Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

ARCHITECTURE FOR MODELING, SIMULATION, AND EXECUTION OF PLC BASED MANUFACTURING SYSTEM

Devinder Thapa
C. M. Park

Dept. of Industrial Engineering
Ajou University
Suwon, 446-749, SOUTH KOREA

Kwan Hee Han

Dept. of Industrial Sys. Engineering
Gyeongsang National University
Jinju, 660-901, SOUTH KOREA

S. C. Park
Gi-Nam Wang

Dept. of Industrial Engineering
Ajou University
Suwon, 446-749, SOUTH KOREA

ABSTRACT

In this paper, we propose an integrated architecture for modeling, simulation, and execution of PLC (Programmable Logic Controller) based manufacturing system. The main objective is to integrate the high level modeling, simulation, and device level executable code generation. This architecture can improve the fidelity between high level system model and lower level PLC controlled devices. In this paper, we model the shop floor controller system using DEVS (Discrete Event System Specification) formalism, subsequently, simulate the model and generate SOP (sequence of operations). We added two algorithms in conventional DEVS, the first algorithm makes an interface between 3D graphic model and DEVS model, whereas, the second algorithm generates SOP. As a result, the generated SOP can be mapped with PLC I/O (Input/Output) address to generate an executable controller code. For the purpose of further validation and implementation, the generated program can be downloaded to software or hardware PLC.

1 INTRODUCTION

The robustness of PLC devices has made them the *de facto* choice in an automobile industry as a process controller machine. A PLC device has multiple I/O functions and logic operation functions, and plays a central role in cell control (Jang, Koo, and Nof 1997). Similarly, PLC programs are the software counterpart of this machine. The design and development of this program for manufacturing systems is a very complex process. Although it is easy to design logic programs for small systems, as the length of the program grows it becomes accordingly more complicated to design and verify the PLC code. On the contrary, the functional operation of any manufacturing unit should be smooth enough to run the operation without any work stoppage. A delay in generating the controlling code of processing devices may cause a major loss of revenue and goodwill. Use of rapid modeling and compiling tools to generate IEC standard PLC program is an efficient method to reduce the PLC development time.

Since development of PLC program using traditional approach is error-prone and time consuming job (Canet et al. 2000; Thapa et al. 2006). A small change in process controller may drastically increase the rate of down-time and ramp-up time. Down time and ramp-up time can cause the production delays; thus, much effort has been given to follow some standard designing and verification techniques. On of the efficient and effective ways to reduce down-time and ramp-up time is formalization of informal specification and converting it into IEC standard PLC code. However, the validity of model itself should be tested before generation of the execution code. Conventional techniques like simulation and visual animation can be used to verify the controller logic; however, the integrated use of formal methods for verification is more desirable to validate the state model for hidden and subtle errors. An integrated framework for modeling, simulation, and code generation is proposed to complete the research work.

In our research, a user oriented simple methodology to model and execute a PLC controlled design with DEVS is proposed. In this framework, we model the shop floor controller system using DEVS formalism (Zeigler, Praehofer, and Kim 2000).

The DEVS is an event based system that is based on modular and hierarchical architecture. A modular architecture is characterized by the description of complex systems by separately defining an individual controller for each piece of devices. It reduces the complexity of modeling a distributed controller system. The DEVS model is The DEVS model is consist of set of coupled and atomic model. An Atomic DEVS model (A1, and A2), as shown in figure 1, is defined as a seven tuples, $M = < X,Y,S,\tau,\delta_{ext},\delta_{int},\lambda >$ where :

- X is the set of input events
- Y is the set of output events
- S is the set of sequential states, {S1,S2,S3, and S4}

- $\tau : S \to R_{[0,\infty]}$ is the time advance function which is used to determine the lifespan of a state where $R_{[0,\infty]}$ is the set of non-negative real numbers plus infinity
- $\delta_{ext} : Q\,X\,S \to S$ is the external transition function where $Q = \{(s, t_e) \mid s \in S, 0 \le t_e \le \tau(s)\}$ and t_e is the elapsed time since the last event. $\delta_{ext} : Q\,X\,S \to S$ specifies how the system changes state when an input is received, the effect is to place the system in a new thus scheduling it for a next internal transition; the next state is computer on the basis of the present state, the input port and value of the external event, and the time that has elapsed in the current state.
- $\delta_{\text{int}} : S \to S$ is the internal state transition function which defines how a state of the system changes internally (when the elapsed time reaches to the lifetime of the state) $\lambda(s)$.
- $\lambda : S \to Y$ is the output function which defines how a state of the system generates an output event (when elapsed time reaches to the lifetime of the state)

There are two cases that an atomic DEVS model *M* can change its state $s \in S$:(1) when an external input $x \in X$ comes into the system *M*; (2) when the elapsed time t_e reaches the lifespan of *s* which is defined by $\tau(s)$. At the same time of (2), *M* generates an output which is defined by $\lambda(s)$.

Whereas, the coupled DEVS model (C1), as shown in figure 1, defines which sub-components (A1, and A2) belong to it and how they are connected with each other. A coupled DEVS model is defined as a eight tuples $N = < X, Y, D, \{M_i\}, C_{xx}, C_{yx}, C_{yy}, Select >$ where

- *X* is the set of input events
- *Y* is the set of output events
- *D* is the name set of sub-components, {A1, and A2}
- *{Mi}* is the set of sub-components where for each , $i \in D$, M_i can be either an atomic DEVS model or a coupled DEVS model
- $C_{xx} \subseteq X \times \bigcup_{i \in D} X_i$ is the set of external input couplings
- $C_{yx} \subseteq \bigcup_{i \in D} Y_i \times \bigcup_{i \in D} X_i$ is the set of internal couplings
- $C_{yy} \subseteq X \times \bigcup_{i \in D} X_i$ is the set of external ouput couplings
- $select : 2^D \to D$ is the tie-breaking function which defines how to select the event from the set of simultaneous events;

Like the behavior of the atomic DEVS model, a coupled DEVS model N changes its components' states (1) when an external event $x \in X$ comes into N; (2) when one of components M_i where $i \in D$ executes its internal state transition and generates its output $y_i \in Y_i$. In both cases (1) and (2), a triggering event is transmitted to all influences which are defined by coupling sets C_{xx}, C_{yx}, and C_{yy}.

Subsequently, we simulate the DEVS model and generate SOP. As a result, the generated SOP can be mapped with PLC I/O (Input/Output) address to generate an executable PLC program. Finally, these PLC program can be downloaded to the software or hardware PLC for the purpose of further validation and implementation.

The rest of paper is organized as follows. Section 2 discusses reference works and Section 3 explains the proposed model based architecture, and two algorithms added to conventional DEVS simulator. Similarly, Section 4 illustrates implementation methodology with a simple example of automobile work cell. And finally, Section 5 concludes the paper with brief notes on future works.

2 REFERENCE WORKS

Although lots of published work can be found about DEVS modeling and simulation, not much literature are related to modeling of PLC controlled manufacturing system. Some of the similar reference works have been briefly described as follows.

Hwang and Choi (2001) presents a new formalism GK-DEVS for simulation of 3-D multi-components systems. This research work advocates constructing a unified virtual manufacturing in which human-in-the-loop simulation can be supported and the simulation model can work as a control model without model modification or additional modeling. In this reference work, the atomic model and coupled model has been integrated to a single structure. Even though it is easy to connect with less coupling, however, direct coupling scheme is less modular and it makes the hierarchical message passing more difficult.

Park (2005) presents a proposed virtual FMS model, which consists of four types of objects: the virtual device model (object model), the transfer handler model (functional model), the state manager model, and the flow controller model (dynamic model). A virtual device model consists of two parts: shell and core. To improve the reusability of a virtual device model, the shell part is designed to adapt to different FMS configurations. For the fidelity of the virtual FMS model, a transfer handler model has a set of device-level commands imitating the physical mechanism of a transfer.

Wu (2005) presents a methodology of formal-DES-theory-based supervisors; it can be applied to industrial work cell. The software has been created in a modular fashion. This allows performing changes to specific modules without the need to modify the others. However, PLC Automata is the core formal modeling deployed to develop hybrid PLC. This work focused on generating PLC code using formal model, however it does not describe about virtual factory and reducing fidelity.

Most of the previous work is suitable for small systems but comprehensive results for real system have yet to come (Lucas and Tilbury 2003). Some commercial products such as EM-PLC and DELMIA V5 are trying to implement the integrated architecture. The EM-PLC can auto-generate the SOP, which is based on 3-D modeling information, and bar chart (Lee et al. 2006). However, the drawbacks of these techniques are heavy resource requirement such as CPU time and memory space. Similarly, these techniques are not effective to predict the possible errors like propagation of errors from one work cell to another. To overcome this problem we define a user oriented simple methodology to generate IEC standard PLC code. A theoretical foundation to develop an auto-generation tool is defined in this research work, and will be subsequently implemented in a real automotive industry to validate the efficiency and effectiveness of the generated PLC code.

The main contribution of the proposed framework is to integrate the factory modeling, simulation, and PLC code generation. By integrating, we can improve the fidelity between high level factory model and lower level PLC controlled devices. In addition, generation of PLC program based on different scenario will be helpful to generate the compact and optimized code. In the existing systems, it is very difficult to generate compact PLC code because it requires solving number of intermediate phases to generate valid input or output signals. Furthermore, simulation based PLC program generation and execution can overcome the state explosion problem (Clarke, Grunberg, and Peled 1999), which exists in other formal verification techniques.

3 PROPOSED MODEL-BASED ARCHITECTURE

The framework described in this paper, employs the DEVS formalism to model a PLC based controller system, as shown in figure 1. The DEVS is an event based system that is based on modular and hierarchical architecture (Zeigler, Praehofer, Kim. 2000). A modular architecture is characterized by the description of complex systems by separately defining an individual controller for each piece of devices. It reduces the complexity of modeling a distributed controller system. After forming the DEVS model, next step is to simulate it. We added two algorithms to conventional DEVS simulator, which is described in section 3.2 and section 3.3. The objective of this algorithm is to make an interface between 3D graphics model and DEVS Model. Therefore, we can simulate both (3D graphics and DEVS) models, simultaneously, and generates SOP. The generated SOP is mapped with PLC's I/O address to create a symbol table. A symbol table stores the signal name and PLC I/O address as shown in figure 1. Therefore, we can manipulate symbol table to generate an executable PLC program. Since verification of the generated executable PLC program is a mandatory job, before executing in the real environment, thus we can use software or hardware test to validate the generated code.

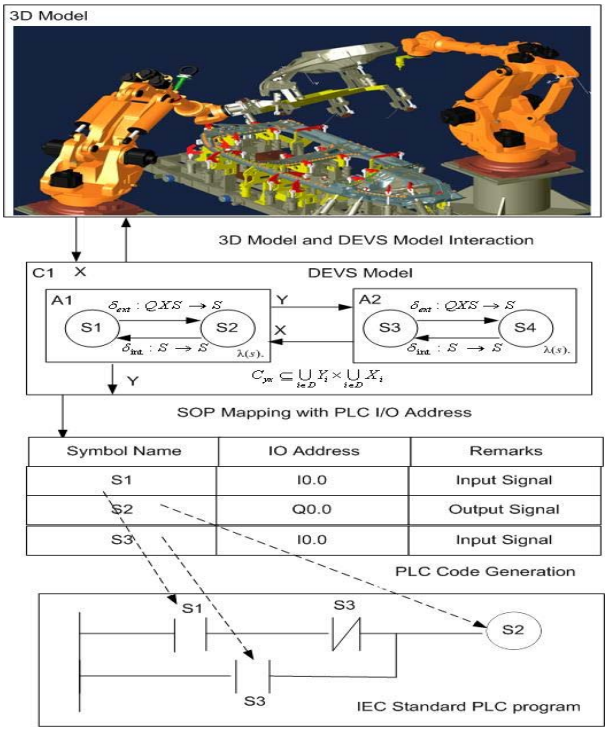


Figure 1: Model-based design for plc controlled manufacturing system

3.1 Algorithm to Integrate 3D Graphics Model and DEVS Model

According to the given algorithm, initially, the DEVS simulator's *when_init()* function detects the 3D graphics model. Then, it retrieves the device name, motion type, and motion time, for instance ROBOT1, MOVE_UP, and 15 second respectively, from the graphics model device database. Accordingly, it stores related information into corresponding variables, as shown in the following *pseudo code*.

```
when_init()
while(!3DGraphics)
var_device_name:=3D_device_name
var_motion_type:=3D_motion_type
var_motion_time:=3D_motion_time
```

After that, it fetches the DEVS atomic model name, event, and time information from DEVS model. The retrieved values of 3D graphics model and DEVS model are compared, subsequently. If the simulator finds the matching values then it executes the corresponding motion in the 3D graphics. This process continues until the defined motion time of 3D graphics model and DEVS atomic model times equalize. After the successful completion of the existing motion, DEVS simulator fetches the next sequence of operation and carries the same process till the end of the simulation; as described in the following pseudo code.

```
when_receive_*(portvalue, time)
if(portvalue=internal)
if(store_values!='NULL')
  if (var_device_name=DEVS_atomic_model_name)
&&
    (var_motion_type=DEVS_atomic_model_event)
  then
   while
  (var_motion_time!=DEVS_atomic_model_time)
  run simulation();
  end
  end
  else
  'wait'
  end
  when_receive_X(portvalue, time)
  if(portvalue=external)
  if(store_values!='NULL')
  f  (var_device_name=DEVS_atomic_model_name)
&& (var_motion_type=DEVS_atomic_model_event)
  then
  while
  (var_motion_time!=DEVS_atomic_model_time)
  run simulation();
  end
  end
  else
  'wait'
  end
```

The changes take place inside the *when_init(), when_receive_*(),* and *when_receive_x(),* functions of *atomic.h* file of the DEVS simulator. The execution of specific function depends on the variable *portvalue* (internal or external).

3.2 Algorithm for SOP Mapping with PLC IO Addresses and Code Generation

In order to achieve more conformity of different PLC programming languages, IEC 61131-3 standard was developed. The IEC 61131-3 has classified five types of PLC program, IL(Instruction List), ST(Structured Text), LD(Ladder Diagram), FBD(Function Block Diagram), and SFC (Sequential Flow Chart) (The International Electro Technical Committee 1993). Where IL and ST are textual programming whereas LD and FBD are graphical program, and SFC is a sequential function chart. The SFC can be used to define the sequence of program that consists of other programs. Based on the algorithm define in the subsequent section, we can convert DEVS simulated data to any of the five IEC standard languages.

Algorithm used for symbol mapping and code generation is described as follows. Initially, the DEVS simulator read the SOP information and maps it to PLC IO address. All the input variables passed to DEVS atomic model can be mapped as INPUT event, whereas, all the output generated by DEVS atomic model can be mapped as OUTPUT event in the symbol table. Subsequently, if the type of event is INPUT then the simulator mapped the variable name with input address (Ix.x) of the PLC. Similarly, if the event type is OUTPUT, then the simulator maps variable name with output address (Qx.x) of the PLC. Where, I and Q denote address type INPUT or OUPUT, and x.x denotes particular byte and bit addresses of the PLC device.

```
Read (SOP)
While (SOP!=NULL)
Check (event)
if (event=INPUT)
DEVS_atomic_model_name<MAP>
INPUT_Address(I_x.x)
end
if (event=OUTPUT)
DEVS_atomic_model_name<MAP>OUTPUT_Addre
ss(Q_x.x)
end; end
```

In terms of PLC code generation, the mapped values can be written in the variable section of the IEC's POU (Program Organization Units) as shown in example 1. The pre-conditions defined in the DEVS atomic model to generate an output event can be implemented as interlocks using IF-ELSE constructs in the IEC's POU as shown in example 2. The pre-conditions are logical expressions formed out of the input variables. For instance, if the values of C1, MH1, and MH2 are one then start the ZIG operation. Similarly to start the WM1 and WM2 operation the value of ZIG must be one. Consequently, by combining the INPUT, OUTPUT, and interlocks we can generate a complete execution code. A simple example of IEC standard PLC program (ST) is shown in figure 7.

Example 1: Write mapped values of symbol table to IEC's POU;

```
INPUT_VAR
C1 : BOOL;
```

```
MH1 : BOOL;
MH2 : BOOL;
END_VAR
OUTPUT_VAR
WM1 : BOOL;
WM2 : BOOL;
ZIG : BOOL;
END_VAR
```

Example 2: Map pre condition to PLC program interlock or condition

```
IF  (C1=1 AND MH1=1 AND MH2=1) THEN
ZIG := 1;
END_IF
IF (ZIG=1) THEN
WM1 :=1;
WM2 :=1;
END_IF
```

The SOP mapping and code generation algorithm is added in the *SRTEngine.h* file of DEVS simulator.

4 AN EXAMPLE TO IMPLEMENT THE PROPOSED FRAMEWORK

4.1 Work Cell Layout of Automobile Assembly Line

To clarify the procedure, we are explaining a work cell layout of assembly line as depicted in figure 2. This layout is divided into three parts for the sake of clarity. Where, the first section known as part handling cell is composed of one carrier and two material handling robots (robot1_MH, and robot2_MH). The function of carrier is to carry the central part of the car floor in the assembly line. We assumed that the part is already loaded in the carrier, whereas the function of robot1_MH and robot2_MH are to bring front part and rear part in the assembly line, respectively. The three numbers defined inside the section one are the sequence of part handling by the corresponding devices.

Likewise, section two known as part assembling cell has two welding robots (robot1_WLD and robot2_WLD). The functions of these two welding robots are to perform spot welding to fix the three components. After the spot welding is finished, assembled part is moved to the section 3 that is known as assembled part cell. In this way, the one cycle of the assembly process finishes.

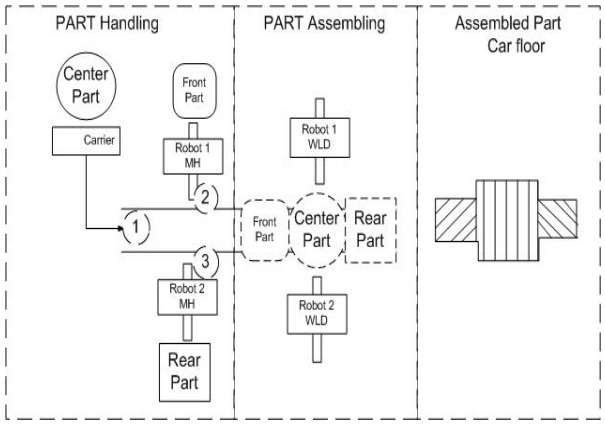
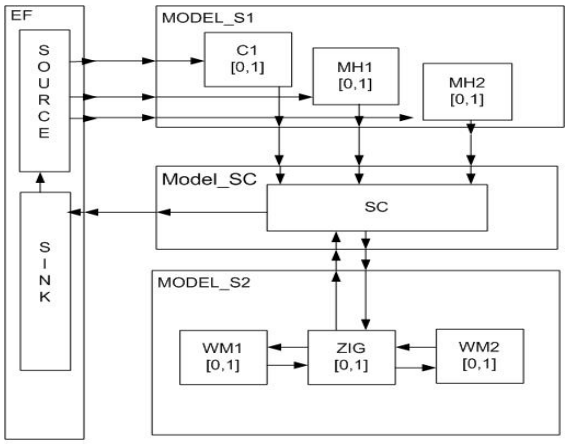


Figure 2: Work cell layout of automobile assembly line for car floor

4.2 Corresponding DEVS Model of the Work Cell

Figure 3 is the corresponding DEVS model and its behavior of work cell layout as described in figure 2. We can develop four coupled model, which are EF, Model_S1, Model_S2, and Model_SC. For instance, EF is an experimental framework which consists of two atomic models called source and sink. Source atomic model is used to generate a part information, whereas, sink atomic model can be used to collect statistical reports. Likewise, Model_S1 is a corresponding model for part handling cell. It consists of three atomic models, that are carrier (C1), robot1_MH (MH1), and robot2_MH (MH2). The binary number [0, 1] simply described the state of the atomic model. Where, 0 is IDLE state of the machine and 1 means BUSY state of the machine.

Similarly, Model_S2 is the part assembling section of the work cell layout as shown in figure 4. It consists of three atomic models, two welding robots (WM1, and WM2) and one ZIG. To maintain the hierarchical and modular structure, we define an extra supervisory model, Model_SC, which consists of one atomic model SC. It can be used as a supervisory controller to keep track of different atomic models. It checks pre-conditions of the system states before firing any event. Similarly, an arrow connecting models shows the flow of information from one module to another. This type of modular and hierarchical architecture is compatible with IEC 61131-3 standard program structures. As a result, it facilitates mapping from DEVS model to IEC standard POUs.



Firing Condition

If (C1==1 & MH1==1&MH2==1)
ZIG=1
Endif

If (C1==0 & MH1==0 & MH2==0 & ZIG==1)
WM1=1
WM2=1
Endif

Figure 3: Corresponding DEVS model of work cell

4.3 Implementation of the Proposed Framework

$S = \{WAIT, FIRE, ZIG, WELD\};$

$X = \{CR, MH1, MH2\};$

$Y = \{ZIG, ZIG\_DONE, WM1, WM2, WM1\_DONE, WM2\_DONE\};$

$\tau_{(WAIT)} = \infty;$

$\tau_{(FIRE)} = INTEGER^{+};$

$\tau_{(ZIG)} = INTEGER^{+};$

$\tau_{(WELD)} = INTEGER^{+};$

$s_0 = \{WAIT\}$

$\delta_{ext}(S, \tau s, \tau e, x) = \delta_{ext}(WAIT, \tau(WAIT), [0, \tau(WAIT)], \{?CR \| ?MH1 \| ?MH2\}) = (FIRE, 1);$

$\delta_{int}(S, [0, \tau s], y) = \delta_{int}(FIRE, \tau(FIRE),$

$if\ ((CR \& MH1 \& MH2) = true)$

$then$

$Y = !ZIG;$

$(ZIG, 1);$

$else$

$Y = !BACK;$

$(WAIT, 1);$

$\delta_{int}(ZIG, \tau(ZIG), \{!ZIG\_DONE, !WM1, !WM2\}) = (WELD, 1);$

$\delta_{int}(WELD, \tau(WELD), \{!WM1\_DONE, !WM2\_DONE\}) = (WAIT, 1);$

Figure 4: Mathematical representation of DEVS simulator

This section describes a mathematical representation of DEVS simulator for the atomic model of supervisory controller (SC). As shown in figure 4, the DEVS model of SC which is described in the previous section IV (B). It controls the different event firing based on some given preconditions. A transition is fired once the conditions are satisfied. Although this section only illustrates the SC atomic model, we need to define other atomic and coupled models in the similar way.

Per the execution of DEVS simulator based on algorithm defined in section 3.2 and 3.3, the final desired output will look similar to example shown in figure 7.

```
FUNCTION SC
INPUT_VAR
C1 : BOOL;
MH1 : BOOL;
MH2 : BOOL;
END_VAR
OUTPUT_VAR
WM1 : BOOL;
WM2 : BOOL;
ZIG : BOOL;
END_VAR
IF  (C1=1 AND MH1=1 AND MH2=1) THEN
ZIG := 1;
END_IF
IF (ZIG=1) THEN
WM1 :=1;
WM2 :=1;
END_IF
END_FUNCTION
```

Figure 5: An example of IEC standard structured text

In this paper, we only applied the techniques for a simple work cell example; however, the procedure can be equally applied to more complex systems.

5 CONCLUSION AND FUTURE WORK

This paper proposed an integrated architecture for PLC based manufacturing line. The framework combines the high level simulation of the virtual factory and device level control code generation. As a result, the proposed framework improves the fidelity between high level system model and lower level PLC controlled devices. For this purpose, DEVS formalism and simulator has been used to model the factory and simulate it. As a result, sequence generated by the simulator can be mapped to PLC I/O addresses for generating the executable PLC program. Finally, these PLC program can be downloaded to software or hardware PLC for real implementation. This framework can reduce the PLC development time and down-time of a manufacturing industry. This is an ongoing research, our future endeavor is to improve DEVS simulator and implement it into real PLC based manufacturing line.

While implementing this approach in the real industries, we face many problems which need to be considered before implementing in the large scale industries. A major problem regarding implementation was dealing with different vendor's software like Rockwell, ABB, MELSEC, and SIEMENS etc. As an individual package they have pretty good properties and features, whereas, it is a cumbersome job to make integration between these various

packages. There is a strong need to develop a low cost integrated package program with plug 'n' play capability for complete solutions of verification and validation of the controller logic. Even the IEC 61131-3 standard categorized 5 types of program languages; it's a complex problem to make an integration among them.

Another major problem is state explosion problem. It confines the formal verification techniques to some simple applications. However, in real situation we need to implement these techniques in a very huge and complex scenario. Tackling these problem and full fledged deployment of this method to real situation is the open research area for academician as well as industrial researchers.

In spite of this, changing scenario in the manufacturing industries may remove the old fashioned PLC with PC controlled environment. It has opened a new door in the field of formal modeling and verification of control logic programs. Whatever the changes may be the role and importance of Modeling, Verification, and implementation of control logics will not relegate.

ACKNOWLEDGMENTS

This work has been partially supported by BK21 (Brain Korea 21st Century) and Digital manufacturing simulator and OLP (Off Line Programming) system for automobile industry project, South Korea.

REFERENCES

Canet, G., S. Couffin, J.-J. Lesage, A. Petit, and P. Schnoebelen. 2000. Towards the automatic verification of PLC programs written in Instruction List. In *Proceedings IEEE SMC*, USA, 2449-2454.

Clarke, E. M., O. Grunberg, and D. A. Peled. 1999. *Model checking*, The MIT Press Cambridge, England.

Hwang, M., and B. Choi. 2001. GK-DEVS: geometric and kinematics DEVS formalism for simulation modeling of 3-dimnesional multi-component systems. *The Society for Modeling and Simulation International* 18(3): 159-173.

Hwang, M. H. 2007. *DEVS++:C++ open source library of DEVS formalism*, available via <http://odevspp.sourceforge.net> [accessed 1 September, 2007].

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

Lee, S., M. A. Ang, and J. Lee. 2006. *Automatic generation of logic control.* Ford motor company, University of Michigan USA and Loughborough University UK.

Lucas, M. R., and D. M. Tilbury. 2003. A study of current logic design practices in the automotive manufacturing industry. *International journal of Human-Computer Studies* 59:725-753.

Park, S. C. 2005. A methodology for creating a virtual model for a flexible manufacturing system. *Computers in Industry* 56:734-746.

Thapa, D., C. M. Park, S. Dangol, and G. N. Wang. 2006. III-Phase verification and validation of IEC standard programmable logic controller. *IEEE Computer Society*, CIMCA Australia.

The International Electro Technical Committee. 1993. *IEC 61131-3, programmable controllers, programming languages*.

Wainer, A. G., and W. Chen. 2003. A framework for remote execution and visualization of cell-DEVS models, *The society for modeling and simulation international* 79(11):626-646.

Wu, Z. W. 2005. Modeling and simulation of an intelligent flexible manufacturing system via high-level objects Petri nets (HLOPN). *International Journal of Production Research* 43(7):1443-1463.

Zeigler P., H. Praehofer, and T. G. Kim. 2000. *Theory of modeling and simulation.* 2nd edition, Academic Press.

AUTHOR BIOGRAPHIES

DEVINDER THAPA is a Postdoc Research Fellow in the Department of Industrial & information systems at AJOU University, South Korea. He has completed his PhD in 22 Feb., 2008 from Ajou University in Industrial and Information Systems Engineering. His PhD thesis was regarding formal modeling, verification, and implementation of PLC program using Timed-MPSG. His area of research is related to manufacturing automation and intelligent decision support system. He can be reached at <debu@ajou.ac.kr>.

C. M. PARK is a research professor in the Department of Industrial & Information Systems Engineering at Ajou University. He has completed his PhD in 2002 from AJOU University, in Industrial Engineering. His research interest is related to Intelligent Information & manufacturing system, and image processing. He can be reached via email at <cmpark@ajou.ac.kr>.

SANG C. PARK is an assistant professor in the Department of Industrial & Information Systems Engineering at Ajou University. Before joining Ajou, he worked for DaimlerChrysler Corp. and CubickTek Co., developing commercial and in-house CAD/CAM/CAPP/simulation software systems. He received his BS, MS, and PhD degrees from KAIST in 1994, 1996, and 2000, respectively, all in industrial engineering. His research interests include geometric algorithms in CAD/CAM, process planning, engineering knowledge management, and discrete event system simulation. He can be reached via email at <scpark@ajou.ac.kr >.

Gi-NAM WANG is a department head and professor in the Department of Industrial & Information Systems Engineering at AJOU University, South Korea. He has completed his PhD in 1992 from Texas A&M University, in Industrial Engineering. He has worked as visiting professor at University of Texas at Austin during 2000-2001. His area of research is related to Intelligent Information & manufacturing system, system integration & automation, e-Business solutions and image processing. He can be reached via email at <gnwang@ajou.ac.kr>.

KWAN HEE HAN is an associate professor in the Department of Industrial & Systems Engineering at Gyeongsang National University, Korea. Before joining Gyeongsang, he worked for Daewoo Information Systems Company. He received his BS in Industrial Engineering from Ajou University in 1982. He received his MSIE and Ph.D. degree from KAIST in 1984, 1996, respectively. His research interests include simulation modeling, virtual manufacturing system and business process management. He can be reached via email at <hankh@gsnu.ac.kr>.