Design-to-test: an approach to enhance testability of programmable controllers for critical systems – two case studies

Canlong Ma & Julien Provost
Assistant Professorship for Safe Embedded Systems, Faculty of Mechanical Engineering
Technische Universität München, Garching bei München, Germany

ABSTRACT: This paper presents a continuation of researches on design-to-test (DTT) approach, which enhances the testability of programmable controllers for critical systems, where the specifications and implementations are modeled as finite state machines. Firstly, the testing problems in the field of programmable controllers and critical systems are defined. To solve these problems, the DTT approach takes testing performance into consideration during the design of specification. Then, the DTT approach is applied on two representative industrial case studies from previously published works.

1 INTRODUCTION

Critical systems are technical or socio-technical systems, where failures can result in significant losses of life, serious environmental damages or high economic costs (Sommerville 2007). According to this definition, many industrial applications are critical systems, therefore a series of international standards such as IEC 61508 (general industry), ISO 26262 (automotive) and DO-178B (avionics) have been established to formulate the functional safety requirements and development guidelines for industrial systems. The concept of Safety integrity level (SIL) is defined to measure the dependability and safety requirements of a system (IEC61508 2010). According to Part-3 in IEC61508 (2010), complete testing is recommended (for all SILs) and highly recommended (for high SILs). For critical systems with highest SILs such as avionic equipment and some automotive applications, a complete testing is mandatorily required.

Programmable controllers are widely used in safety-critical industries. They are designed to withstand harsh industrial environments. Their programming languages are standardized and easy to understand. Besides, their cyclic execution mode enables them to fulfill 'hard' real-time requirement (Bolton 2006).

However, the beneficial characteristics of programmable controller also bring some testing issues. Meanwhile, the complexity of automations systems nowadays is continuously increasing (Rösch, Ulewicz, Provost, & Vogel-heuser 2015), and as mentioned above critical systems are always demanding high requirement of testings such as complete coverage. To cope with all these challenges, recently, a design-to-test (DTT) approach has been proposed (Ma & Provost 2015). The DTT approach aims at making a bit more efforts in design, in order to gain a huge benefit during testing.

In this paper, the DTT approach is further improved and systematically presented. Then, the DTT approach is applied on two industrial case studies from previously published works. The first one is a wind farm operation and maintenance system, which is a typical safety critical system because it is in an extreme environment and may cause enormous losses. The second one is an automated manufacturing system, which can be a general reference for large scale complex systems.

The paper is organized as follows: Section 2, as a background, presents the syntax and graphics of finite state machine used in this paper, as well as black-box testing for programmable controllers. Section 3 provides a detailed introduction of testing issues covering single-input-change (SIC) -testability, observability and controllability problems, and also presents the DTT approach, including T-guard, O-action and C-guard methods. In section 4 and 5, two case studies are introduced, described, analyzed and modified by the DTT approach. Finally, section 6 presents the conclusions.

2 BACKGROUND

2.1 *Finite state machine with boolean signals*

Finite state machine (FSM) is a bunch of modeling languages, which is firstly developed in academi-

cal fields such as mathematics and computer science. Benefit from the abundant theoretical results from these fields, FSM has also been widely introduced into industrial applications such as programmable controllers to formalize different kinds of practical processes.

Moore machine is a basic and popular form of FSM. Many other FSMs can be automatically or easily transformed into Moore machines, e.g. Mealy machine (Lee & Seshia 2011) and Petri net (Chang & Huang 1990).

In this paper, a specification model is supposed to be Moore machine with boolean signal inputs, which is presented as a 6-tuple $(S, s_{init}, \Omega, \Lambda, \delta, \lambda)$ where:

- S is a finite set of states
- s_{init} is the initial state, $s_{init} \in S$
- Ω is a finite set of input signals
- Λ is a finite set of output actions
- $\delta : S \times \Omega \to S$ is the transition function, that maps the current state and the input signals to the next state
- $\lambda : S \to \Lambda$ is the output function, that maps the states to their corresponding output actions

In the paper, Moore machines are also presented with graphics. As shown in Fig. 1, a state s is drawn as a circle / an ellipse (e.g. S1 is an initial state), inside which its corresponding action λ is written (e.g. 'A3' in S3). A transition δ is represented by lines with arrows, upon which the corresponding guard is placed (e.g. '$\neg a \cdot b$' from S1 to S2).

A state can either have an external observable action, e.g. 'A3' in S3 and 'A4' in S4, or no observable action, e.g. '$\emptyset$' in S1 and S5. Besides, a state can also be given an internal control action, e.g. 'XS2' in S2 and 'XS6' in S6. The internal control actions are used as boolean transition guards, e.g. when the state S2 is activated, 'XS2' is assigned the value '1', then the transition from S4 to S5 can be fired.

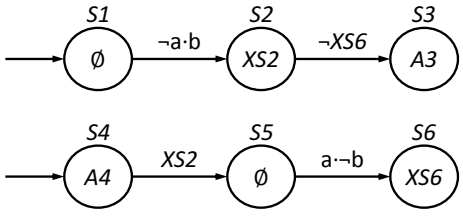

Figure 1: A simple example of Moore machine with boolean signals

Applying the DTT approach, a set of individual Moore machines are parallel composed with stability research. In the composed machine, a location means a combination of states from individual machines. An evolutions is a function that maps the current location to a next one.

2.2 *Black-box conformance testing*

Software testing is a process, or a series of processes, designed to make sure computer codes perform what they were designed to do and conversely, that they not do anything unintended, as stated in Myers, Badgett, & Sandler (2011).

Black-box testing is an important testing strategy, which doesn't concern the internal structure but only concentrates on the input-output behavior of to-be-tested software products (Pressman 2010). It is widely applied in later phases of testings, where the internal structures are not easily visible.

Conformance testing is aimed at checking whether an implementation, seen as a black-box with inputs / outputs, behaves correctly with respect to its specification (Cheriaux, Picci, Provost, & Faure 2010).

In this paper, the to-be-tested objects are implemented programmable controllers and the specifications are FSM models. The goal of this testing is to verify if a controller has the same behaviors as its designed specification.

A test unit is made up of 3 stages, each contains a few steps:

- Before testing:
  - initialize both the specification model and controller
  - control the model to a certain state by inputting a signal sequence and apply the same sequence to the controller
  - verify that the model and controller are in the same state
- During testing:
  - apply the testing input signals to both the model and controller
- After testing:
  - observe the states in the model and controller
  - compare the results and record

2.3 *Observability & Controllability*

To realize the third testing step, it needs to be identified which state is currently active. Recent researches in this field e.g. (Provost, Roussel, & Faure 2014) and (Guignard & Faure 2014) were carried out under a strong assumption: all states are identifiable by observing the emitted outputs at time t. However, if this is not satisfied, problems of state identification arise (Lee & Yannakakis 1996). This issue is named observability problem.

For Moore machines, output actions are linked to respective states. As in the domain of black-box testing all internal structures (states and transitions) are

invisible to testers. If some states have the same output actions, they can not directly be distinguished from each other. This state identification problem might be solved by searching for a distinguishing sequence. However, the existence of such sequences is not always guaranteed. And if they exist, they might be of exponential length (Lee & Yannakakis 1996).

Similarly, a FSM model should be brought to a specific state to execute the first testing step. A system is said to be controllable at time t_0, if it is possible by means of a control vector to transfer the system from any initial state $x(t_0)$ to any other state in a finite interval of time (Ogata 2012)(Page 675).

The controllability problem can be solved in two steps: identify the final state after some operations and then move to the desired state. The first step is solved by searching for a homing or synchronizing sequence. Both methods determine the final state of a machine after applying the sequence (Lee & Yannakakis 1996). After that, the machine can be brought to any specific state by applying the corresponding input sequences derived from the specification model in the design phase. Of course, this process is conditioned on fulfillment of the previously mentioned observability requirement.

3 DESIGN-TO-TEST (DTT) APPROACH

As testing reference, the specification i.e. a FSM model will be controlled, observed and tested as well as the controller. However, in practice due to some issues a FSM itself might not be (well) testable, e.g. the upper system in Fig. 2.

To solve these problems in testing phase, the design-to-test approach modifies the specification models in design phase, which are implemented as executable code in programmable controllers later on. Respectively, DTT approach solves the single-input-change (SIC) -testability, observability and controllability problems by modifying the models with T-guards, O-actions and C-guards, i.e. the lower system in Fig. 2.

3.1 *SIC-Testability & T-guard method*

Due to the cyclic execution mode, programmable controllers might read biased inputs when several input signals change the values at the same time (see the upper right diagram in Fig. 3). To exclude this faulty behavior, a complete testing should only contain single-input-change (SIC) sequences. This issue is named as SIC-testability problem (Provost, Roussel, & Faure 2014).

The T-guard method is developed to transform multiple-input-changes (MICs) into SICs. As presented in the lower sketch of Fig. 3, for all the MIC related transitions, a T-guard will be added to their guards. Before a MIC happens, the T-guard is set the value '0', so all outgoing transitions are frozen from

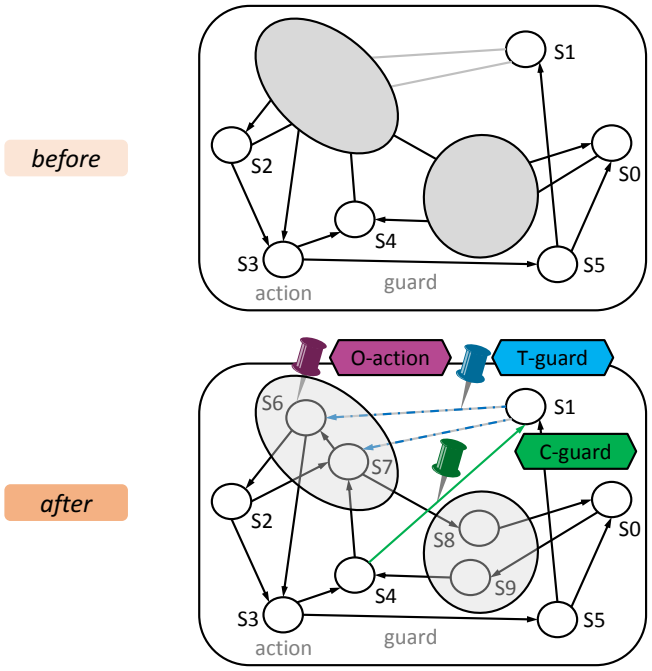

Figure 2: Basic idea of DTT approach: adding T-guards, O-actions and C-guards to modify the initial specification models

being fired. After the MICs are stabilized, the T-guard is set the value '1' again, now only the correct outgoing transition will be fired.

Through this way, any non-trivial specification model can be then fully SIC-testable.

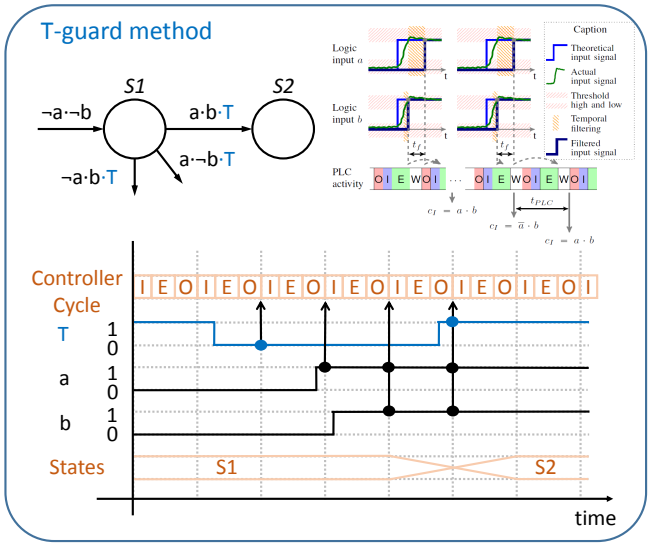

Figure 3: T-guard method. Upper left: an excerpt of Moore machine that has SIC-testability problem and modified by T-guards; Upper right: the physical cause of SIC-testability problem in programmable controllers; Bottom: the way T-guard works to transform a MIC into SIC

3.2 *Observability & O-action method*

As introduced in 2.3, so far there exists no method that can completely solve the observability problem.

Fig. 4 presents the O-action method to handle this issue. In the composed machine, if several locations have the same output actions, the states that compose these locations will be analyzed. The involved states that have the same output actions will be added a set of O-actions.

Since any O-action can be set the value ‘True’ or ‘False’, the minimum number needed to distinguish n states is $\lceil \log_2(n) \rceil$. After all the locations are checked and the related states are modified with O-actions, the complete model is then fully observable.

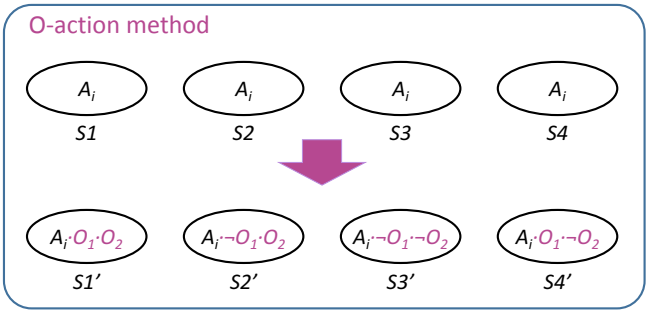

Figure 4: O-action method: the figure shows an example of four states that initially have the same action, and modified by O-actions

3.3 *Controllability & C-guard method*

Unlike the observability problem, the controllability problem can be solved by existing methods. However, those methods will cost a long control time for complex systems, which increases testing execution time. How to rapidly control the system from an arbitrary state to another, is the new task of the controllability problem.

An example of the controllability problem, i.e. an excerpt of a large scale FSM model, is presented in Fig. 5, where the states constitute a long chain. The tester has to pass through a long path from a state in ‘i-group’ to a state in ‘j-group’ or vice versa.

In order to achieve a better controllability, i.e. shorter paths between any couple of states, DTT approach adds C-guards to the models. C-guards builds new transitions, i.e. shortcuts, between some long distance couple of states. It is noteworthy that after adding one C-guards, the distance of some other states has been also indirectly shorten. So the DTT approach uses an algorithm to calculate a globally optimal set of C-guards to fulfill the controllability requirement of testers.

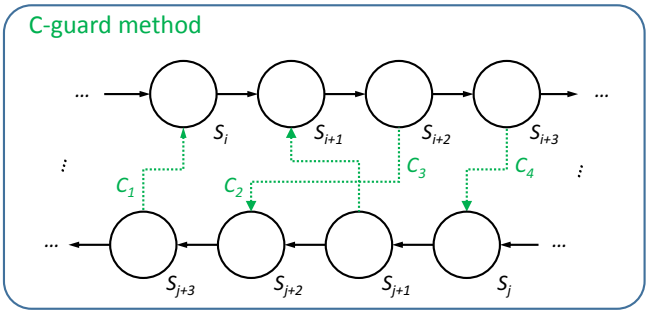

Figure 5: C-guard method: the figure shows how C-guards create new transitions and shorten the path costs between states

3.4 *Design, Testing & normal execution*

Applying the DTT approach in the design phase, the specification models are added T-guards, O-actions and C-guards according to testing requirements on SIC-testability, observability and controllability.

In the testing phase, T-guards are set the value ‘0’ for MIC test steps and set back the value ‘1’ after the MIC is completed. O-actions are assigned the values in accordance with the O-action method calculations. C-guards are set the value ‘1’ to enable the shortcuts.

After testing, T-guards will be set the value ‘1’, so they are always ‘True’ and will not affect the original transition guards. C-guards will be set the value ‘0’, so the new transitions cannot be fired. O-actions are just outputs, so they can anyway not affect the transition behavior of the FSM models. In summary, DTT approach will not change the nominal behavior of the specification systems during normal execution.

4 CASE STUDY 1: A WIND FARM OPERATION AND MAINTENANCE SYSTEM

4.1 *Description of system*

The first case study is a wind farm operation and maintenance system taken from Byon, Perez, Ding, & Ntaimo (2010). Fig. 6 shows the basic components of a wind turbine. A wind turbine is made up of a tower, two or three-bladed rotors, and a nacelle which houses several critical components such as the drive train, gearbox, generator, and the electrical system.

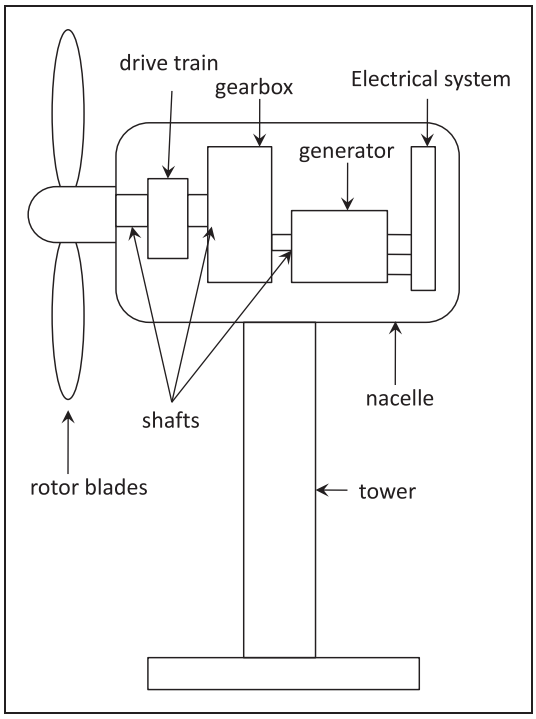

Figure 6: CASE-1: Wind turbine components

According to the research, the critical components will degrade their functions based on a constant probability during a fixed time period. In scope of FSM modeling, four different groups of states are used

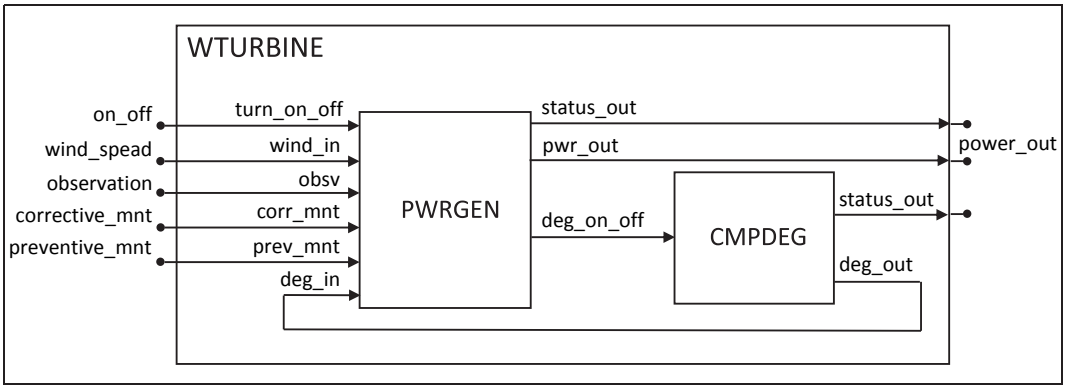

Figure 7: CASE-1: Wind turbine block diagram

to represent their statuses: 'Normal', 'Alert', 'Alarm' and 'Failed'.

The complete wind farm system contains a large number of subsystems. Two key subsystems of the Wind Turbine (WTURBINE) are chosen to be studied in this paper: Power Generator (PWRGEN) and Component Degradation (CMPDEG). The inputs, outputs and coupling relationship of PWRGEN and CMPDEG are displayed in Fig. 7.

4.2 *Modeling of system*

The two subsystems are modeled as Moore machines: PWRGEN in Fig. 8 and CMPDEG in Fig. 9. The two individual models contain 13 inputs and 8 outputs.

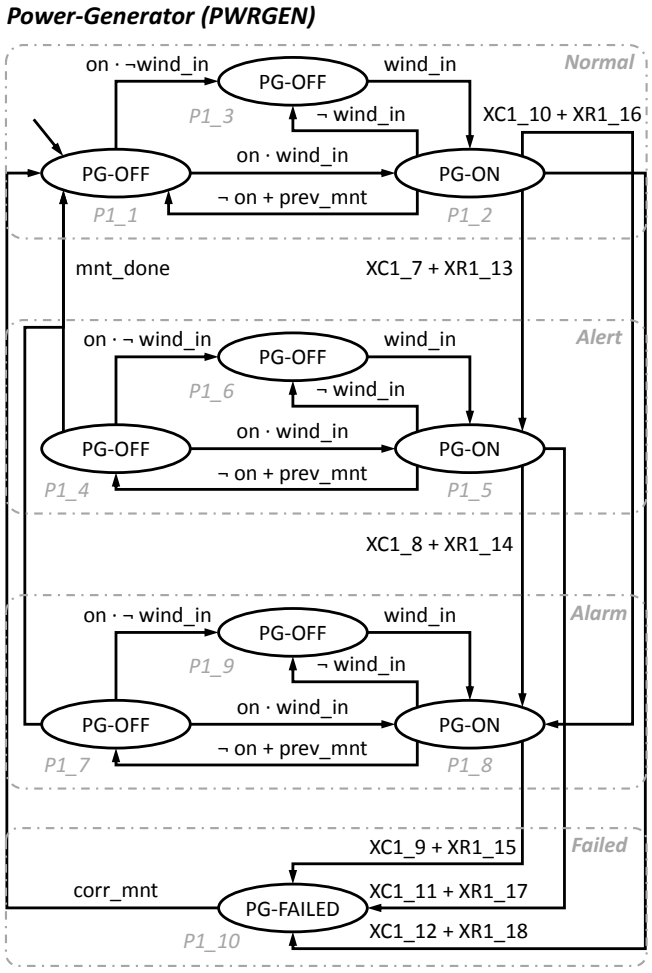

Figure 8: Model of subsystem: Power Generator

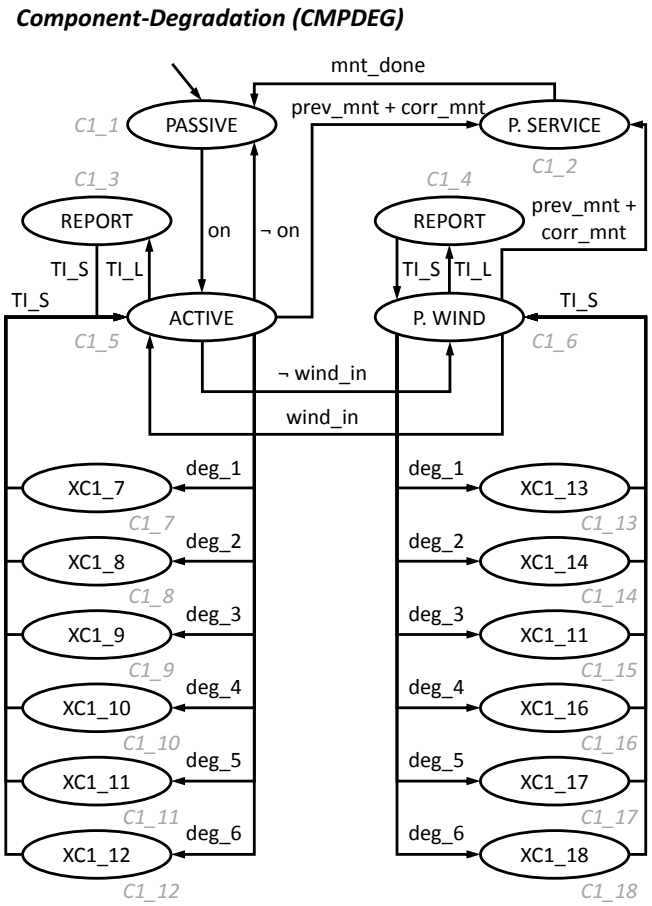

Figure 9: Models of subsystem: Component Degradation

PWRGEN is initialized in a 'PG-OFF' state. A state in PWRGEN has one of the observable actions: 'PG-OFF', 'PG-ON' or 'PG-FAILED'. Only when the switch is turned on ('on'), the wind speed is under the cutoff threshold ('wind-in'), and no preventive maintenance (prev-mnt) is executed, PWRGEN has the action 'PG-ON'. Otherwise it has the action 'PG-OFF'. Once a heavy degradation occurs, PWRGEN has the action 'PG-FAILED' and has to wait for a corrective maintenance ('corr-mnt') to return to the initial state.

From economic and technical aspects, a wind turbine cannot be frequently inspected, therefore the classification of 'Normal', 'Alert' and 'Alarm' are merely internally marked and cannot reflected in the observable actions, though they are of great importance from the maintenance viewpoint.

CMPDEG is initialized in a 'PASSIVE' state. A state in CMPDEG can have one of the actions: 'PASSIVE', 'ACTIVE', 'REPORT', 'P. SERVICE' (i.e. passive service), 'P. WIND' (i.e. passive wind) or '$\emptyset$' (i.e. empty).

When CMPDEG is in state C1-5 or C1-6, i.e. has the action 'ACTIVE' or 'P. WIND', it will regularly go to state C1-3 or C1-4, i.e. has the action 'REPORT'. The frequency that CMPDEG switch to / back from the action 'REPORT' is controlled by time interval long / short ('TI-L' / 'TI-S').

When a certain type of degradation (i.e. 'deg-1' to 'deg-6') occurs, CMPDEG will go to a certain state that will control the activation of different groups in the subsystem PWRGEN.

4.3 *Testing issues*

It can be observed that the states in PWRGEN and CMPDEG have complex transition relations with each other. Several inputs might change their values between two transitions. During testing, SIC-testability problem will occur when such changes happen.

When observing the individual models, it is easy to find that some states have the same output actions. For example, in PWRGEN, the states P1-1, P1-3, P1-4, P1-6, P1-7 and P1-9 all have the action 'PG-OFF', and in CMPDEG, 12 states don't have an observable action. This implies that, after composition, it is probable that there are some locations that have same actions, which leads to observability problem.

Since the states have complex transition relations and somehow 'strongly connected', it is hard to analyze manually whether this system would have controllability problem.

According to the above qualitative estimations, this system is problem-prone in testing.

4.4 *Applying DTT approach*

The composed machine of this system has 150 locations and 2117 stable evolutions.

Based upon qualitative analysis of the system done by DTT approach, 149 out of 150 locations in the composed machine contain non-SIC-testable parts. According to DTT approach, T-guards are added 58 out of the 59 transitions in the two subsystems.

In the composed machine, several locations share the same output actions. Calculated by DTT approach, 10 O-actions will be added to the two subsystems. With help of these O-actions, the three statuses 'Normal', 'Alert', 'Alarm' are now observable, which provides useful information to maintenance management.

The initial maximum of path cost between any couple of locations is 5. Although it is already not a bad result, DTT approach can help to reach a better performance. After adding 18, 21 or 25 C-guards, the maximum path cost is reduced respectively to 4, 3 or 2 steps.

Now the modified specification model achieves a full SIC-testability, a full observability and a better controllability for testing purpose.

5 CASE STUDY 2: AN AUTOMATED MANUFACTURING SYSTEM

5.1 *Description of system*

The second case study is an automated manufacturing system taken from Cabasino, Giua, & Seatzu (2009). Its layout is shown in Fig. 10.

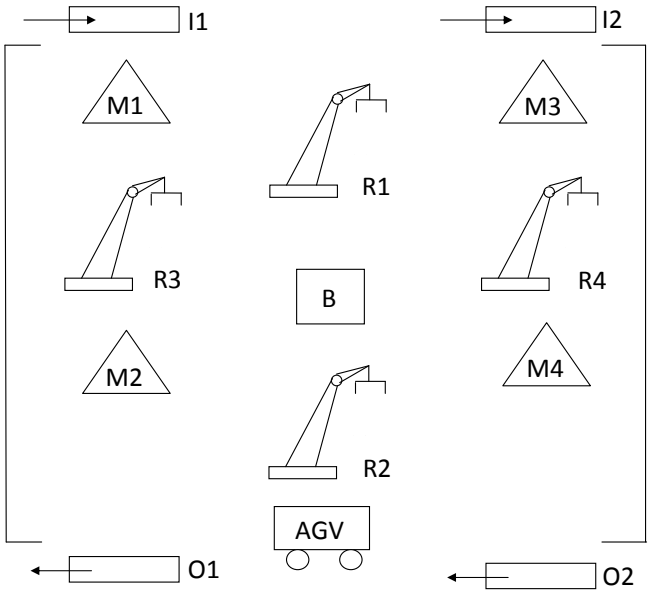

Figure 10: CASE-2: Layout of a manufacturing system

The system is made up of four machines (M1 to M4), four robots (R1 to R4), one AGV system (AGV), one buffer of finite capacity (B), two inputs of material pieces to be processed (I1 and I2) and two outputs for the processed pieces (O1 and O2). The two production lines produce two different kinds of final product.

In the upper left quarter of Fig. 10, robot R1 picks up a piece of raw material from input I1 when available and places it to machine M1. M1 begins its machining work, and when it finishes, its sensor gives a signal 'M1-finished'. If buffer B is not full at the moment, robot R3 carries the semi-manufactured product and places it to B.

In the lower left quarter of Fig. 10, robot R3 takes a semi-manufactured product from B when it is not empty and places it in machine M2. Similarly, a signal 'M2-finished' is sent after M2 has done its machining. Robot R2 then picks the final product and places to AGV, which transports the product to its corresponding output O1.

Noticing that each robot interacts with two machines and other parts such as input, output or the buffer, a controller has been set up for each robot to regulate its operations.

On the right side, the same kind of work is executed by R1, M3, R4, M4 and R2.

5.2 *Modeling of system*

In this paper, only the left half of the whole system (I1, M1, R1, R3, B, M2, R2, AGV and O1) is investigated due to two reasons:

1. Simplicity reason. The left and right halves of the system are symmetrical and have the same behavior, so the result of left part can also be applied to the right part.
2. State explosion. The current version of DTT approach has a space limitation for locations and evolutions, i.e. approximately 1000 locations and 10,000 evolutions. The composed machine of the complete case study is estimated to contain more than 10,000 locations and 1000,000 evolutions. This limitation is planed to be overcome in following research works.

In the rest part of this section, 'the system' is referred to the left part.

The system contains 10 subsystems: robot R1, controller C1, machine M1, robot R3, controller C3, buffer B, machine M2, robot R2 and controller C2 and AGV. Besides, 13 inputs and 9 outputs are specified in this system.

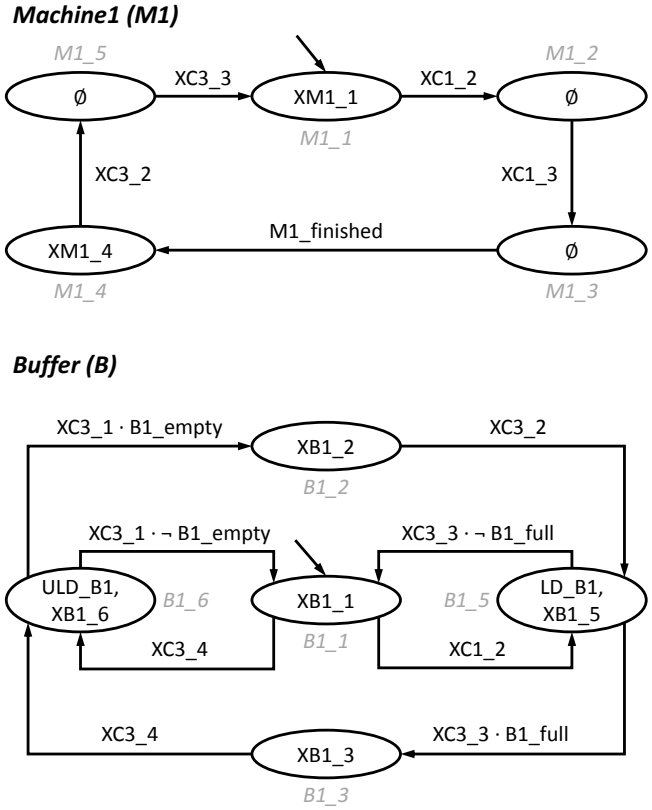

Figure 11: Models of subsystem: Machine & buffer

Considering the fact that every machine does the same kind of work, so does also every robot and controller. Again for simplicity reason the models for M1, B, R3 and C3 are selected as examples to be presented in Fig. 11 and Fig. 12.

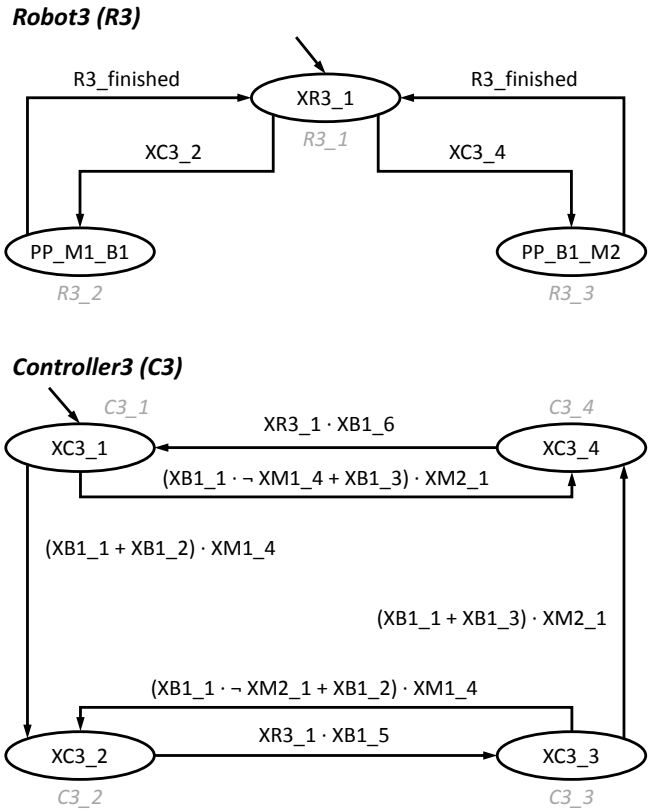

Figure 12: Models of subsystem: Robot & Controller

5.3 *Testing issues*

It is obvious that several parts in the system can be executed in parallel. So in the composed machine, some inputs can change for the same evolutions. In testing, such changes will lead to SIC-testability problems.

When observing the individual models, it is easy to find that some states have the same output actions. For example, in M1 the states M1-2, M1-3 and M1-5 don't have an observable action. Similar to the first case study, this will cause the observability problem.

Since the system contains 10 subsystems, in the composed machine, it is highly possible that many locations are not strongly or even not connected. When testing large scale parallel systems, controllability is an important performance factor.

According to the above qualitative estimation, this system is also problem-prone in testing.

5.4 *Applying DTT approach*

The composed machine of this system has 271 locations and 15089 stable evolutions.

Checked by the DTT approach, 260 out of 271 locations contain non-SIC-testable parts. Applying the T-guard method, a optimal feasible solution is obtained: T-guards are added to 6 transitions.

To solve the observability problem, 5 O-actions are added to the related states.

Checked by C-guard method, the maximum path cost between two locations is 'inf.', which means some locations are not reachable from some other states. This is normal with regard to specification for system function. However, from the view of testing,

especially for critical systems, a complete testing is often required. By using the C-guard method, this system can be fully reachable and testable.

After adding 13, 17 or 36 C-guards, this path cost is reduced respectively to 4, 3 or 2 steps.

Also, the modified specification model for this case study now achieves a full SIC-testability, a full observability and a better controllability for testing purpose.

6 CONCLUSIONS

This paper has presented the design-to-test (DTT) approach and the application on case studies.

The DTT approach was first proposed in (Ma & Provost 2015) and aims to improving the testability of programmable controllers, reducing the testing overhead with small amount of design overhead and also keeping the nominal behavior unchanged during normal execution.

Inputs of the DTT approach are specification modeled as Moore machines. By running the T-guard, O-action and C-guard methods, the specification models are modified to meet the requirements of full SIC-testability, full observability and better controllability.

The way to apply DTT approach and the advantageous results have been shown through two representative case studies in the field of critical systems.

Further works are aiming at an improved implementation of the DTT approach, which can handle systems of larger state space. This will be realized by a series of measures, e.g. by combining the controller specification models with plant behavioral models in order to consider only physically-feasible combinations of input signals.

REFERENCES

Bolton, W. (2006). *Programmable logic controllers* (4th ed.). Elsevier.

Byon, E., E. Perez, Y. Ding, & L. Ntaimo (2010). Simulation of wind farm operations and maintenance using discrete event system specification. *Transactions of the Society for Modelling and Simulation International 87*(12), 1093–1117.

Cabasino, M. P., A. Giua, & C. Seatzu (2009). Discrete Event Diagnosis Using Petri Nets. In *ICINCO09: 6th Int. Conf. on Informatics in Control, Automation and Robotics*, pp. 15–29.

Chang, C. K. & H. Huang (1990). On transforming Petri net model to Moore machine. In *14th Annual Int. Computer Software and Applications Conf.*, Number 052, Chicago, pp. 267—-272. IEEE.

Cheriaux, F., L. Picci, J. Provost, & J.-M. Faure (2010). Conformance test of logic controllers of critical systems from industrial specifications. In *European Conference on Safety and Reliability - ESREL 2010*, Rhodes, Greece, pp. paper—-308.

Guignard, A. & J.-M. Faure (2014). A Conformance Relation for Model-Based Testing of PLC. In *12th Int. Workshop on Discrete Event Systems*, Cachan, pp. 412–419.

IEC61508 (2010). *Functional safety of electrical / electronic / programmable electronic safety-related systems* (2nd ed.). International Electrotechnical Commission.

Lee, D. & M. Yannakakis (1996). Principles and methods of testing finite state machines – a survey. *Proceedings of the IEEE 84*(8), 1090–1123.

Lee, E. A. & S. A. Seshia (2011). *Introduction to embedded systems: A cyber-physical systems approach*.

Ma, C. & J. Provost (2015). Design-to-Test Approach for Black-Box Testing of Programmable Controllers. In *IEEE Int. Conf. on Automation Science and Engineering (CASE), 2015*, pp. 1018–1024.

Myers, G. J., T. Badgett, & C. Sandler (2011). *The Art of Software Testing 3rd Edition* (3rd ed.), Volume 1. John Wiley & Sons, Inc.

Ogata, K. (2012). *Modern Control Engineering*. Prentice Hall.

Pressman, R. S. (2010). *Software Engineering A Practitioner's Approach Seventh Edition*. MC Graw Hill.

Provost, J., J.-M. Roussel, & J.-M. Faure (2014). Generation of Single Input Change Test Sequences for Conformance Test of Programmable Logic Controllers. *IEEE Trans.on Ind. Inform. 10*, 1696–1704.

Rösch, S., S. Ulewicz, J. Provost, & B. Vogel-heuser (2015). Review of Model-Based Testing Approaches in Production Automation and Adjacent Domains — Current Challenges and Research Gaps. *Journal of Software Engineering and Applications 8*(September), 499–519.

Sommerville, I. (2007). *Software Engineering*. Pearson Education Limited.