2012 IEEE Sixth International Conference on Software Security and Reliability

Reliability Analysis of PLC Systems by Bayesian Network

Hehua Zhang*, Yu Jiang†, Xun Jiao‡, Xiaoyu Song§, William N. N. Hung¶ and Ming Gu*
*School of Software, Tsinghua university, Beijing
†School of Computer Science and Technology, Tsinghua university, Beijing
§Dept. ECE, Portland State University, Oregon, USA.
¶Synopsys Inc., Mountain View, USA.
‡School of international studies, Beijing university of post and telecommunication, Beijing

***Abstract*—Reliability analysis is important in the life cycle of safety critical Programmable Logic Controller (PLC) system. The complexity of PLC system reliability analysis arises in handling the complex relations between hardware components and embedded software. Different embedded software may lead to different arrangements of hardware execution and different system reliability quantities. In this paper, we propose a novel probabilistic model, named hybrid relation model (HRM), for the reliability analysis of PLC systems. It is constructed based on the distribution of the hardware components and the execution logic of the embedded software. We map the hardware components to the HRM nodes and embed the failure probabilities of them into the well defined conditional probability distribution tables of the HRM nodes. Then, HRM model handles the failure probability of each hardware component as well as the complex relations caused by the execution logic of the embedded software, with the computational mechanism of Bayesian Network. Experiment results demonstrate the accuracy of our model.**

***Keywords*-Programmable Logic Controller; hybrid relation model; Bayesian Network; reliability analysis.**

I. Introduction

Reliability analysis has become an important part of PLC system life cycle. This is especially true for PLC systems performing critical applications such as nuclear power plants and spaceport devices control. Reliability is defined as the duration of its mission or the probability of generating a wrong output in given conditions. The typical task for the reliability analysis is to build a statistical mathematical model representing the system through a set of random variables. Then, distributions of these variables are fully specified to calculate the system reliability. For PLC systems, the reliability analysis refers to evaluating the effects of uncertain errors caused by noise, environment, and hardware execution. We need to calculate the failure probability of the system with these effects in consideration.

Traditionally, the reliability analysis of PLC systems is realized by combinatorial methods such as Fault Tree [1] and Reliability Block Diagram [2]. Fault Tree involves specifying a top event to analyze, such as the failure of the system, followed by identifying all associated events that could lead to the top event. It can be solved using techniques such as Binary Decision Diagrams [3]. Analysts extend the traditional Fault Tree by associating a particular markov process to the leaf node to improve the modeling power [4]. Reliability Block Diagram is a graphical depiction of the system components and connectors. It can be used to determine the overall system reliability, when the reliability of each system component is given. Similar work for extending the traditional Reliability Block Diagram with the markov process is presented in [5]. Recently, a more flexible modeling framework named Bayesian Network (BN) has been applied in system reliability [6], [7], [8]. It is based on the graphical and probabilistic reasoning theory for handling uncertain probabilistic events. System Reliability can be expressed as a joint probability function over some random variables and mapped onto a BN. If the qualitative part of the BN follows the logic structure of the system components, the causal dependencies among the system are thought to be handled. Some comparisons between BN and FT in terms of the modeling and analysis capabilities are presented in [9]. Dynamic Bayesian Network (DBN) is a generalization of BN, and is used for the reliability analysis in [10]. The generalization is introduced to deal with the temporal correlation between time slices of the system. Those methods are the most widely used models for reliability analysis. They are efficient and easy to use.

However, those methods present the designer and analyst with a high level abstraction of PLC systems, demonstrating the distribution and connection of the system components and events. They do not consider the complex relations among system components caused by the embedded software. For those reasons, we propose a novel probabilistic model named HRM to handle the distribution of the system components as well as the complex relations due to the embedded software. The HRM model construction is based on the translation of the embedded software. The constructed HRM model is a BN that captures the underlying dependency model of the execution logic of the embedded software. Each node of the HRM model is mapped to a corresponding system component, and the failure probability of the component is used to initiate the conditional probabilistic distribution (CPD) table. Then, the HRM model handles the failure probability of each system component and the execution logic of the embedded software, and supports both predictive and diagnostic inference about reliability

978-0-7695-4742-8/12 $26.00 © 2012 IEEE
DOI 10.1109/SERE.2012.26

properties of PLC system with the inference algorithms used in BN.

The paper is organized as follows. Section II introduces some background on PLC systems and Bayesian Network. Section III presents the proposed HRM model construction and initialization. Section IV presents the experiments on a real industry PLC system. Section V concludes the paper.

II. BACKGROUND

In this section, we summarize the core concepts of PLC systems and Bayesian Network.

A. PLC System

A simple PLC system is composed of a microprocessor, input devices and output actuators. Signals can be received from input devices such as sensors and switches, processed by the microprocessor according to the arrangement of the embedded software, and sent to actuators [11], [12], [13]. It is essentially an industrial control computer, which works in periodic scanning mechanism. Each cycle is composed of three stages as shown in Fig. 1. The first stage is sampling. The system reads the input data into the corresponding I/O. The second stage is processing. The microprocessor applies the embedded software to the control circuit, and refreshes the state of the I/O image area. The last stage is actuating. The microprocessor refreshes the output according to the state of the I/O image, and actuate the peripheral via the output circuit.

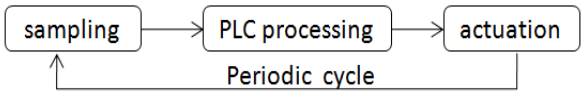

Figure 1. The three stages of a PLC system cycle

The embedded software is used to control and combine the signals of these system components together. The International Electrotechnical Committee has defined four standard programming languages for PLC [14]: ladder diagram (LD), instruction list (IL), functional block diagram (FBD), and structured text (ST). LD has its roots in the United States. It is based on the graphical presentation of the relay ladder logic. IL is the European counterpart of LD. As a textual language, it resembles assembler. FBD expresses the behavior of a controller as a set of interconnected graphical blocks, like in the electronic circuit diagrams. ST is a very powerful high-level language that is close to Pascal. In this paper, we focus on LD, which is one of the most popular programming language among PLC applications.

The basic principle of the ladder logic is the current flowing through the networks and rungs, from left to right, and top to bottom. There are two types of basic instructions in LD. The first type is the regular instructions representing

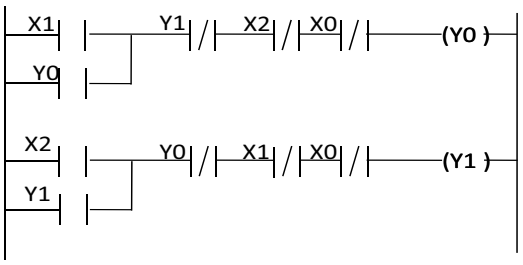

Figure 2. A simple ladder example for the motor reversible control

the conditions on the ladder rung. It is composed of contacts and logic connections. The second type is some special instructions such as timer, counter and coil. If a path can be traced through the asserted contacts, the rung is true and the output storage bit will be asserted true. Fig. 2 shows a simple LD program. It is made up of two ladder rungs. The symbol $-|\ |-$ is a normally open contact, representing a primary input. When the value of X_1 is 1, the contact stays in the closed state, and the current flows through the trace. The symbol $-|/|-$ is a normally closed contact. When the value of Y_1 is 0, the contact stays in the closed state, and the current flows through the trace. Detail information about the other instructions can be found in [14].

B. Baysian Network

Bayesian Network [15] is a directed probabilistic graphical model. Each node in the graph represents a random variable, and the arc between two nodes expresses the conditional probabilistic dependency between the two random variables. The formal definition of BN is the tuple $\langle U, E, P\rangle$:

- N is the set of nodes: $N = \{n_1, n_2, \cdots n_n\}$, n_i is the label of node.
- E is the set of arcs: $E = \{e_{ij}|there\ is\ an\ arc\ from\ node\ n_i\ to\ n_j\}$, e_{ij} is the label of arc.
- P is the set of conditional probability distribution: $P = \{f(n_i|parent(n_i))\}$. $parent(n_i)$ denotes the parents of node n_i, $f(n_i|parent(n_i))$ is the conditional distribution of variable n_i given all its parents.

The first two items make up a directed acyclic graph, which is the qualitative part of BN. The qualitative part is used to encode the conditional independence statements of a multivariate statistical distribution, representing that a variable is conditionally independent of its non-descendants given its parents. All these conditional independence can be extracted from a BN structure using the *d-separation* rules.

The third item is the quantitative part of BN. The conditional independencies embedded among the random variables are described through CPD tables of the corresponding nodes. CPD tables allow us to calculate the joint probability function in a simplified form (formula 2) compared to the original form (formula 1), where $f(n_1, n_2 \cdots n_n)$ is the joint distribution of those variables and $f(n_n|n_{n-1} \cdots n_1)$ is the

distribution of n_n given all the other variables:

$$f(n_1, n_2 \cdots n_n) = f(n_n|n_{n-1} \cdots n_1) \cdot f(n_{n-1}|n_{n-2} \cdots n_1) \cdots f(n_1) \quad (1)$$

$$f(n_1, n_2 \cdots n_n) = \prod_{i=1}^{n} f(n_i|parent(n_i)) \quad (2)$$

III. HRM MODEL CONSTRUCTION

This section introduces a probabilistic modeling methods of PLC systems embedded with the LD program. An algorithm is proposed to translate the LD program into the HRM model, the constructed HRM model is a BN that captures the underlying dependency model of the execution logic of the LD program, and some CPD tables for the HRM nodes are defined and initiated with the failure probabilities of the mapped system components.

A. Qualitative part structure construction

The execution logic of the embedded LD program is ignored by the original component based FT, RBD and BN methods. We intend to model the complex relation into an HRM model with an automatical method. The key challenge to build the HRM model is to organize these contacts, coils, special instructions and connections in a structured way. Since the microprocessor processes the input signals according to the arrangement of the ladder program from left to right and from top to bottom, we develop an iterative traversal algorithm for the translation. The translation algorithm is presented in Fig. 3. The algorithm makes use of a structure node **Struct node** {**char* type; node* left; node* right;**}. Based on the structure node, the algorithm builds an undirected graph model, and all the contacts, special instructions and connections are mapped to the nodes in the graph structure.

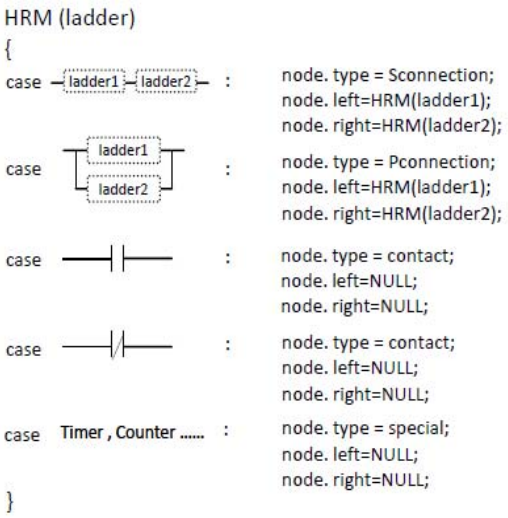

Figure 3. The algorithm to translate a LD into a graph structure

Let us see the algorithm in detail. The coil of each ladder rung is ignored and will be processed in the next procedure. In the first case, ladder2 is the minimal ladder block that is serially connected to the rest of the ladder logic block. We traverse the contacts of the ladder rung to find the last contact I and the last parallel connected structure IS, which are serially connected with the proceeding contacts. Then, ladder2 consists of the ladder block, starting from I and IS to the end of the ladder rung, and ladder1 consists of the rest of the ladder rung. The original ladder rung is translated into a *Sconnection*node, with two sub-graph structure construction tasks. In the second case, ladder1 and ladder2 are the maximal ladder logic blocks that are in parallel connection, which is processed similar to the first case. The atomic instructions of a ladder rung must be one of the last three cases, including regular instruction such as logic contacts, and special instructions such as timer. The result of the translation algorithm for the first ladder rung of Fig. 4 is a graph structure presented as follow.

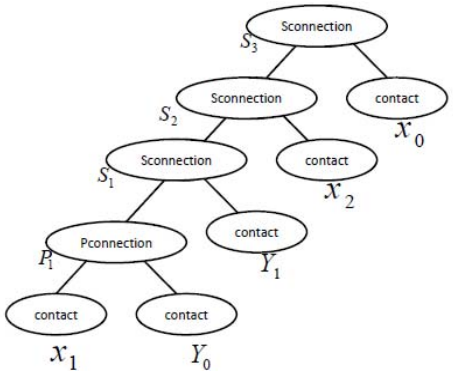

Figure 4. The graph structure got by the algorithm

When we get the undirected graph structure with the proposed algorithm, we can accomplish the construction process by the following actions: adding a coil node to the root of the graph structure, rotating the graph structure, and adding all the arcs with direction from the top to bottom. The final graph structure in Fig. 5 is the HRM model corresponding to the first ladder rung of Fig. 2. It is obvious that the execution logic of the LD program, processing from the left to right is captured in this directed acyclic graph (DAG), and we present more formal proof process below. If we apply this translation framework from the first ladder rung to the final ladder rung repeatedly, the execution logic of the LD program, processing from the top to bottom is also captured.

The execution logic of the LD program are mapped into the HRM nodes through the translation algorithm and the direction on the translated graph structure. Those PLC system components can be mapped to the contacts, special instructions, coils and connections of the LD program. For

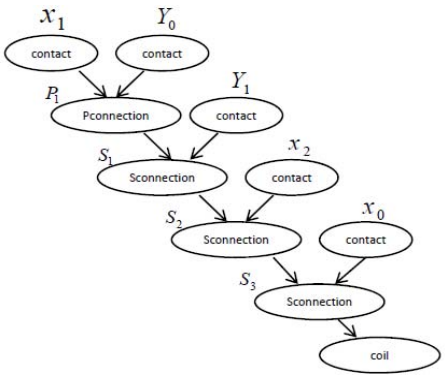

Figure 5. The HRM model for the PLC system controlled by the ladder diagram in Fig. 2

example, the contacts can be mapped to input devices such as sensors, the coils can be mapped to actuators such as motors, and the connections and special instructions can be mapped to the execution of the PLC microprocessor. We harmonize the system components and the embedded software through this mapping process.

B. Quantitative part structure construction

The quantitative part of BN is a set of CPD tables. Each CPD table is driven by the core distribution function $P = \{f(n_i^t|parent(n_i^t))\}$. The joint probability distribution can be expressed by the product of each conditional probability. In an HRM model, we have four types of nodes: the *coil* node, *contact* node, *special* node, and *connection* node. Each node can be mapped to a corresponding hardware component. We need to incorporate the failure probability of the corresponding hardware components into these variables by well defined CPD tables. Those CPD tables for different kinds of nodes are listed in the TABLE I, II, III, IV.

Table I
THE CPD FOR THE CONTACT NODE

$P(O_s\|x)$	$O_s = 10$	$O_s = 01$	$O_s = 00$	$O_s = 11$
1	ε_s	0	0	1 - ε_s
0	0	ε_s	1 - ε_s	0

TABLE I is the CPD table of the *contact* node. Because the *contact* node is mapped to the input devices such as the sensor and switch, we must use the failure probabilities of these components to initialize the table. In the table, x represents the correct input of the PLC system and ε_s represents the failure probability of the input devices. The *contact* node O_s has four possible values, where (1) 10 represents that the correct input should be 1, but the actual sampling value of the component turns out to be 0; (2) 01 represents that the correct input should be 0, but the actual sampling value of the component turns out to be 1; (3) 00 represents that the correct input should be 0, and the actual sampling value of the component is 0; (4) 11 represents that the correct input should be 1, and the actual sampling value of the component is 1.

Table II
THE CPD FOR THE SPECIAL INSTRUCTION NODE

$P(O_i\|r)$	$O_i = 10$	$O_s = 01$	$O_s = 00$	$O_s = 11$
1	ε_p	0	0	1 - ε_p
0	0	ε_p	1 - ε_p	0

The *special* node mapped to special instructions such as timer and counter can be regarded as an execution of the PLC microprocessor, just like an input sampling of the sensor. It is also translated as a leaf node with the proposed algorithm, just like the *contact* node, as shown in Fig. 4. Hence, the CPD table for the *special* node is the same as the structure of the TABLE II, except that ε_s must be changed to ε_p. ε_p is the reliability of PLC microprocessor, denoting the probability that the processor generates an error output of the instruction execution.

Then, let us consider the CPD table of the *connection* node. The CPD for the serial logic connection is presented in the TABLE III. The table is coincident to that of the *contact* node and *special* node. The connection between atomic instructions can be regarded as parallel and serial logic execution of the PLC microprocessor. Each *connection* node (O_e) has two parent nodes (O_1, O_2). The *connection* node also has four possible values. 10 represents that the correct output of this logic execution should be 1, but the microprocessor generates actual output 0 due to uncertain errors. The uncertain errors include the errors inheriting from its parent nodes, and the errors caused by the current logic execution of the PLC microprocessor. The other three values can be explained in the same way.

Table III
THE CPD TABLE FOR THE CONNECTION NODE THAT REPRESENTS THE SERIAL LOGIC EXECUTION OF PLC MICROPROCESSOR

$P(O_e\|O_1, O_2)$	$O_e = 10$	$O_e = 01$	$O_e = 00$	$O_e = 11$
(00, 00)	0	ε_s	1 - ε_s	0
(00, 01)	0	ε_s	1 - ε_s	0
(00, 10)	0	ε_s	1 - ε_s	0
(00, 11)	0	ε_s	1 - ε_s	0
(01, 00)	0	ε_s	1 - ε_s	0
(01, 01)	0	1 - ε_s	ε_s	0
(01, 10)	0	ε_s	1 - ε_s	0
(01, 11)	0	1 - ε_s	ε_s	0
(10, 00)	0	ε_s	1 - ε_s	0
(10, 01)	0	ε_s	1 - ε_s	0
(10, 10)	1 - ε_s	0	0	ε_s
(10, 11)	1 - ε_s	0	0	ε_s
(11, 00)	0	ε_s	1 - ε_s	0
(11, 01)	0	1 - ε_s	0	ε_s
(11, 10)	1 - ε_s	0	0	ε_s
(11, 11)	ε_s	0	0	1 - ε_s

Take the Fig. 5 as an example, we can map the *contact* node Y_1 to O_1, the parallel *connection* node P_1 to O_2, and generate the value of the serial *connection* node S_1. The

value of (Y_1, P_1) is (11, 01). It means that the correct output of the two nodes should be (1, 0), but the actual output of the two nodes is (1, 1). The error of P_1 may be caused by the sampling error inheriting from X_1 and Y_0, or by the logic execution of the PLC microprocessor. The error will be propagated to the serial *connection* node S_1. When S_1 inherits this error, the actual output will turn out to be 1 while the correct output should be 0. Further more, when the PLC microprocessor happens to be in error at the same time, the wrong output due to the error inhering from P_1 will be inverted. The error will be canceled out. Hence, the value of S_1 is 10 with probability $1 - \varepsilon_p$, and 11 with probability ε_p. Detail information about this case are presented in the fourteenth line of the TABLE III. The other lines can be explained in the same manner. The CPD table for the parallel connection node is the same as the structure of the TABLE III. The probability of each value is defined according to the semantic of the serial execution logic.

Finally, we initiate the CPD table for the *coil* node. It is presented in the TABLE IV. The *coil* node can be mapped to the actuator device of the PLC system. The actuator device is just controlled by the output signals of the PLC microprocessor. Hence, the *coil* node can not cancel out the errors inheriting from its parent node. When the value of its parent is 01 and 10, the actuator will be in error with probability 1. When the value of its parent is 00 and 11, the *coil* node will be in error with the probability ε_a. ε_a is the reliability of actuator, denoting the probability that the actuator fails to accomplish the task.

Table IV
THE CPD TABLE FOR THE COIL NODE

$P(O_c\|O_e)$	$O_c = 10$	$O_c = 01$	$O_c = 00$	$O_c = 11$
00	0	ε_a	1 - ε_a	0
01	0	1	0	0
10	1	0	0	0
11	ε_a	0	0	1 - ε_a

We have incorporated the reliability of the PLC input devices, PLC microprocessor and actuators into these CPD tables. What we need is to change the value of $\varepsilon_s, \varepsilon_p$, and ε_a according to the stated operation environment. We can perform predictive and diagnostic inferences to study the behavior of the whole system or a single component.

IV. EXPERIMENT RESULT

Given an HRM model and the corresponding CPD tables, we can evaluate all possible predictive and diagnostic inferences efficiently. A number of algorithms have been invented to solve the inference problem. One of the most popular algorithms is the message passing algorithm, which is an exact inference algorithm. It is based on the local message passing on a junction-tree structure, the nodes of which are subsets of the original random variables. It is computationally expensive when there are many variables. Some approximate inference algorithms have been invented to solve the limitation, such as Probabilistic Logic Sampling [16] and Evidence Pre-propagated Importance Sampling [17]. All the approximate inference methods are based on sampling techniques. Based on those two kinds of inference algorithms, many software tools such as BUGG, CODA and Nertica [18], [19], [20] have been implemented.

We apply our framework to an actual industrial PLC system, which was originally published in [21]. The hardware component distribution of the system is shown in Fig. 6. It consists of four pistons (A, B, C, D) which are operated by four solenoid valves (V_1, V_2, V_3, V_4). Each piston has two corresponding normally open limit sensor contacts. Three push buttons are provided to start the system, to stop the system normally and to stop the system immediately in emergency. In a manufacturing facility, such piston system can be used to load/unload parts from a machine table, and extend/retract a cutting tool spindle.

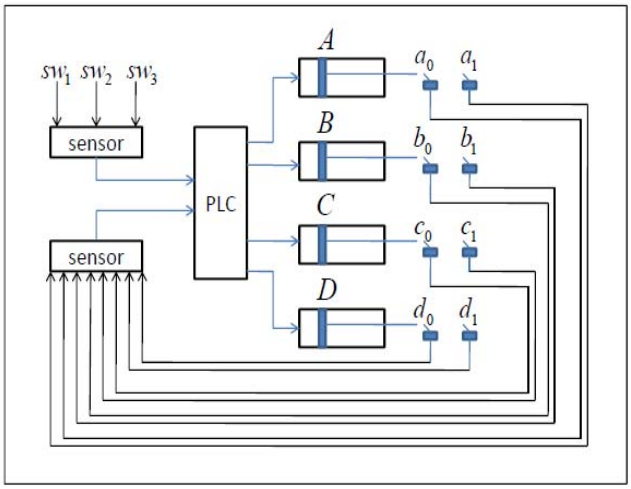

Figure 6. Industrial automated system originally published in [21]. The input devices of this PLC system are the sensors. The actuators are the four pistons. The four embedded control ladder program is introduced in [21]. The third program is introduced in detail in [22]

There are four embedded LD programs used to control the above hardware component deployment. The four LD programs are presented in Fig. 7, 8, 9, 10. We set the failure probability of these system components in TABLE V. We can also change the failure probability of the processor and the others keep the same.

Table V
THE STATIC FAILURE PROBABILITY OF EACH COMPONENT

component	failure probability
sensors	ε_s 0.05
switches	ε_s 0.05
PLC microprocessor	ε_p 0.05
pistons	ε_a 0.05

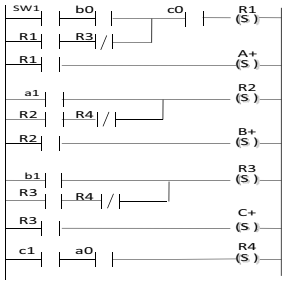

Figure 7. The LADDER1 used to control the piston system in Fig. 6

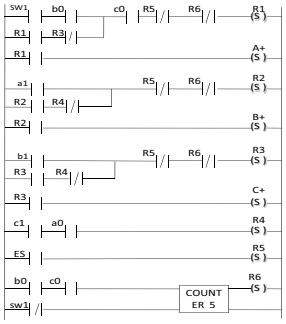

Figure 8. The LADDER2 used to control the piston system in Fig. 6

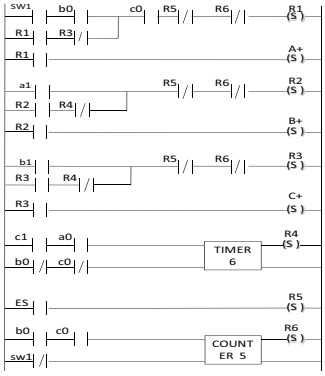

Figure 9. The LADDER3 used to control the piston system in Fig. 6

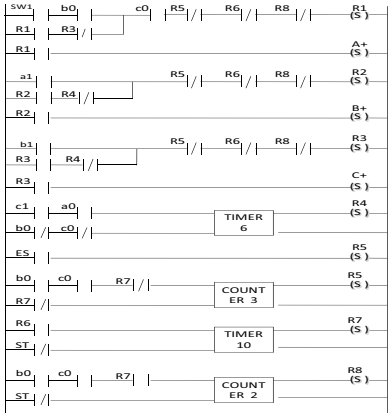

Figure 10. The LADDER4 used to control the piston system in Fig. 6

Table VI
THE FAILURE PROBABILITY FOR THE EMBEDDED SYSTEM WITH THE LADDER1

processor ε	BN	HRM	simulation
0.02	0.57%	0.73%	0.82%
0.04	0.94%	1.34%	1.41%
0.06	4.61%	5.37%	5.64%
0.08	8.03%	9.26%	9.61%
0.10	9.24%	14.21%	14.81%

Table VII
THE FAILURE PROBABILITY FOR THE EMBEDDED SYSTEM WITH THE LADDER2

processor ε	BN	HRM	simulation
0.02	0.57%	0.81%	0.89%
0.04	0.94%	1.53%	1.62%
0.06	4.61%	6.47%	6.78%
0.08	8.03%	11.26%	11.92%
0.10	9.24%	16.02%	16.67%

Table VIII
THE FAILURE PROBABILITY FOR THE EMBEDDED SYSTEM WITH THE LADDER3

processor ε	BN	HRM	simulation
0.02	0.57%	1.09%	1.20%
0.04	0.94%	2.46%	2.81%
0.06	4.61%	7.03%	7.31%
0.08	8.03%	11.97%	12.21%
0.10	9.24%	17.17%	17.50%

Table IX
THE FAILURE PROBABILITY FOR THE EMBEDDED SYSTEM WITH THE LADDER4

processor ε	BN	HRM	simulation
0.02	0.57%	1.22%	1.30%
0.04	0.94%	2.71%	2.82%
0.06	4.61%	7.45%	7.52%
0.08	8.03%	12.37%	12.62%
0.10	9.24%	18.57%	18.92%

For this application, the HRM model is constructed with the proposed algorithm for each LD program and the conditional probabilities are assigned by the well defined CPD tables. We use Netica for compiling the junction tree and propagating the probabilities, to compute the reliability characterization of these systems. We also calculate the reliability characterization of those systems with the original component-based RBD and BN methods. Finally, we devise some random simulations to confirm the correctness of the reliability characterization. The simulations run on a computer with CPU 3.06 GHz and 2 GB of memory. The Monte Carlo framework for reliability analysis is based on fault injection. We embed the error probabilities of the system components into the Monte Carlo simulator. Then, the failure probabilities of the system corresponding to the four LD programs are listed in the TABLE VI, VII, VIII, IX, respectively. Where the first column is the failure probability of the processor, the second column is the reliability of the system based on original component and event based

on original component and event based Bayseian Network framework presented in Section I, the third column is the failure probability of the system based on HRM model, and the fourth column is based on random simulations.

As can be observed from the simulation results, presented in the fifth column of the TABLE VI, VII, VIII, IX, the final failure probability of the piston system are different when they are controlled by different LD programs. With the same PLC processor failure probability, more complex LD program will lead to higher failure probability. Because more complex LD program decides more complex arrangements of the logic executions of the system components. While the reliability characterization obtained by the original component-based BN method are the same, as presented in the second column column of the four tables. The values are not closed to the simulation results. Because they mainly consider the distribution of the system components and the signal dependencies among them, and the complex relations caused by the execution logic of the LD program are ignored. The HRM model based method is more accurate. As shown in the third column of each table. The error between the HRM model results and the simulation results is less than 3%. The results get by the HRM model is more closed to the run time station. Because the failure probability of the single system component and the complex execution logic of the control program are captured by the HRM model and the CPD tables. Furthermore, for the four LD programs, the total time for the HRM model construction of our algorithm and the evidences propagation in Nertica is within 0.1 second.

From those results and the complexity analysis, it is reasonable to draw the conclusion that the proposed HRM model is more straightforward and close to the simulation results than that of the original component-based BN and RBD framework, and the calculation time consumption is not expensive. Moreover, the flexibility of the HRM model is also useful, because for different samples of failures, only the failure probability of the component is changing. The constructed HRM model of the system will not change, and the CPD table for each node needs to be adapted only once. It makes the HRM model based reliability analysis very efficient for large PLC systems with complex LD programs.

V. CONCLUSION

In this paper, we propose a hybrid relation model for the reliability analysis of PLC system. The model is constructed based on the embedded LD program. The execution logic of the embedded program is mapped to the hybrid relation model through the proposed translation algorithm. The model consists of four kinds of nodes and some arcs among those nodes. The hybrid relation model is a Bayesian Network. Then, we define some conditional probability distribution tables for the nodes according to the semantic of each kind of node. All the nodes can be mapped to some corresponding hardware component through the conditional probability table initialization. Through the two mapping processes, both the execution logic of the embedded control software and the hardware components are captured. Then, it is straightforward, and provides us a convenient way to carry on predictive inferences about the reliability properties of PLC systems.

ACKNOWLEDGMENT

This research is supported in part by 973 Program (No.2010CB328003) of China.

REFERENCES

[1] W. Lee, D. Grosh, and F. Tillman, "Fault tree analysis, methods, and applications- a review." *IEEE transactions on reliability*, vol. R-34, pp. 194–203, 1985.

[2] M. L. Shooman, *Reliability of Computer Systems and Networks: Fault Tolerance,Analysis,and Design*. New York, NY, USA: John Wiley & Sons, Inc., 2002.

[3] X. Zang, H. Sun, and K. Trivedi, "A BDD-based algorithm for reliability evaluation of phased mission systems," *IEEE Transactions on Reliability*, vol. 48, no. 1, pp. 50–60, 1999.

[4] M. Bouissou and J. Bon, "A new formalism that combines advantages of fault-trees and Markov models: Boolean logic Driven Markov Processes," *Reliability Engineering & System Safety*, vol. 82, no. 2, pp. 149–163, 2003.

[5] S. Distefano and L. Xing, "A new approach to modeling the system reliability: dynamic reliability block diagrams," in *Reliability and Maintainability Symposium, 2006. RAMS'06. Annual*. IEEE, 2006, pp. 189–195.

[6] D. Wooff, M. Goldstein, and F. Coolen, "Bayesian graphical models for software testing," *IEEE Transactions on Software Engineering*, pp. 510–525, 2002.

[7] C. Bai, Q. Hu, M. Xie, and S. Ng, "Software failure prediction based on a Markov Bayesian network model," *Journal of Systems and Software*, vol. 74, no. 3, pp. 275–282, 2005.

[8] H. Boudali and J. Dugan, "A discrete-time bayesian network reliability modeling and analysis framework," *Reliability Engineering & System Safety*, vol. 87, no. 3, pp. 337–349, 2005.

[9] A. Bobbio, L. Portinale, M. Minichino, and E. Ciancamerla, "Improving the analysis of dependable systems by mapping fault trees into Bayesian networks," *Reliability Engineering & System Safety*, vol. 71, no. 3, pp. 249–260, 2001.

[10] "Complex system reliability modelling with Dynamic Object Oriented Bayesian Networks (DOOBN)," *Reliability Engineering and System Safety*, vol. 91, no. 2, pp. 149–162, 2006.

[11] K. Erickson, "Programmable logic controllers," *Potentials, IEEE*, vol. 15, no. 1, pp. 14–17, 1996.

[12] W. Bolton, *Programmable logic controllers*. Newnes, 2009.

[13] G. Dunning and Dunning, *Introduction to programmable logic controllers*. Delmar Thomson Learning, 2002.

[14] *IEC 61131-3 Standard (PLC Programming Languages)*, 2nd ed., International Electrotechnical Commission (IEC), 2003.

[15] J. Pearl, "Fusion, propagation, and structuring in belief networks* 1," *Artificial intelligence*, vol. 29, no. 3, pp. 241–288, 1986.

[16] M. Henrion, "Propagating uncertainty in Bayesian networks by probabilistic logic sampling," in *Uncertainty in artificial intelligence*, vol. 2, 1988, pp. 149–163.

[17] C. Yuan and M. Druzdzel, "An importance sampling algorithm based on evidence pre-propagation," in *Proceedings of the 19th Annual Conference on Uncertainty on Artificial Intelligence*, 2003, pp. 624–631.

[18] W. Gilks, A. Thomas, and D. Spiegelhalter, "A language and program for complex Bayesian modelling," *Journal of the Royal Statistical Society. Series D (The Statistician)*, vol. 43, no. 1, pp. 169–177, 1994.

[19] N. Best, M. Cowles, and S. Vines, "CODA Manual version 0.30," *MRC Biostatistics Unit, Cambridge, UK*, vol. 46, pp. 2020–2027, 1995.

[20] N. Manual, "Netica V1.05," *Norsys Software Corp*, 1997.

[21] K. Venkatesh, M. Zhou, and R. J. Caudill, "Comparing ladder logic diagrams and petri nets for sequence controller design through a discrete manufacturing system," *IEEE Transactions on Industrial Electronics*, vol. 41, no. 6, pp. 611–619, December 1994.

[22] H. Zhang, Y. Jiang, W. N. N. Hung, X. Song, and M. Gu, "Domain-driven probabilistic analysis of programmable logic controllers," in *Proceedings of the 13th international conference on Formal methods and software engineering*. Springer-Verlag, 2011, pp. 115–130.