns: first column is an index. Second column is LU type. Third column is IF-clause (input condition). And last column is THEN-clause (output). Control logic is executed by rule firing in the rule table sequentially and iteratively. Table 2 shows executable rule table for the control logic depicted in Figure 4. LU4 in the Table 2 is a complex type which needs pre-operation of grouping for the simplification of control logic as follows: 'Group K ':= (C1=on. OR. L=on).

Table 2. Executable rule table for Figure 4

LU	LU type	IF (input activity)	THEN (output activity)
LU1	3	R_A=NOT on	T1(OD:5s)=on
LU 2	2-3	T1=on	C1=on AND C2=on
LU3	1	C2= on	M=on
LU4	4-1	(K=on).AND.M=on	L=on
		⬇	
		(C1=on.OR.L=on). AND. M=on	L=on

IV. Virtual Prototyping and Operational Monitoring

After the control logic design and execution preparation, next step is virtual prototyping by integrating UML control logic model and physical layout model using HMI package. By using virtual prototype, control logic in the design-time is verified using logic execution and material flow animation. After physical installation, this prototype can be reused as a tool for operational monitoring. Sub-section A explains the procedure of control logic verification using virtual prototyping. Sub-section B describes the reuse of design-time virtual prototype as a tool of operational monitoring in the run-time.

A. Control Logic Verification Using Virtual Prototype in the Design Time

Developed software in this study (called DEX-CL) consists of two modules: one is design module in the form of a UML activity diagram. The other is execution module of an activity diagram. The structure of execution part is depicted in Figure 6. Major classes are thread manager, execution manager and IO manager. The control logic of each required function in the automation system is implemented as a runtime thread instance. Thread manager inherited from super manager manages runtime thread instances. Execution manager consists of decomposition manager and runtime thread. For the control logic execution, decomposition manager decomposes activity diagram into several LUs and generates rule table. IO manager which is inherited from communication manager is responsible for communicating with external software.

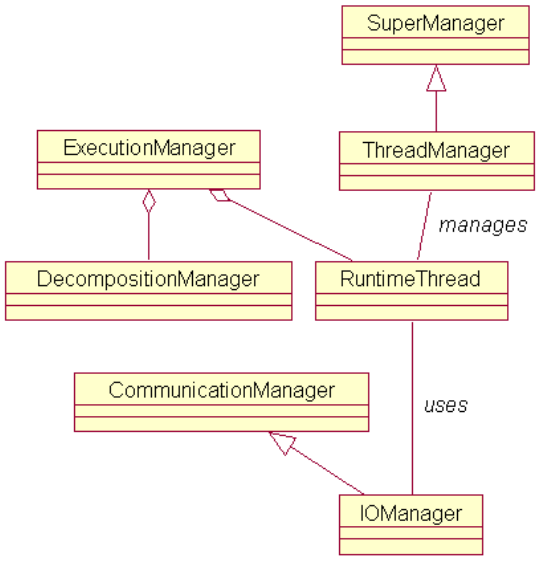


Figure 6. Structure of DXE-CL executor module

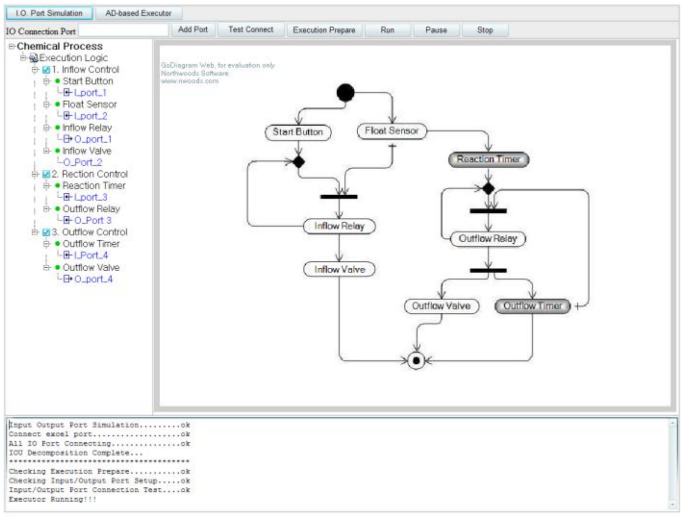
Proposed virtual prototyping procedure is explained using illustrative applications in this section. First example application is a kind of fluid storage tank for chemical reaction. This system provides three functionalities: fluid supply, chemical reaction and drainage.

Detailed procedure is explained as follows:

1) From the system specification, control logic is designed by using DEX-CL. Figure 7 shows control logic for supply, chemical reaction and drain of fluid by using DEX-CL module in a form of UML activity diagram.

2) In parallel with control logic design, physical layout is constructed by using commercial HMI or VMS software. Figure 8 shows layout model using Wonderware's InTouch HMI package [25].

3) Excel-based interface is established between control model and layout model. Input/output port type is classified to input (I), output (O) and internal memory (M). The port value is 0 or 1. The left part of Figure 7 shows mapping between sensor/actuator and IO port. Figure 9 shows interface structure of virtual prototype. 4) After designing control logic, he or she generates rule table for the execution of control logic using DEX-CL execution preparation function. Figure 10 shows LU decomposition. Control logic of example system is decomposed into 5 LUs. Table 3 shows generated rule table for the control logic of Figure 7. 5) Finally, he or she runs the executor of control logic. After that, various stakeholders of automation system verify the control logic by simultaneously investigating animated plant model and running control model. Control logic execution is conducted by rule firings in the rule table with continuous loop. Each required function described in the form of an activity diagram is implemented as an independent thread object, and each thread performs its operation by interacting with device element of a plant model independently or sequentially depending on whether there is a temporal relationship between functions or not. Before starting loop, the status of input port within the excel sheet is read. During loop, each row of rule table is checked whether input condition is satisfied. If satisfied, the status of output port is updated to Excel sheet.



```
Input Output Port Simulation.........ok
Connect excel port...................ok
All IO Port Connecting...............ok
IOU Decomposition Complete...
*************************************
Checking Execution Prepare...........ok
Checking Input/Output Port Setup.....ok
Input/Output Port Connection Test....ok
Executor Running!!!
```

Figure 7. Control logic design of 1st example by DXE-CL

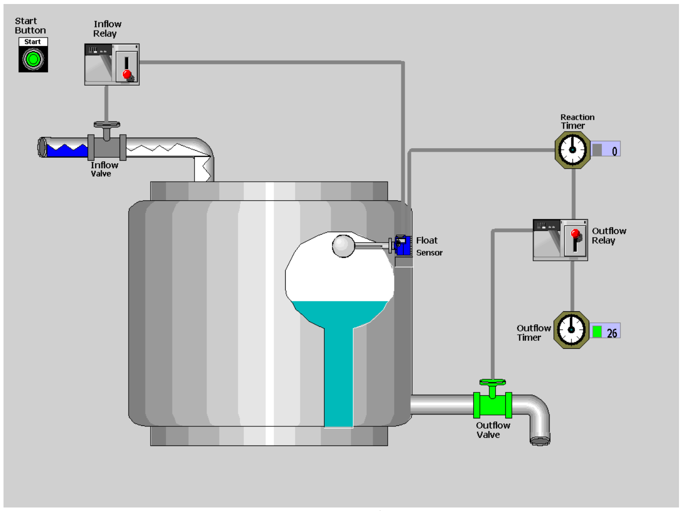


Figure 8. Plant layout model of 1st example by HMI package

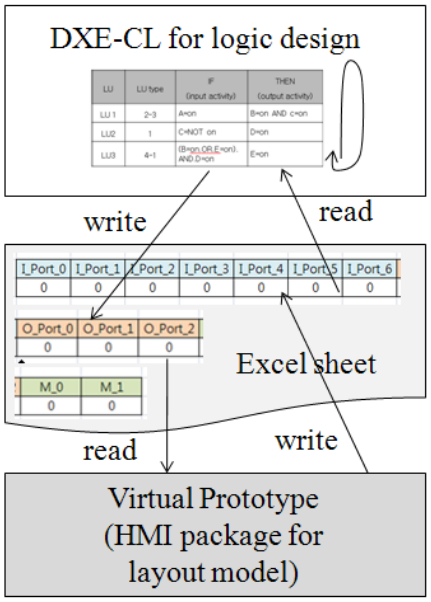
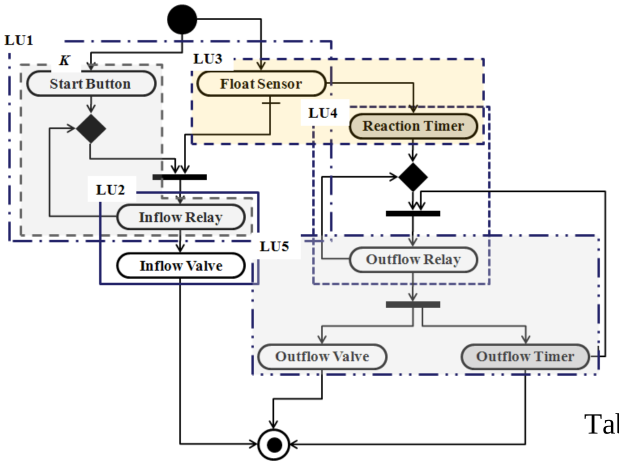

DXE-CL for logic design
LU | LU type | IF (input activity) | THEN (output activity)
LU1 | 2-3 | A=on | B=on AND c=on
LU2 | 1 | C=NOT on | D=on
LU3 | 4-1 | (B=on OR E=on). AND D=on | E=on
write
read
I_Port_0 I_Port_1 I_Port_2 I_Port_3 I_Port_4 I_Port_5 I_Port_6
O_Port_0 O_Port_1 O_Port_2
M_0 M_1
Excel sheet
read
write
Virtual Prototype (HMI package for layout model)

Figure 9. Structure of virtual prototype



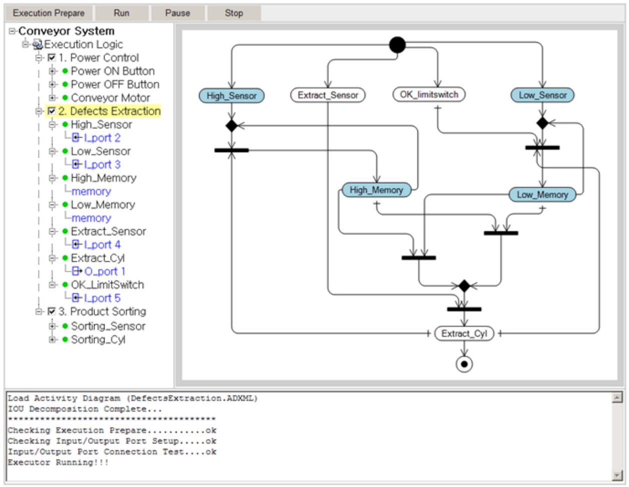
Table

Figure 10. LU decomposition for 1st example system

Table 3. Executable rule table for Figure 7

LU	LU Type	IF (input activity)	THEN (output activity)
LU1	4-1	(*K* = on) AND (Float Sensor = NOT on)	(Inflow Relay = on)
		↓	
		{(Start Button = on) OR (Inflow Relay = on)} AND (Float Sensor = NOT on)	
LU2	1	(Inflow Relay = on)	(Inflow Valve = on)
LU3	1	(Float Sensor = on)	(Reaction Timer = on)
LU4	2-1	(Reaction Timer = on) OR (Outflow Relay = on)	(Outflow Relay = on)
LU5	2-3	(Outflow Relay = on)	(Outflow Valve = on) AND (Outflow Timer = on)

Second example application prototype is a kind of conveyor-based material handling system which identifies and removes defective parts, and sorts good parts according to their material properties. This material handling system provides three functionalities: power control function, defective parts identification and removal function, and part sorting function. Figure 11 shows DEX-CL screen displaying the logic of defective parts identification and removal function. Physical layout model of example application using InTouch is shown in Figure 12. Table 4 shows generated rule table for the control logic of Figure 11.



```
Load Activity Diagram (DefectsExtraction.ADXML)
IOU Decomposition Complete...
**************************************
Checking Execution Prepare...........ok
Checking Input/Output Port Setup.....ok
Input/Output Port Connection Test....ok
Executor Running!!!
```

Figure 11. Control logic design of 2nd example using DEX-CL

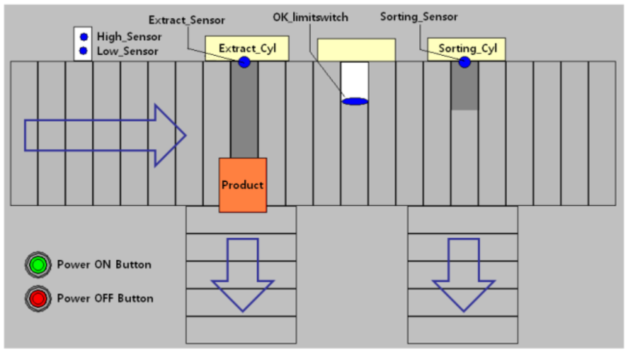


Figure 12. Plant layout model of 2nd example using HMI package

Table 4. Executable rule table for Figure 11

LU	LU type	IF (input activity)	THEN (output activity)
LU1	4-1	(High Sensor=on OR High Memory=set).AND. (Extract Cyl = NOT on)	High Memory
LU2	4-1	(Low Sensor=on OR Low Memory=set).AND. (OK Limit Switch=NOT on) AND (Extract Cyl=NOT on)	Low Memory
LU3	4-1	{(High Memory=set AND Low Memory=set).OR. (High Memory=NOT set AND Low Memory=NOT set)} AND (Extract Sensor=ON)	Extract Cyl

B. Operational Monitoring in the Run Time

After verifying the control logic using virtual prototype, real manufacturing system is implemented and its operation is started. During operating stage, remote monitoring of system behavior is necessary. In this case, HMI layout model of virtual prototype in the design-time can be reused with slight modifications.

The right part of Figure 13 shows schematics of operational monitoring at run time using virtual prototype constructed in design time. Control model of virtual prototype is substituted to PLC, and layout model of virtual prototype is substituted to real system. Figure 14 shows the monitoring display of 1st example application. In addition to the monitoring of elapsed time for chemical reaction and drainage of example system as shown in the right part of Figure 14, two additional monitored objects are included: 1) planned daily production versus actual daily production amount (left upper part of Figure 14), 2) temperature of inner tank (on the storage tank of Figure 14).

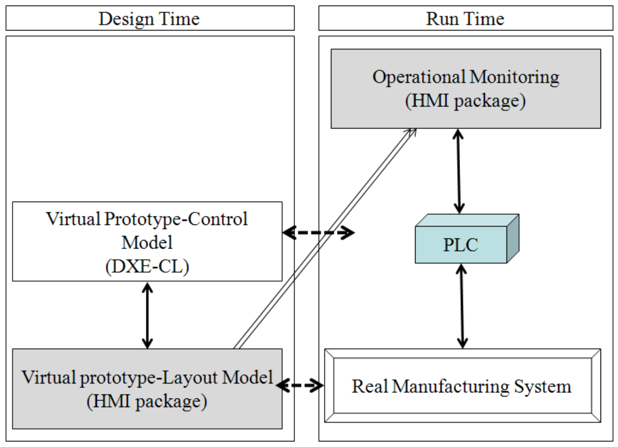

Design Time
Run Time
Operational Monitoring (HMI package)
Virtual Prototype-Control Model (DXE-CL)
PLC
Virtual prototype-Layout Model (HMI package)
Real Manufacturing System

Figure 13. Operational monitoring using virtual prototype

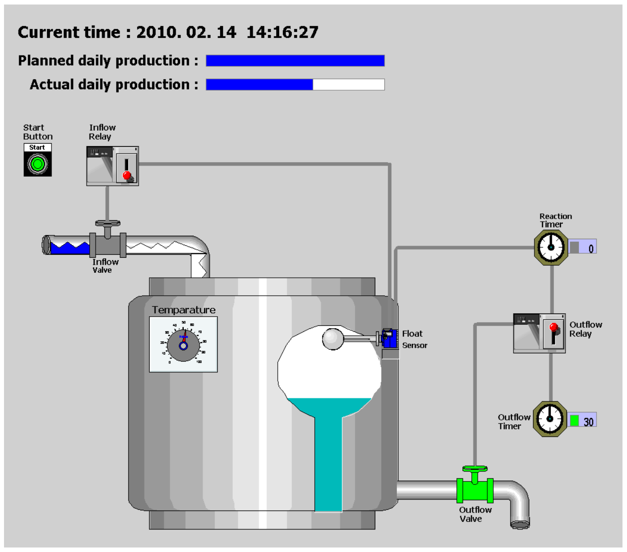


Figure 14. Monitoring display in the run-time

V. Conclusion

To meet emerging requirements of reconfigurable control system, it is quintessential to design and verify industrial control logic rapidly and easily during the life cycle of manufacturing system. And it is also necessary that the model for verification in design time is reused for operational monitoring in run time.

However, existing verification methods such as discrete event simulation couldn't provide the functionalities to fulfil the new requirements such as rapid verification of control logic and simultaneous processing of mechanical and control design task.

Proposed integrated model for virtual prototyping and operational monitoring of industrial control logic is to improve the limitation of current control logic verification and model reuse.

By proposed method, control logic can be easily modifiable to accommodate changes in manufacturing plant configuration during the control system life cycle. It also facilitates the generation of control logic easily within a short time without considering complicated control behavior based on verification result. In addition, this method serves as an operational monitoring tool with slight modifications in the run time.

As a further research, the integration method between coordination control at the cell level and PLC-based procedural control at the device level is necessary to develop a unified tool for the design, verification, and monitoring of reconfigurable control system.

Acknowledgment

This research was financially supported by Ministry of Education, Science and Technology (MEST) and National Research Foundation of Korea (NRF) through the Human Resource Training Project for Regional Innovation. It is also conducted as a part of project for the development of digital based real-time adaptive manufacturing system platform sponsored by Ministry of Knowledge Economy (MKE) and VMS Solutions co., ltd.

References

[1] Baresi L, Mauri M, Monti A, Pezze M (2000) PLCTools: design, formal validation, and code generation for programmable controllers. Proc of 2000 IEEE Conference on Systems, Man and Cybernetics, Nashville, USA

[2] Bruccoleri M., and Diega S. N., An object-oriented approach for flexible manufacturing control systems analysis and design using the unified modeling language, *International Journal of Flexible Manufacturing System*, Vol.15, No.3, 2003, pp.195-216.

[3] Bruccoleri M., Reconfigurable control of robotized manufacturing cells, *Robotics and Computer-Integrated Manufacturing*, Vol.23, 2007, pp.94-106.

[4] Calvo I., Marcos M., Orive D., and Sarachaga I., Using Object-Oriented Technologies in Factory Automation, *Proceedings of 2002 IECON Conference*, Sevilla, Spain, 2002, pp.2892-2897.

[5] Choi B., Park B., and Ryu H. Y., Virtual factory simulator framework for line prototyping, *Journal of Advanced Manufacturing System*, Vol.3, No.1, 2004, pp.5-20.

[6] Ekberg G. and Krogh H. K., Programming discrete control systems using state machine templates, *Proceedings of the 8th international workshop on discrete event systems*, 2006, pp.194-200, Ann Arbor, USA.

[7] Epureanu, A., Marin, F.B., Marinescu, V., Banu, M., Constantin, I, Reconfigurable machine tool programming - A new approach, *WSEAS Transactions on Systems and Control*, Vol. 3, No. 5, 2008, pp. 463-472.

[8] Folch, J.R., Pérez Cruz, J., Pineda Sánchez, M., Puche Panadero, R., Aragó Ramos, E., Object oriented framework for PLC software development, *WSEAS Transactions on Systems*, Vol. 4, No. 9, 2005, pp. 1522-1529.

[9] Frey G., and Minas M., Internet-based development of logic controllers using signal interpreted Petri nets and IEC 61131, *Proceedings of the SCI 2001*, Vol.3, 2001, pp. 297-302, Orlando, FL, USA

[10] Hajarnavis V., and Young K., A comparison of sequential function charts and object modeling with PLC programming, *Proceedings of American Control Conference*, 2005, 2034-2039.

[11] Han K. H. and Park J.W., Object-oriented ladder logic development framework based on the unified modeling language," *Studies in Computational Intelligence*, Vol. 208, 2009, pp.33-45.

[12] Han K. H., Park J. W. and Choi Y., Object-oriented modeling and simulation for the verification of industrial control logic, *Proceedings of*

the 37th International Conference on Computers and Industrial Engineering, Alexandria, Egypt, 2007, pp. 2377-2384.

[13] Jack H., Automating manufacturing systems with PLCs, http://clay-more.engineer. gvsu.edu/ ~jackh/books.html, 2007.

[14] Jalilvand, A., Khanmohammadi, S., Nia, F.S., Hybrid modeling and simulation of a robotic manufacturing system using timed Petri nets**,** *WSEAS Transactions on Systems*, Vol. 4, No. 5, 2005, pp. 513-520 **.**

[15] Lee G. B., Zandong H., and Lee J. S., Automatic generation of ladder diagram with control Petri net, *Journal of Intelligent Manufacturing*, Vol. 15, No. 2, 2004, pp.245-252.

[16] Liu J., and Darabi H., Ladder logic implementation of Ramadge-Wonham supervisory controller, *Proceedings of Sixth International Workshop on Discrete Event Systems*, 2002, pp.383-389.

[17] Manesis S., and Akantziotis K., Automated synthesis of ladder automation circuits based on state diagrams, *Advances in Engineering Software*, Vol. 36, No. 4, 2005, pp. 225-233.

[18] Mehrabi, M. G., Ulsoy, A. G. and Koren, Y., Reconfigurable manufacturing systems**:** Key to future manufacturing**,** *Journal of Intelligent Manufacturing*, Vol.11, No. 4, 2000, pp. 403-419.

[19] Oulidis, K.T., The function block model in embedded control and automation from IEC61131 to IEC61499, *WSEAS Transactions on Computers*, Vol. 8, No. 9, 2009, pp. 1597-1609.

[20] Peng S. S., and Zhou M.C., Ladder diagram and Petri net based discrete event control design methods, *IEEE Transactions on Systems, Man and Cybernetics- Part C*, Vol. 34, No.4, 2004, pp.523-531.

[21] Sacha K., Automatic code generation for PLC controllers, *LNCS*, Vol. 3688, 2005, pp.303-316.

[22] Schreyer M. and Tseng M. M., Design framework of PLC-based control for reconfigurable manufacturing systems, *Proceedings of international conference on flexible automation and intelligent manufacturing (FAIM 2000)*, Vol.1, 2000, pp.33-42.

[23] Spath D., and Osmers U., Virtual reality- an approach to improve the generation of fault free software for programmable logic controllers, *Proceedings of IEEE International Conference on Engineering of Complex Computer Systems, Montreal*, Canada, 1996, pp.43-46.

[24] Taholakian A., and Hales W. M. M., PN <-> PLC: a methodology for designing, simulating and coding PLC based control systems using Petri nets, *International Journal of Production Research*, Vol. 35, No. 6, 1997, pp.1743-1762.

[25] Wonderware, http://global.wonderware.com /EN/ Pages/ WonderwareInTouchHMI.aspx, 2009.

[26] Young K. W., Piggin R., and Rachitrangsan P., “An Object-Oriented Approach to an Agile Manufacturing Control System Design,” *International Journal of Advanced Manufacturing Technology*, Vol.17, No.11, 2001, pp.850-859.