System Reliability Calculation Based on the Run-Time Analysis of Ladder Program

Yu Jiang, Hehua Zhang, Han Liu, William N. N. Hung, Xiaoyu Song, Ming Gu and Jiaguang Sun

***Abstract*—Programmable logic controller (PLC) system, a typical member in the embedded family, is now widely applied in industry. For safety critical PLC systems, reliability is of top significance. However, due to subcomponents' temporal correlations caused by the run-time execution of embedded ladder programs, the complexity of reliability analysis is greatly increased. In this paper, we propose a novel probabilistic model to analyze reliability of PLC systems, called run-time reliability model (RRM). RRM is automatically constructed based on the structure and run-time execution of the embedded ladder program. Moreover, it is also a dynamic bayesian network (DBN) capturing full dependencies in a PLC system. Then, according to execution semantics of RRM nodes, we present customized conditional probability distribution (CPD) tables to calculate final reliability of the system, with failure probability of every referenced component as refinement. The strength of this model is that not only does it explicitly specify the correlations between run-time execution of embedded software and system components, but also it serves as a computational mechanism for probabilistic inference. Besides, the proposed approach is superior to previous works in both accuracy and efficiency. Compared to monte carlo based simulation, the average error rate of reliability values inferred from RRM model is small.**

***Index Terms*—Programmable Logic Controller, Ladder Program, Dynamic Beyesian Network, Run-Time Reliability Model.**

I. Introduction

In industrial practice, high fault rate imposes negative effect on the performance of embedded systems. This is especially true for PLC systems with critical applications running, such as nuclear power controlling plants and spaceport devices [1]. Reliability analysis of PLC systems refers to evaluating the effects of errors due to failure on individual system components, such as sensors, actuators, processors, etc. As a result, many attempts are motivated to find accurate and scalable reliability analysis methods for complex system designs [2]. Reliability is defined as the probability that a system will perform its intended function during a specified period of time under stated conditions. The typical task for reliability analysis is to build a mathematical model to represent the system with a set of random variables. Based on the environment, design weakness and mishandling, distributions of theses variables are specified to calculate a system level reliability [3],[4], [5]. Most of the exiting models focus on critical hardware parts of the system. Since the widespread use of digital integrated circuit technology, software has been gaining increasing importance in most systems. Because interactions between hardware and software components are complex, a recent surge in demand for methods to capture correlations of the software and hardware components in reliability engineering springs up.

In this paper, we propose a novel probabilistic model, named RRM, to handle the spatial dependencies and the high order temporal dependencies, with both hardware and software. From a global perspective, main contributions of our approach are: (1) We propose a model for reliability analysis including the spatial dependencies among system components in a single time slice, and the run-time temporal dependencies among system components caused by the execution of the embedded ladder program. We prove that the model is a minimal representation of the underlying dependency model of the spatial dependencies as well as the run-time temporal dependencies of the system, and hence is a DBN. (2) With our customized CPD tables, reliability analysis is accurate, fast and scalable for complex designs. We prove that the constructed RRM model is a DBN that captures the execution semantic of the embedded ladder program totally. Then, we present some customized CPD tables according to the execution semantics of the RRM nodes, and initialize those tables with the failure probability of the corresponding system hardware components. Finally, the joint reliability can be mapped to an RRM model. The model preserves the underlying dependencies of the spatial correlations and the run-time temporal correlation of the embedded ladder program.

Manuscript received July 13, 2013; revised October 28, 2013 and December 18, 2013; accepted March 6, 2014.

Yu Jiang is with the Department of Computer Science and Technology, School of Software, Tsinghua university, Tsinghua National Laboratory for Information Science and Technology, Key Laboratory for Information System Security, Ministry of Education, China. (Phone: (+86)13810353960; e-mail:jiangyu198964@gmail.com)

Hehua Zhang, Han Liu, Ming Gu and Jiaguang Sun are with the School of Software, Tsinghua university, Tsinghua National Laboratory for Information Science and Technology, Key Laboratory for Information System Security, Ministry of Education, China. (Phone: (+86)13691349135; e-mail:zhanghehua@gmail.com)

Xiaoyu Song is with the Dept. ECE, Portland State University, USA.

William N. N. Hung is with Synopsys Inc., Mountain View, USA.

This research is supported by NSFC Programs (No.61202010, No.91218302), National Key Technologies R&D Program (No.SQ2012BAJY4052) and 973 Program (No.2010CB328003) of China, and Tsinghua University Initiative Scientific Research Program (20131089331).

II. Related Work

Traditionally, reliability analysis techniques mainly contain combinatorial methods such as fault tree (FT) [6], reliability block diagram (RBD) [7], and bayesian network (BN) [8]. FT is a top-down, deductive reasoning method where an undesired state of a system is analyzed using boolean logic to combine a series of lower-level events. The basic symbols used in FT are grouped as events, gates, and transfer symbols. Gate symbols are derived from boolean logic symbols and used to describe the relationship between input and output events. It involves specifying a top event for analysis, such as the failure of the system, and identifying all associated events that could lead to the top event. Then, it can be solved using techniques such as binary decision diagrams (BDD)[9]. However, FT only represents a logical function and is insufficient in handling complex functional dependencies between internal components of the system. In order to enhance the modeling power of FT, analysts extend traditional FT by associating a particular markov process to the leaf nodes, named Dynamic Fault Tree(DFT) [10], [11]. RBD is also a diagrammatic method for showing how component reliability contributes to the success or failure of a complex system. Similar work for extending the traditional RBD with the markov process, named Dynamic Reliability Block Diagram(DRBD), is presented in [12]. BN has been applied in system reliability [8] with more flexible modeling capabilities. 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 can be mapped onto a BN. If the qualitative part of BN follows the logic connection of system components, the causal dependencies of the system are then thought to be captured. However, they present the designer with a high level abstraction, to demonstrate the distributions of the system components and events. According to our knowledge, most analysis works based on these techniques can not deal with the complex relations among system components caused by the run-time execution of embedded ladder program, automatically. They mainly focus on critical hardware parts of the system or model some dynamic behaviors manually, while softwares have been gaining increasing significance in this domain.

Hence, some custom work for reliability analysis of PLC systems with ladder program are conducted. In [13], the researchers model the ladder diagram as an abstract syntax tree and give a single topological traverse through the primary inputs during the software execution. In [14], the PLC system is built as a hidden markov model and is solved by some predefined domain knowledge. Then, the run-time reliability probability can be tested by some formal techniques, with probabilistic model checking. In [15], they propose a probabilistic model based on BN. The ladder program is modeled as a hybrid relation model(HRM). The model only captures the dependencies of the system in a single time slice. Those custom methods improve the accuracy of reliability analysis to a certain extent, but none of them considers the temporal dependencies of multiple time slices or supports the automatic analysis. This is also the original motivation of this paper. Based on the descriptions above, the main differences between the proposed model and the previous techniques are abstracted in the table I. More detail quantitative comparisons are presented in the experiment result section.

III. Background

The programmable logic controller is essentially an industrial computer designed to execute specific tasks quickly and efficiently. Its sole purpose is to automate processes such as assembling lines, manufacturing cells, and material handling. The system consists of the following components: input module, micor-processor cooperated with the embedded ladder program, memory, and output module. The hardware block diagram is presented in the Fig. 1. The solid lines among hardware blocks indicate the information flow direction. Signals can be received from input modules and memory, processed by the microprocessor according to the arrangement of the embedded software, and sent to output modules [16].

TABLE I
THE LEVEL OF SUPPORT FOR EACH RELIABILITY TECHNIQUE

SUPPORT FOR PLC SYSTEM		FT	RBD	BN	HRM	**RRM**
Spatial dependence	component level	√	√	√	√	√
	code level				√	√
Temporal dependence	single period	√	√	√	√	√
	multiple periods					√
Dependence abstraction	manual	√	√	√	√	√
	automatic					√

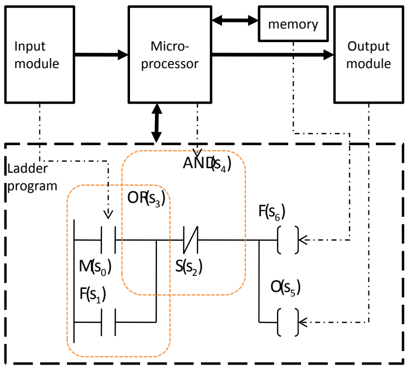


Fig. 1. The hardware blocks of PLC system. The simple ladder program is for the motor control system. M is the motor start button, S is the stop button, and F is the memory storage unit. M is the name of the contact.

The dotted block contains the embedded application programs. The international electrotechnical committee has defined four standard programming languages for PLC system [17]: ladder diagram (LD), instruction list (IL), functional block diagram (FBD), and structured text (ST). The most widely used application program is called the ladder diagram, which appears in many PLC systems currently on the market. Different brands of PLC may have different formats for ladder program instructions, but all formats share common elements and can be manually converted. Each component in the ladder diagram is called an instruction, which can be connected in parallel and serial to create a ladder rung. Typical logic instructions include: **Normally open contact** is used to denote the examination of closed status, and its symbol is $-|\ |-$. For example, when the value of M in Fig. 1 is 1, the contact stays in the closed status, and the path can be traced through this contact. **Normally closed contact** is used to denote the examination of open status, and its symbol is $-|/|-$. For example, when the value of S is 0, the contacts stays in open status, and the path can be traced through this contact. **Parallel and Serial connections** are used to denote the relation of two logic blocks, similar to the semantics of circuits AND and

OR. For example, the normally open contacts M and F is in parallel connection, and their result $OR(s_3)$ is connected to the normally closed contact $S(s_2)$ in serial, denoted by $AND(s_4)$. Other instructions such as **timer, counter** can be connected at the right side of the ladder rung. We can find detailed explanations for those instructions in [17]. When the micro-processor executes the ladder program, it will find a path traced through each ladder rung, from left to right, and from top to bottom. The dotted lines from hardware blocks to program blocks indicate some mapping from the hardware components to the instruction elements of the ladder program.

IV. Run-time Reliability Model

In this section, we introduce the probabilistic model named RRM for reliability analysis. RRM model can be described as a triple: $\langle \cup_{t=1}^{n}\{x_1^t, x_2^t, \cdots x_n^t\}, (\cup_{t=1}^{n}(x_i^t, x_j^t)) \cup (\cup_{t=1}^{n-1}(x_i^t, x_j^{t+1})), f(x_i^t | \ parent(x_i^t))\rangle$, where the first two elements are the qualitative part and the third element is the quantitative part. The model construction algorithm and the proof that the constructed RRM is a DBN are described in detail. The methods to define and initialize some customized CPD tables are based on the failure probabilities of the mapped system components and the execution semantics of the nodes.

A. Constructing Graph Structure

As mentioned in the background section, PLC system works under a periodic scanning mechanism according to the embedded ladder program. We intend to model these complex relations into an RRM model automatically. The key challenge of building an RRM model is to organize these contacts, coils, special instructions and connections in a well-formed way with the corresponding run-time execution logic. The steps to construct the qualitative part is presented as follows.

The first step is to construct a BN for the PLC system, considering the execution logic of the ladder program in a single period, while ignoring the correlations with multiple periods. Since the microprocessor processes the input signals according to the arrangement of 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 algorithm 1. Based on the structure node and the algorithm, all contacts, special instructions, connections and execution logic of ladder program in a single period can be captured in the generated directed graph structure. In detail, the function $read\_file$ is used to parse the five kinds of structure of the ladder program. In the serial connection structure presented in the first case, *ladder2* is the minimal ladder block that is serially connected to the rest of ladder logic block. We traverse the contacts of the ladder rung to find the last contact I or the last parallel connected structure IS, where I or IS is serially connected with the proceeding contacts. The *ladder2* consists of the ladder block I or IS, and *ladder1* consists of the rest of the ladder rung. The original ladder rung is translated into a *Serial_Connection* node, with two sub-graph structure construction tasks. In the parallel connection structure presented in the second case, *ladder1* and *ladder2* are the maximal ladder logic blocks that are in parallel connection,

Algorithm 1: Translation Algorithm

```
Translation_RRM(File Ladder){
repeat
    switch ($structure$) do
        case File Ladder1 AND File Ladder2
            Tree_nodetype = Serial_Connection;
            Tree_nodeleft =
            Translation_RRM(File Ladder1);
            Tree_noderight =
            Translation_RRM(File Ladder2);
        endsw
        case File Ladder1 AND OR Ladder2
            Tree_nodetype = Parall_Connection;
            Tree_nodeleft =
            Translation_RRM(File Ladder1);
            Tree_noderight =
            Translation_RRM(File Ladder2);
        endsw
        case Input Contact,Output Coil,Special
            Tree_nodetype = Contact,Special;
            Tree_nodeleft = NULL;
            Tree_noderight = NULL;
        endsw
    endsw
until $structure = eof$;
}
```

which can be processed similarity to the first case. The atomic instructions of a ladder rung must be one of the last three cases, including the regular instructions such as logic input contacts and output coils, and special instructions such as timer. The timer and others special instructions are captured in the default branch of the switch block. When we use the translation algorithm to parse the ladder program in Fig 1, the result is presented in Fig 2.

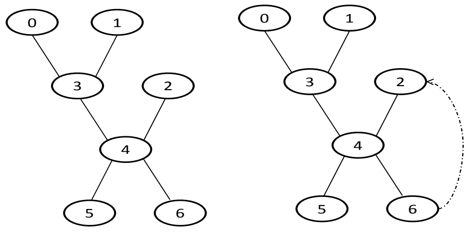


Fig. 2. The left is the constructed BN model for the execution logic of a single period, and the right is the graph structure capturing the correlations between variable 6 and 2. In the ladder program in Fig 1, the output signal s_6 is stored into memory for the input of signal s_2. (the dotted line is not an arc of BN, it is used to present correlation dependencies of multiple periods)

The second step is to unroll the constructed BN to capture the high order temporal dependencies among several periods caused by the runtime execution of ladder program. This can be done by adding arcs between the nodes from adjacent time slices and correlated signals. The number of time slices is dependent on its underlying structure and different systems and applications may need different amount of time slices to capture the temporal dependencies. For example, the motor control system just needs two cycles to finish the motor start application. The unrolled BN structure corresponding to the above BN is shown in the Fig. 3. There are two kinds of temporal correlations. The thick dotted line between node x_i^t and x_i^{t+1} for any i denotes the potential temporal correlation of signal i. The thick dotted line, between node x_i^t and x_j^{t+1} for some i and j, denotes the temporal correlation caused by run-time feedback of the ladder program. We also need to cascade the translated graph structure of the first ladder rung to the node corresponding to the input contact.

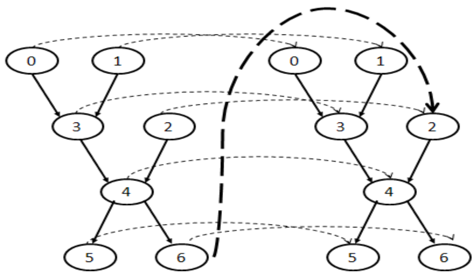


Fig. 3. Unrolled graph structure for the corresponding BN. All possible dependencies are presented first. For example, if the signal s_0 read form the sensor turns out to be wrong, it is possible that the sensor is wrong. Hence, the value of s_0 will also turns out to be wrong in the next cycle with high probability. Hence, there should be a link between s_i^j and s_i^{j+1} .

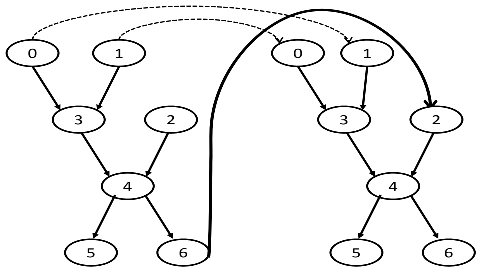


Fig. 4. Qualitative part of the RRM model for the motor control PLC system. The application just needs tow cycles of period to finish, hence, we unroll the basic BN of the ladder program for two slices. When the probability distributions of s_0^2 and s_1^2 are known, the value of signal s_3^2 is known even without the value of signal s_3^1.

The third step is to modify the unrolled BN to get RRM, which will be proved to be a DBN in the following subsection. This can be done by deleting some arcs to make the unrolled BN structure minimal. For example, given the node 2 and node 3 at time slice 2, the node 4 at time slice 2 is independent of the node 4 at time slice 1. We also assume that the root node signal is independent between time slices. Then, we can delete the arcs between the node x_i^t and x_i^{t+1} for any i while preserving the dependency correlations. The modified RRM is shown in Fig. 4, and the high order temporal dependence caused by the run-time feedback between signal x_6 and x_2 is captured by the thick full line.

Finally, we can draw the conclusion that the constructed RRM model in Fig. 4 captures the underlying dependency model of the ladder program presented in Fig 1. The proof process can be referenced from the support files [18]. The execution logic of the ladder program can be mapped onto the RRM nodes through the translation algorithm and the arcs of the translated graph structure. The contacts, special instructions, coils and connections of the ladder program can be mapped to PLC system components. 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. In this way, the system components and the embedded software are harmonized together.

B. Constructing Conditional Probabilistic Distribution

In RRM, we have five types of nodes: *coil*, *contact*, *special*, *parallel_connection*, and *serial_connection* nodes. Each node can be mapped to a corresponding hardware component. We need to incorporate the reliability of the corresponding hardware components into these variables by customized conditional probability distribution tables. Tables are constructed according to execution semantic of the RRM nodes and the corresponding hardware component.

TABLE III and II are the conditional probabilistic distribution tables of *contact* node. The *contact* node x_i^t has four possible values 10, 01, 00, and 11, where 01(10) represents that the correct input should be 0(1), but the actual sampling value of the input module or the value read from the memory turns out to be 1(0), and 00(11) represents that the correct input should be 0(1), and the actual sampling value of the sensor or the value read from the memory is 0(1). TABLE II is the conditional probabilistic distribution table for the contact node that is mapped to the memory, where x_i^t is the variable read form the memory, x_j^{t-1} represents the parent of x_i^t from the previous period. As presented in the qualitative part construction and the proof procedure, the value read from the memory are produced by the output of the previous period. This captures the high order correlations of the run-time dependencies. The errors inherited from the parent nodes may be propagated to son nodes when the current node works well, and may be canceled when the current node fails. This rule also applies to the instructions presented in the other CPD tables except for the coil node. When the parent signal is correct with 11(00), x_i^t will take the value 1(0) with probability ε_m, where ε_m is the reliability of the component feedback memory, denoting the reliability of the memory reading and writing. The other cases can be expressed in similar. For example, in Fig. 4, when the signal x_6^1 is correct with 11, then, the reliability of feedback signal x_2^2 is ε_m.

TABLE II
CPD FOR THE CONTACT NODE MAPPED TO MEMORY

$P(x_i^t \vert x_j^{t-1})$	$x_i^t = 10$	$x_i^t = 01$	$x_i^t = 00$	$x_i^t = 11$
11	$1-\varepsilon_m$	0	0	ε_m
00	0	$1-\varepsilon_m$	ε_m	0
10	ε_m	0	0	$1-\varepsilon_m$
01	0	ε_m	$1-\varepsilon_m$	0

TABLE III is the CPD table for the contact node that is mapped to the input devices such as switch and sensor. I represents the correct input of the PLC system and ε_s represents the reliability probability of the input devices. For example, when the correct input is 1, the x_i^t will take value 0 when the sensor is in failure with probability $1 - \varepsilon_s$. For sensors that sample multi-bits single time, it can be defined on the combination of the TABLE III. Corresponding to the input module, we can define the conditional probabilistic distribution tables of *coil* nodes that are mapped to output module in the same manner.

TABLE III
CPD FOR THE CONTACT NODE MAPPED TO INPUT MODULES

$P(x_i^t \| I)$	$x_i^t = 10$	$x_i^t = 01$	$x_i^t = 00$	$x_i^t = 11$
1	1- ε_s	0	0	ε_s
0	0	1 - ε_s	ε_s	0

The conditional probabilistic distribution table for the *special* node is also considered. Those instructions can be regarded as an execution of the microprocessor. As to the counter instruction, the result of the instruction **counter N** is determined by the conditional input and the current count value. When the current value of the counter is between 0 and the preset value N, the conditional input changes from 0 to 1, and the value of the reset variable is 0, then, the output of the instruction is 1. Otherwise, the output of the instruction is 0, and the current value increases by 1. The conditional probability distribution table for the *special* node mapped to **counter N** can be derived in the same manner.

TABLE IV
CPD TABLE FOR SPECIAL NODE MAPPED TO COUNTER INSTRUCTION WITH PRESET VALUE N

$P(x_i^t \| x_j^t, x_k^t)$	$x_i^t = 00$	$x_i^t = 01$	$x_i^t = 10$	$x_i^t = 11$	$Cond$
(00, zz)	ε_p	$1 - \varepsilon_p$	0	0	$cv < N$
(01, z0)	$1 - \varepsilon_p$	ε_p	0	0	$cv < N$
(01, z1)	ε_p	$1 - \varepsilon_p$	0	0	$cv < N$
(10, 0z)	0	0	ε_p	$1 - \varepsilon_p$	$cv < N$
(10, 1z)	ε_p	$1 - \varepsilon_p$	0	0	$cv < N$
(11, 00)	0	0	$1 - \varepsilon_p$	ε_p	$cv < N$
(11, 01)	0	0	ε_p	$1 - \varepsilon_p$	$cv < N$
(11, 10)	$1 - \varepsilon_p$	ε_p	0	0	$cv < N$
(11, 11)	ε_p	ε_p	0	0	$cv < N$
(zz, zz)	ε_p	$1 - \varepsilon_p$	0	0	$cv = N$

TABLE V
CPD TABLE FOR SERIAL CONNECTION NODE MAPPED TO LOGIC AND EXECUTION OF MICRO-PROCESSOR

$P(x_i^t \| x_j^t, x_k^t)$	$x_i^t = 10$	$x_i^t = 01$	$x_i^t = 00$	$x_i^t = 11$
(00, zz)	0	$1 - \varepsilon_p$	ε_p	0
(01, z0)	0	$1 - \varepsilon_p$	ε_p	0
(01, z1)	0	ε_p	$1 - \varepsilon_p$	0
(10, 0z)	0	$1 - \varepsilon_p$	ε_p	0
(10, 1z)	ε_p	0	0	$1 - \varepsilon_p$
(11, 00)	0	$1 - \varepsilon_p$	ε_p	0
(11, 01)	0	ε_p	0	$1 - \varepsilon_p$
(11, 10)	ε_p	0	0	$1 - \varepsilon_p$
(11, 11)	$1 - \varepsilon_p$	0	0	ε_p

The *connection* node is used to denote the relation of two logic blocks, similar to the semantics of circuits AND and OR, and can also be mapped to a logic execution of the PLC microprocessor. Similar to the *special* node of **timer N** instruction, it will also has two parent nodes (x_j^t, x_k^t) to represent the input signal of this logic connection (x_i^t). For example, the node x_3, x_2 are connected in serial by the node x_4 in Fig. 2, which are traced back to the block $OR(s_3)$, signal $S(s_2)$ and block $AND(s_4)$, respectively. As described in the *contact* node for memory, the error propagation and cancel rule must be obeyed in the definition of conditional probability distribution. Let us consider two situations. We assume that the values of x_2 and x_3 are right, taking value 11 and 11, respectively. When the processor works well with probability ε_p, the output of x_4 will be right, taking the value 11. When the processor turns out to be failure with probability $1 - \varepsilon_p$, the output generates an error taking the value 10. In the other situation, we assume that the values of x_2 and x_3 are wrong, taking value 01 and 01, respectively. Then, when the processor works well with probability ε_p, the output of x_4 will propagate the error to the node x_5 and x_6, taking the value 01. When the processor happens to be failed with probability $1 - \varepsilon_p$, the errors of x_2 and x_3 will be canceled, taking the value 00. Other situations can be explained in the same way and are presented in the compressive table V.

Then, possible predictive and diagnostic inferences about the reliability can be evaluated. For example, when all the reliability of hardware components are given, the total reliability of the system can be predicted. Many algorithms have been invented for the inferences, among which message passing is comparatively popular. Based on the inference algorithms, many software tools such as CODA and Nertica [19], [20] have been implemented. In our experiments, we use Nertica. After reliability of each ladder rung R_i is obtained by applying the inference algorithm and the software, we can get the final reliability function for the whole PLC system as $\prod_{i=1}^{i \leq n}(R_i)$. R_i is defined on the probability of the final output node, and equals to $P(00) + P(11)$, as described in those conditional probabilistic distribution tables.

V. EXPERIMENT RESULT

First, the reliability evaluation of a small motor control system is presented to show how RRM works. Then, We conduct some experiments on the reliability evaluation of more complex industry systems to validate the accuracy and the scalability of RRM model. For comparison, we evaluate the reliability of those systems with the original RBD method, FT method, and with our previous HRM [15] method considering dependencies in a single time slice. Finally, reliability collected from simulations on the real systems are used to confirm the correctness. The monte carlo simulation framework is based on fault injection of the GX Simulator6-C of Mitsubishi [21], [22]. The values of these parameters are set according to the description in [17], [23]. We embed the sampling error probability and the processing deviation probability into the monte carlo simulator to collect the reliability of PLC systems. Test benches are also generated to collect output. Each application is simulated for 10000 times with random generated inputs. The average value is the final result of simulation reliability value.

The small motor control system consists of an input module connected with two sensors to sample the move instruction and stop instructions, a latch memory to store the temporal variable that will be used as the input of the next period, a processor to process those three inputs according to the embedded ladder program shown in Fig. 1, and an output module connected to an actuator motor and the latch memory. As shown in Fig. 1, the output value of the variable F will be stored and propagated to the next period as an input. The order of RRM model is supported for two cycles, because we just need two cycles to finish the application of starting and stoping the motor. The RRM model is shown in Fig. 4. We need to initiate the reliability of each component to construct the quantitative part of RRM. We set the possible initiation for each component as: Sensor(0.95), Memory(0.99), Processor(0.98), Actuator(0.99).

Then, we can use these parameters to initiate the conditional probabilistic distribution for each node. For example, the initialization of node s_4 in Fig. 4 is presented in the table VI. After we input the graph structure in Fig 4 and those cpd tables into the software Nertica, we can get the final reliability of the motor system. The reliability is 0.91. If we do not consider the high order temporal correlations caused by the run-time execution logic of the ladder program, the reliability for the BN in Fig. 2 is 0.67. The simulation result is 0.83. It is clear that RRM model is more closed to the simulation results.

TABLE VI
CPD FOR NODE s_4 IN FIG 4.

$P(x_i^t\|x_j^t, x_k^t)$	$x_i^t = 10$	$x_i^t = 01$	$x_i^t = 00$	$x_i^t = 11$
(00, zz)	0	$1-0.98$	0.98	0
(01, z0)	0	$1-0.98$	0.98	0
(01, z1)	0	0.98	$1-0.98$	0
(10, 0z)	0	$1-0.98$	0.98	0
(10, 1z)	0.98	0	0	$1-0.98$
(11, 00)	0	$1-0.98$	0.98	0
(11, 01)	0	0.98	0	$1-0.98$
(11, 10)	0.98	0	0	$1-0.98$
(11, 11)	$1-0.98$	0	0	0.98

Our approach is applied in some complex applications via more experiments. The first is an actual manufacturing system originally published in [24]. It consists of four pistons (A, B, C, D) to load parts from a machine table. There are four different embedded ladder programs used to control hardware component deployment to finish the application $[A+, B+, A-C+, B-C-]$, where +(-) means moving left(right). The four ladder programs are presented in [15]. For the application we mentioned above, we need to unroll the graph structure of each ladder program for four cycle of periods. We set the failure probabilities of the system components as: sensor(0.95), Memory(0.99), PLC microprocessor (0.98), pistons(0.95). We also change the reliability probability of the processor and keep the others the same. Then, the reliability probabilities of the system corresponding to the four ladder programs are listed in the table VII.

From the six column of the table, we can see that for the same application, the final reliability of the system are different when it is controlled by different ladder programs. The simulation time for the four ladder programs are 2813, 3021, 3527, and 3841 minutes, respectively. That means even with the same hardware distributions and reliability for each component, more complex ladder program will lead to lower system reliability. Because more complex ladder program will engender more complex arrangements of the logic executions of system components, and more complex temporal dependencies of between two periods.

TABLE VII
RELIABILITY FOR THE PISTON SYSTEM CONTROLLED BY THE LADDER 1, LADDER 2, LADDER 3, AND LADDER 4, RESPECTIVELY

processor ε	RBD	FT	BN(HRM)	RRM	simulation
0.99	99.54%	99.54%	99.42%	99.33%	99.31%
0.98	99.43%	99.43%	99.27%	99.20%	99.18%
0.97	99.28%	99.28%	99.02%	98.93%	98.95%
0.96	99.06%	99.06%	98.66%	98.62%	98.59%
0.95	97.90%	97.90%	97.28%	97.18%	97.15%
0.94	95.39%	95.39%	94.63%	94.41%	94.36%
0.93	93.88%	93.88%	92.73%	92.36%	92.41%
0.92	91.97%	91.97%	90.74%	90.46%	90.39%
0.91	91.29%	91.29%	88.63%	88.36%	88.28%
0.90	90.76%	90.76%	85.25%	85.19%	85.10%
0.99	99.54%	99.54%	99.31%	99.27%	99.26%
0.98	99.43%	99.43%	99.19%	99.10%	99.08%
0.97	99.28%	99.28%	98.88%	98.76%	98.77%
0.96	99.06%	99.06%	98.47%	98.38%	98.36%
0.95	97.90%	97.90%	96.06%	95.88%	95.86%
0.94	95.39%	95.39%	93.53%	93.26%	93.22%
0.93	93.88%	93.88%	91.68%	91.41%	91.37%
0.92	91.97%	91.97%	88.74%	88.14%	88.08%
0.91	91.29%	91.29%	86.49%	86.26%	86.19%
0.90	90.76%	90.76%	83.98%	83.31%	83.33%
0.99	99.54%	99.54%	99.13%	99.06%	99.09%
0.98	99.43%	99.43%	98.91%	98.82%	98.80%
0.97	99.28%	99.28%	98.29%	98.19%	98.17%
0.96	99.06%	99.06%	97.54%	97.20%	97.19%
0.95	97.90%	97.90%	95.33%	95.22%	95.20%
0.94	95.39%	95.39%	92.97%	92.74%	92.69%
0.93	93.88%	93.88%	90.48%	90.32%	90.28%
0.92	91.97%	91.97%	88.03%	87.84%	87.80%
0.91	91.29%	91.29%	85.34%	84.99%	85.08%
0.90	90.76%	90.76%	82.83%	82.61%	82.49%
0.99	99.54%	99.54%	99.01%	98.95%	98.94%
0.98	99.43%	99.43%	98.78%	98.66%	98.70%
0.97	99.28%	99.28%	98.02%	97.95%	97.91%
0.96	99.06%	99.06%	97.29%	97.17%	97.16%
0.95	97.90%	97.90%	95.01%	94.89%	94.88%
0.94	95.39%	95.39%	92.65%	92.50%	92.48%
0.93	93.88%	93.88%	89.73%	89.43%	89.39%
0.92	91.97%	91.97%	87.63%	87.41%	87.36%
0.91	91.29%	91.29%	84.59%	84.25%	84.20%
0.90	90.76%	90.76%	81.43%	80.90%	81.07%

From the second and third column of the table, we can see that reliability obtained by the original component-based FT and RBD methods, are the same for the four embedded ladder programs. The points of the red line in the four figures are with the same value for each reliability of processor. This is not consistent to the simulation results. They mainly consider the distribution of the hardware components, which are the same for the four ladder programs. The signal dependencies among them, and the complex relations caused by the execution logic of the ladder program are ignored. Besides, the values are all far from the simulation results. The HRM model [15] improves the accuracy to some extent, as demonstrated in the fourth column. It captures the semantic of ladder program in a single time slice, and the run-time execution is not considered. Hence, the temporal relations between two time slices are ignored. The RRM model is more accurate. As shown in fifth column of the table, the error between the RRM results and the simulation results is less than 0.2%. The results gathered by way of RRM is more close to the run time station, because the reliability probability of the single system component and the complex execution logic of the ladder program are captured by the graph structure of RRM model. Furthermore, for the four ladder programs, the total time for RRM construction of our algorithm and the evidences propagation is within 0.5 second.

The second complex systems is a control system for a secondary clarifier scum removal [25]. It has been installed in the Deer Island Water Pollution Treatment Facility near Boston, MA. The system is controlled by a ladder diagram which consists of 30 ladder rungs, which need to be unrolled for ten cycles of periods. The third complex system is a double door control system [26]. It has been installed in the Lingshan stage in Qingdao city, China. It is controlled by a ladder diagram which consists of 27 ladder rungs, which need to be unrolled for eight cycles of periods. We set the reliability of the system components similar to the piston system. Then, the reliability of the two systems are presented in the table VIII. From the table, the RRM is accurate, even with these complex applications. The error between the simulations results and the RRM based results is within 0.2%, and time is within 1 second. The simulation time is thousands of minutes.

TABLE VIII
RELIABILITY FOR THE COMPLEX CLARIFIER DOUBLE DOOR CONTROL SYSTEMAND CLARIFIER SCUM REMOVAL SYSTEM, RESPECTIVELY.

processor ε	RBD	FT	BN(HRM)	RRM	simulation
0.99	98.47%	98.47%	98.23%	98.18%	98.17%
0.98	98.34%	98.34%	97.94%	97.90%	97.89%
0.97	97.87%	97.87%	97.30%	97.25%	97.24%
0.96	97.06%	97.06%	96.56%	96.52%	96.49%
0.95	96.13%	96.13%	95.24%	95.18%	95.17%
0.94	93.88%	93.88%	91.46%	91.40%	91.38%
0.93	92.33%	92.33%	89.17%	88.96%	88.98%
0.92	90.87%	90.87%	86.89%	86.61%	86.58%
0.91	88.68%	88.68%	84.20%	83.83%	83.79%
0.90	86.89%	86.89%	81.28%	80.88%	80.77%
0.99	98.43%	98.43%	98.16%	98.14%	98.13%
0.98	98.31%	98.31%	97.80%	97.76%	97.75%
0.97	97.29%	97.29%	96.64%	96.58%	96.57%
0.96	96.88%	96.88%	96.09%	95.99%	96.02%
0.95	95.41%	95.41%	93.63%	93.54%	93.51%
0.94	94.59%	94.59%	91.17%	91.00%	91.04%
0.93	92.38%	92.38%	88.19%	87.98%	87.88%
0.92	90.11%	90.11%	86.11%	85.80%	85.74%
0.91	88.69%	88.69%	83.22%	82.83%	82.78%
0.90	85.81%	85.81%	79.86%	79.55%	79.42%

From these experiments, we can see that the simulation is the gold standard, but we need to set up the simulation environment for each application separately very hard with programming efforts of the test bench. Besides, the simulation time is 2813, 4398, and 3524 minutes for the three complex applications, respectively; the runtime of RRM is 0.49, 0.92, and 0.84 seconds, respectively; the runtime of BN based HRM is 0.12, 0.23, 0.19 seconds, respectively; and the runtime of RBD and FT is 0.03, 0,06, 0.05 seconds, respectively. The simulation is the most correct for the practice, but for the big applications are even more time-consuming and resource-consuming. The complexity of the graph construction algorithm is $O(m \cdot n_{max} \cdot T)$, where m is the number of ladder rungs in the ladder program, n_{max} is the number of contacts in the longest ladder rung, and the T is the number of unrolled time slices. The time complexity of the exact inference in Nertica is $O(m \cdot c_n \cdot 4^{|c_{max}|})$ [27], [28], where c_n is the number of cliques in the compiled junction tree of the original RRM model, and the $|c_{max}|$ is the number of contacts in the largest clique. Although the runtime of RRM is longer than FT, RBD, BN and HRM, the time is tolerant compared to simulation and the accuracy is higher. From the complexity and accuracy analysis above, it is safe to conclude that the proposed RRM model is closer to the simulation results than that of the original component-based FT, RBD and HRM framework with an acceptable calculation time consumption.

VI. Conclusion

In this paper, we propose a run-time reliability model, named RRM, to handle spatial and higher order temporal dependencies among system components of the PLC system, especially for the dependencies caused by the execution logic of the embedded ladder program. With the proposed translation algorithm, the execution logic of the embedded program is mapped to RRM, which is formally proved to be a dynamic bayesian network. According to semantics of RRM nodes, conditional probability distribution tables are defined to calculate reliability. All nodes can be mapped to corresponding hardware components through the conditional probabilistic distribution table initialization. Through these two mapping processes, both the execution logic of the embedded control software and the hardware components are captured. As a result, our approach brings convenience for predictive inferences about the reliability of PLC systems.

References

[1] M. Villani, M. Tursini, G. Fabri, and L. Castellini, "High reliability permanent magnet brushless motor drive for aircraft application," *IEEE Trans. Ind. Electron.*, vol. 59, no. 5, pp. 2073–2081, 2012.

[2] P. Jalote, B. Murphy, and V. S. Sharma, "Post-release reliability growth in software products," *ACM Transaction on Software Engineering and Method.*, vol. 17, no. 4, p. 17, 2008.

[3] X. Yu and A. Khambadkone, "Reliability analysis and cost optimization of parallel inverter system," *IEEE Trans. Ind. Electron.*, no. 99, pp. 1–9, 2011.

[4] F. Chan and H. Calleja, "Reliability estimation of three single-phase topologies in grid-connected pv systems," *IEEE Trans. Ind. Electron.*, vol. 58, no. 7, pp. 2683–2689, 2011.

[5] G. Petrone, G. Spagnuolo, R. Teodorescu, M. Veerachary, and M. Vitelli, "Reliability issues in photovoltaic power processing systems," *IEEE Trans. Ind. Electron.*, vol. 55, no. 7, pp. 2569–2580, 2008.

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

[7] H. Guo and X. Yang, "A simple reliability block diagram method for safety integrity verification," *Reliability Engineering & System Safety*, vol. 92, no. 9, pp. 1267–1273, 2007.

[8] 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.

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

[10] 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.

[11] J. Dugan, S. Bavuso, and M. Boyd, "Dynamic fault-tree models for fault-tolerant computer systems," *IEEE Trans. Rel.*, vol. 41, no. 3, pp. 363–377, 1992.

[12] 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.

[13] H. Zhang., Y. Jiang., G. Yang., W. N. Hung., and J. Sun, "New strategies for reliability analysis of programmable logic controllers," *Mathematical and Computer Modelling*, vol. 55, no. 9, pp. 1916–1931, 2012.

[14] H. Zhang., Y. Jiang., X. Song., W. N. Hung., M. Gu., and J. Sun, "Symbolic analysis of plc systems," *IEEE Trans. Comput.*, vol. 90, no. 90, pp. 1–15, 2013.

[15] Y. Jiang., H. Zhang., X. Jiao., W. N. Hung., M. Gu., and J. Sun, "Bayesian network based reliability analysis of plc systems," *IEEE Trans. Ind. Electron.*, vol. 60, no. 11, pp. 5325–5336, 2013.

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

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

[18] Y. Jiang, "Formal proof of mapping rrm to dbn," *https://sites.google.com/site/jiangyu198964/home*.

[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] MITSUBISHI, "Gx simulator6-c," *http://www.filecrop.com/Mitsubishi-GX-simulator.html*, vol. 1, 2013.

[22] M. MELSOFT, "Gx simulator version 6: Operation maintenance programming," *http://www.filecrop.com/Mitsubishi-GX-simulator.html*, vol. 1, 2013.

[23] L. Portinale and A. Bobbio, "Bayesian networks for dependability analysis: an application to digital control reliability," in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, ser. UAI'99. ACM, 2010, pp. 551–558.

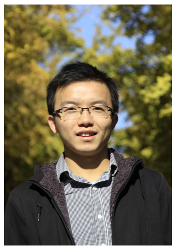
[24] K. Venkatesh, M. Zhou, and R. J. Caudill, "Comparing ladder logic diagrams and petri nets for sequence controller design through a discrete manufacturing system," *IEEE Trans. Ind. Electron.*, vol. 41, no. 6, pp. 611–619, December 1994.

[25] M. Zhou and E. Twiss, "Design of industrial automated systems via relay ladder logic programming and petri nets," *IEEE Trans. Syst. Man Cybern. C, Appl. Rev.*, vol. 28, no. 1, pp. 137–150, 1998.

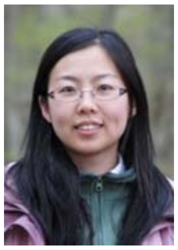
[26] R. Wang, X. Song, J. Zhu, and M. Gu, "Formal modeling and synthesis of programmable logic controllers," *Computers in Industry*, vol. 62, no. 1, pp. 23–31, 2011.

[27] J. Pearl, "Probabilistic reasoning in intelligent systems: networks of plausible inference," *Artificial intelligence*, pp. 1–288, 1988.

[28] K. P. Murphy, "Dynamic bayesian networks: representation, inference and learning," Ph.D. dissertation, University of California, 2002.

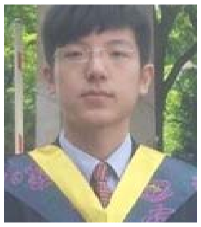


Yu Jiang received the BS degree in software engineering from Beijing University of post and telecommunication, beijing, China, in 2010. He is currently studying for the PhD degree in computer science from Tsinghua University, Beijing, China. His current research interests include domain specific modeling, formal verification and their applications in embedded systems.



Hehua Zhang received the BS and MS degree in computer science from Jilin University, Changchun, China, in 2001 and 2004, respectively. She received the PhD degrees in computer science from Tsinghua University, Beijing, China, in 2010. She is currently a lecturer in the School of Software at Tsinghua University. Her current research interests include domain specific modeling, formal verification and their applications in embedded systems.

Xiaoyu Song received the PhD degree from the University of Pisa, Italy, 1991. In 1999, he joined the faculty at Portland State University. He is currently a professor in the Department of Electrical & Computer Engineering at Portland State University, Oregon. His current research interests include formal methods, design automation, embedded system design, and emerging technologies.



Han Liu is a received the BS degree in software engineering from Beijing University of post and telecommunication, beijing, China, in 2012. He is currently studying for the PhD degree in software engineering from Tsinghua University, Beijing, China. His current research interests include domain specific modeling, formal verification and their applications in embedded systems.

William N.N. Hung received the BS and MS degrees in electrical and computer engineering from the University of Texas at Austin in 1994 and 1997, respectively. He received the PhD degree in electrical and computer engineering from Portland State University, Oregon, in 2002. His research interests include constraint solving, logic synthesis, physical design, formal methods, combinatorial optimization, nanotechnology, and quantum computing.

Ming Gu received the BS degree in computer science from the National University of Defence Technology, Changsha, China, in 1984, and the MS degree in computer science from the Chinese Academy of Science at Shengyang in 1986. Since 1993, she has been working as a professor in Tsinghua University. Her research interests include formal methods, middleware technology, and distributed applications.

Jiaguang Sun received the BS degree in automation science from Tsinghua University in 1970. He is currently a professor in Tsinghua University. He is dedicated in teaching and R&D activities in computer graphics, computer-aided design, formal verification of software, and system architecture. He is currently the director of the School of Information Science & Technology and the School of Software in Tsinghua University.