FBMTP: An Automated Fault and Behavioral Anomaly Detection and Isolation Tool for PLC-Controlled Manufacturing Systems

Arup Ghosh, Shiming Qin, Jooyeoun Lee, and Gi-Nam Wang

***Abstract*—The fault and behavioral anomaly detection and isolation (FBADI) in programmable logic controller (PLC) controlled systems has been under an active study for several decades. In this paper, we present a tool, called the fault and behavior monitoring tool for PLC (FBMTP) that can solve the FBADI problem in PLC-controlled manufacturing systems effectively. FBMTP first creates a nominal deterministic finite-state automaton-based model of the PLC control process, and then utilizes that model to detect and isolate the faults and the behavioral anomalies. The key idea is to check whether the observed behavior is consistent with the modeled behavior or not. The control signal data stored in the PLC memory is used for this whole procedure. With growing size of the manufacturing systems, it becomes difficult to obtain all the control signal data accurately because of the limited data acquisition time in each PLC scan cycle. The proposed control process modeling and the FBADI methodology of FBMTP can cope with such data inaccuracy problems associated with large manufacturing systems efficiently. Our experiments show that FBMTP provides highly accurate FBADI results for both small and large manufacturing systems.**

***Index Terms*—Anomaly detection and isolation, behavior monitoring, fault detection and isolation (FDI), log data analysis, manufacturing systems, programmable logic controller (PLC).**

I. Introduction

CURRENTLY, most manufacturing systems are controlled by programmable logic controllers (PLCs) because of its adaptability, modularity, robustness, and low cost [1]–[4]. The PLC-controlled manufacturing systems are often characterized as discrete event systems (DESs) [5]. It is very difficult to detect and isolate the faults and behavioral anomalies associated with PLC control processes [2]–[4]. In order to increase the availability and productivity of the manufacturing systems, it is necessary to detect and isolate the control process faults automatically, thus the engineers can fix them quickly before they cause more damage to the production operation. The automatic detection and isolation of the behavioral anomalies in a PLC-controlled system is another important requirement of the manufacturing industry and is attracting increased attention among researchers and industry personnel. This is because, it is possible to get a clear indication of the fault prior to its occurrence by the deviation from the system's actual behavior. Moreover, a behavioral anomaly detection and isolation (BADI) approach can also help operators to identify several other issues, such as device performance degradation, significant increment in occurrence of a system alarm, and so on. Therefore, such an automated tool that can solve the above-mentioned purposes effectively has been addressed in this paper.

In this paper, we present a tool called fault and behavior monitoring tool for PLC (FBMTP) that can solve the fault and behavioral anomaly detection and isolation (FBADI) problem in PLC-controlled manufacturing systems efficiently. FBMTP first constructs a deterministic finite-state automaton (DFA) with signal-status vector and time-out function (DSVTF)-based fault-free nominal model of the PLC control process; and then detects and isolates the faults and the behavioral anomalies using that model. A DSVTF automaton, informally, is a kind of DFA where a state is represented by a unique vector, called signal-status vector; and an additional function, i.e., time-out function is included into the DFA framework. This allows FBMTP to incorporate more information into the control process model. The core idea behind our approach is to compare the modeled and the observed behavior of the manufacturing system in order to identify and localize the faults and the behavioral anomalies. FBMTP utilizes log data records of the control signals acquired from PLC memory to accomplish this purpose. The existing literature on this subject (presented in the next section) primarily focuses on theoretical framework design. Most of them have used the control signal log data taken from small manufacturing systems for their experiments. The log data records obtained from a large PLC-controlled system generally contain numerous data inaccuracies. This is because, the complete process environment of a large PLC-controlled system cannot always be captured accurately due to an ultra-short data acquisition period in each PLC scan

Manuscript received June 28, 2016; accepted October 25, 2016. Date of publication December 16, 2016; date of current version November 16, 2017. This work was supported in part by the Defense Acquisition Program Administration and Agency for Defense Development, South Korea, under Grant UD140022PD, in part by the Ministry of Trade, Industry and Energy (MOTIE), South Korea, under Grant 10051146, in part by the ICT Research and Development Program of the Ministry of Science, ICT and Future Planning and Institute for Information and Communications Technology Promotion, South Korea, under Grant 10047046, and in part by the MOTIE and Korea Institute for Advancement of Technology, South Korea, under Grant N0001083. This paper was recommended by Associate Editor F. Sun.

A. Ghosh and S. Qin are with the Unified Digital Manufacturing Laboratory, Department of Industrial Engineering, Ajou University, Suwon 443-749, South Korea (e-mail: arupghosh22.3.89@gmail.com; taihejiang11@ajou.ac.kr).

J. Lee and G.-N. Wang are with the Department of Industrial Engineering, Ajou University, Suwon 443-749, South Korea (e-mail: jooyeoun325@ajou.ac.kr; gnwang@ajou.ac.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2016.2633392

cycle. Unfortunately, none of the existing approaches have addressed this fundamental issue. Our proposed tool FBMTP is completely results-oriented and can handle the log data inaccuracy problem associated with large manufacturing systems effectively (using a novel model approximation technique).

The remainder of this paper is structured as follows: an overview about the existing works in this field is presented in Section II. In Section III, the problem description and a brief overview of FBMTP tool are given. The control process modeling and the FBADI procedure of FBMTP are discussed in Sections IV and V, respectively. Section VI contains our conclusive remarks of the work followed by a list of relevant and state-of-the-art references.

II. Background Study

Over the past three decades, a lot of research works on fault detection and isolation (FDI) in DESs have been carried out. Those FDI approaches can broadly be classified into two categories.

A. Decentralized FDI Approaches

This type of FDI approaches are proposed for "informationally and geographically decentralized" DESs [6]. In a decentralized architecture, there are several local sites where the sensors and actuators report their data. As the amount of data collected at each site continues to increase, it becomes progressively more impractical to gather all the data at a centralized site for analysis. These approaches propose a set of local diagnosers (one per site) to solve the global diagnosis problem. Each local diagnoser reports its own diagnosis decision to a coordinating site based on the local observations. A global diagnoser running at the coordinating site scrutinizes those decisions and issues the final diagnostic decision based on that system architecture (there are also some approaches that do not require a global coordinator system). Important examples of such works include [6]–[15].

1) Limitations: The decentralized FDI approaches are mainly focused on regular large-sized DESs. The PLC-controlled large-sized DESs are different in nature from other large-sized DESs because: 1) they are naturally decomposed into several subsystems due to the input/output (I/O) port limitations of the PLC hardware and 2) each such subsystem controlled by a separate PLC does not (usually) have any interrelation with other subsystems (so, each PLC-controlled subsystem can be seen as a standalone centralized system). Among the aforementioned articles, only [11] and [13] directly address the FDI issue in PLC-controlled manufacturing systems. They basically partition the whole manufacturing system controlled by a single or multiple PLCs into several small subsystems and apply the decentralized FDI algorithms to avoid the issues associated with a single large-sized system. However, the virtual partitions do not solve the practical log data inaccuracy problem. This is because the sensor and actuator signal data are collected from the PLC memory through a single communication channel (as collecting data from the actual sensors and actuators is extremely costly); and only a fraction of those signals can be accessed at a given time. None of those papers have provided detailed insight of their data collection method. Anyhow, even for "truly" decentralized systems, FBMTP can effectively serve the purpose of the local diagnoser (please note that if the size of any PLC-controlled subsystem is large, then the FDI approach may encounter the same log data inaccuracy problem).

B. Centralized FDI Approaches

Most of the research articles on FDI in PLC-controlled systems fall into this category. These approaches are proposed for centralized DESs and therefore, the global diagnostic decision is taken by a centralized FDI algorithm based on only one system model. In past work [2], two complementary diagnosis models, i.e., logical and sequential diagnosis model were proposed for fault diagnosis. In another paper [16], a hierarchical diagnosis model based on the fault tree analysis was proposed (in addition to the logical and sequential diagnosis model) to improve the fault diagnosis procedure. A hamming distance-based signal event sequence matching mechanism for fault diagnosis was proposed in [3]. In a related article [4], a method was presented that automatically generates a knowledge base from the PLC program and circuit diagrams; and using that knowledge base performs fault diagnosis. A similar work was carried out by Fan *et al.* [17]. In [18], an approach was proposed that automatically generates the PLC code from the discrete event process model (expressed as either PLC code or as extended finite automata) for fault detection. A completely different type of approach that models the control process by using a distinct class of automata, called nondeterministic autonomous automaton with output (NDAAO) was introduced by Klein *et al.* [19]. There are many articles which later extended that basic work. Most recent examples of such articles include [20]–[23]. In these types of approaches, the nominal manufacturing process behavior is represented by an NDAAO automaton and the concept of residuals adapted for DESs is applied for fault localization. In [24], a similar automaton model named, timed autonomous automaton with output was introduced for control process modeling. This paper basically includes the state transition time information into the earlier NDAAO model and uses a novel timed residuals and generic fault symptoms-based method for fault isolation. In more recent work [25], a finite state machine (FSM)-based approach was proposed for fault detection. In [26], the P-invariant of Petri Nets was applied to discover the sequence faults and the exclusive logic functions were applied to detect the sensor and actuator faults.

1) Limitations: In summary, the centralized FDI approaches propose a single nominal control process model for the whole manufacturing system by which they can detect and isolate the faults. All of them utilize the log data records of the PLC control signals to accomplish their purpose. As stated earlier, in large PLC-controlled systems, it is nearly impossible to capture all the signal event data accurately. Unfortunately, none of the above approaches address that fundamental issue. The approaches proposed in [18] and [26] do not fall into this category as they use the PLC program for fault diagnosis. However, approaches like this require additional PLC scan time and therefore, increases the difficulty to achieve fast responses in real-time control. Moreover, there is no obvious way to extend these approaches to also tackle the BADI problem.

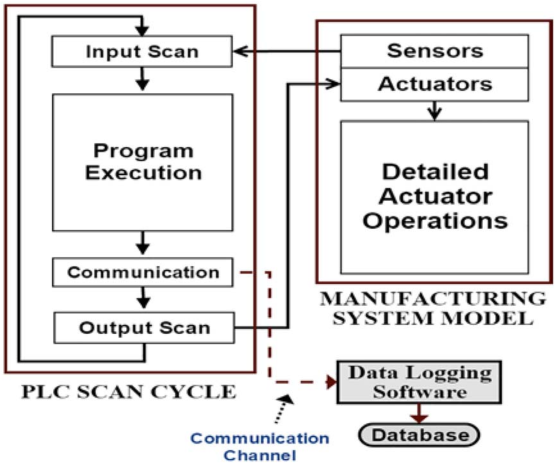


Fig. 1. PLC-controlled manufacturing system model.

There is an increasing need in industry for an automated tool that can overcome the above-mentioned limitations and can solve the BADI problem effectively. In this paper, we have addressed those needs.

III. Problem Statement and FBMTP System Overview

In Fig. 1, an overall model of the PLC-controlled manufacturing system is given. The PLC controls the manufacturing process according to the control program embedded in its controller. The control program is executed as a part of a repetitive process, known as PLC scan cycle. In order to interact with the physical manufacturing system, the PLC controller needs to perform following four steps procedure in each PLC scan cycle (see Fig. 1).

1) *Input Scan:* All the input signal values from the sensors are collected.
2) *Program Execution:* The control program is executed by using the sensor signal values and the output values for the actuators are determined.
3) *Communication Phase:* The PLC communication processor communicates with the external modules (such as, data logging module).
4) *Output Scan:* The newly determined output values are sent to the physical actuators (this causes the actuator operations to take place).

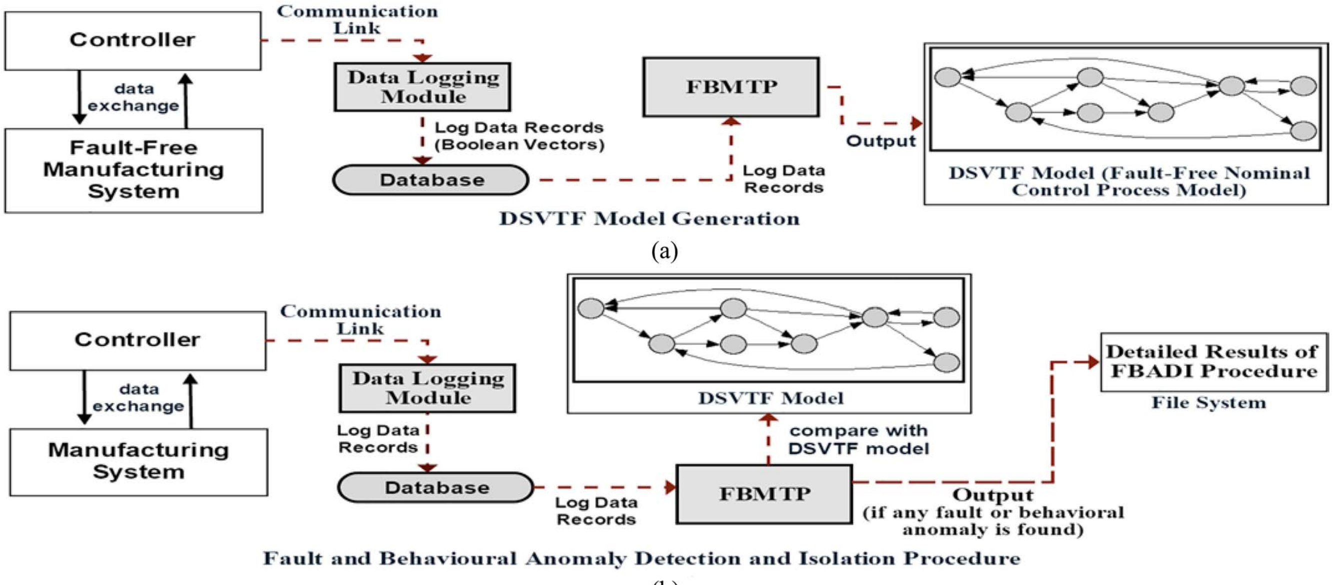
During the input scan and the program execution phase, the sensor and the actuator signal values are written to an internal PLC memory block, known as PLC memory table. It typically takes a very few milliseconds to complete a PLC scan (known as, the PLC scan time) and most of them are actually spent on the I/O scan and the program execution phase.

The discrete state I/O signal values (i.e., sensor and actuator signal values) stored in PLC memory table indicate the operating states of the manufacturing system by which the FBADI procedure can be carried out. The data logging module of FBMTP continuously records the I/O signal data from the PLC memory table and inserts them into the log database for further analysis. The objective is to detect and isolate the faults and the behavioral anomalies by using those log data records. The log data records are actually the Boolean vectors of length N, where N is the total number of PLC I/O signals. The Boolean values 1 and 0 represent the ON and the OFF status of a PLC I/O signal, respectively. A fault occurs in the control process, if any signal is found in an erroneous state (i.e., instead of ON state, an OFF state is observed and vice versa) and behavioral anomalies take place, if significant change in any signal behavior (such as, status transition time behavior) is observed. The total number of PLC I/O signals N can be very high in a large-sized manufacturing system. As can be seen in Fig. 1, the log data records can be collected for a very short duration of time in each PLC scan cycle. For this reason, in practice, only a fixed and limited number of signals, say n signals, can be accessed in each PLC scan cycle. Actually, a data logging software maintains a Boolean vector containing the N number of I/O signal values and updates the n signal values (in general, sequentially) in each PLC scan cycle ($n \leq N$). It is easy to realize, if $n < N$ then the acquired log database can contain numerous data inaccuracies (in this paper, with the term "large manufacturing system," we actually refer this kind of systems, where $n < N$, rather than its actual size). Unlike the other approaches cited previously, FBMTP can efficiently deal with this data inaccuracy problem.

In Fig. 2, an overall model of the working procedure of FBMTP is given. As can be seen from Fig. 2(a), at first a nominal DSVTF model of the PLC control process is built by using the signal log data taken from a fault-free manufacturing system. In the literature, it is often argued that it is hard to find a fault-free manufacturing process in real life. Actually, we use the term "fault-free" to denote the acceptable one. This makes it possible that some soft faults remain in the system [in PLC terminology, the term "soft fault" refers to a fault that is not fatal to the system operation (in other words, does not trigger an uncontrollable situation in the physical plant)]. However, since the model contains the soft faults in an acceptable range (a very few), it is always possible to identify the real fatal one. The presence of low number of soft faults also help to avoid a huge number of unimportant fault detections (this is also the case for the behavioral anomalies). The overview of the FBADI procedure of FBMTP is depicted in Fig. 2(b). As we can see, the nominal DSVTF model is compared with the observed signal log data in order to solve the FBADI problem. If no (or little) difference between the observed and the expected behavior of the PLC I/O signals is found then the system is supposed to be the fault and behavioral anomaly free. Otherwise, it signifies a fault or a behavioral anomaly has taken place, which is reported to the file (with detailed diagnosis information).

IV. FBMTP: Control Process Modeling Approach

This section is divided into three sections. In Section IV-A, the DSVTF model building procedure is presented. The DSVTF model approximation technique for large manufacturing systems is discussed in Section IV-B. In Section IV-C, some additional process information related to the multiple transitions is included in the DSVTF model.



(a)

(b)

Fig. 2. FBMTP system overview. (a) DSVTF model generation procedure. (b) FBADI procedure.

A. DSVTF Model Formulation

In FBMTP, the complete PLC-controlled manufacturing process is designed by using a single DSVTF automaton-based model (centralized approach). The DSVTF automaton is similar in concept to the timed automata [5], [27] or the timed FSM [28]–[30], which are the most widely used framework for DES modeling. We have particularly used the deterministic automaton for control process modeling because, even though the physical manufacturing plant can be nondeterministic in nature, the PLC control program must have to be deterministic (see [2], [31], [32]). So, the DSVTF model of the PLC control process must have also to be deterministic. The DSVTF model can be characterized as follows.

Definition 1: A DFA, denoted by G, is a six-tuple automaton of the form.

$$G = (X, E, f, \Gamma, x_0, X_m)$$

where

X finite set of states and E is the finite set of events associated with G;

$f : X \times E \to X$ transition function;

[So, $f(x, e) = x'$ means that there exists a transition labeled by event e from state x to x'.]

$\Gamma : X \to 2^E$ active or feasible event function;

[So, $\Gamma(x)$ represents the set of all events e for which $f(x, e)$ is defined.]

$x_0 \in X$ initial state and $X_m \subseteq X$ is a nonempty set of final states.

Definition 2: A DSVTF (a special kind of DFA), denoted by G_{ST}, is a seven-tuple automaton of the form

$$G_{\text{ST}} = (X, E, \text{TO}, f, \Gamma, x_0, X_m)$$

where, a new function, i.e., time-out function expressed as $\text{TO} : X \times E \to \mathbb{N}_0$ ($\mathbb{N}_0$ is the set of all natural numbers) is incorporated into Definition 1; and the concept of "event" in Definition 1 is substituted with the concept of "signal-status vector" (other symbols has the same meaning).

[A signal-status vector actually indicates the signal status change (SSC) operations between the two states (defined later).]

So, if $f(x, e) = x'$ and $\text{TO}(x, e) = t$, denoted as $x \to_{e(t)} x'$, then we can say that: automaton G_{ST}, being in state x, accepts the signal-status vector e within the time t and moves to the state x'.

[The above-specified time is often referred to as the Transition Time-Out (TTO) time (each state transition is required to be finished within its corresponding TTO time).]

Definition 3: A state belongs to a DSVTF automaton G_{ST} (constructed for a PLC-controlled manufacturing system) is represented by the Boolean vector: $[\text{IO}_1, \text{IO}_2, \ldots, \text{IO}_N]$, where N is the total number of PLC I/O signals and IO_i represents the status of the ith PLC I/O signal $(\forall i = 1, 2, \ldots, N)$

$$\text{IO}_i = \begin{cases} 1 & \text{if the status of the } i\text{th signal is ON} \\ 0 & \text{if the status of the } i\text{th signal is OFF} \end{cases} (\forall i = 1, \ldots, N).$$

Definition 4: A signal-status vector that causes a transition between the two states of a DSVTF automaton, is composed of the SSC events between those two states (see the formulation below).

Suppose, the Boolean vectors associated with state x and x', denoted by BV_x and $\text{BV}_{x'}$, are of length N; and the value stored at the ith position of the Boolean vector BV is represented by $\text{BV}[i]$; and the actual name of the ith signal is denoted by Sig_i; then, the signal-status vector e that causes a transition between the state x and x' is represented by

$$e = [\Psi(\text{BV}_x[1], \text{BV}_{x'}[1]), \ldots, \Psi(\text{BV}_x[N], \text{BV}_{x'}[N])]$$

where, Ψ is the SSC detection function and is formally expressed as follows:

$$\Psi(\text{BV}_x[i], \text{BV}_{x'}[i]) = \left(\begin{array}{ll} \text{Sig}_i\text{\_ON} & \text{if } \text{BV}_x[i] = 0 \text{ and } \text{BV}_{x'}[i] = 1 \\ \text{Sig}_i\text{\_OFF} & \text{if } \text{BV}_x[i] = 1 \text{ and } \text{BV}_{x'}[i] = 0 \\ \text{null}* & \text{otherwise} \end{array} \right. \quad (\forall i = 1, \ldots, N)$$

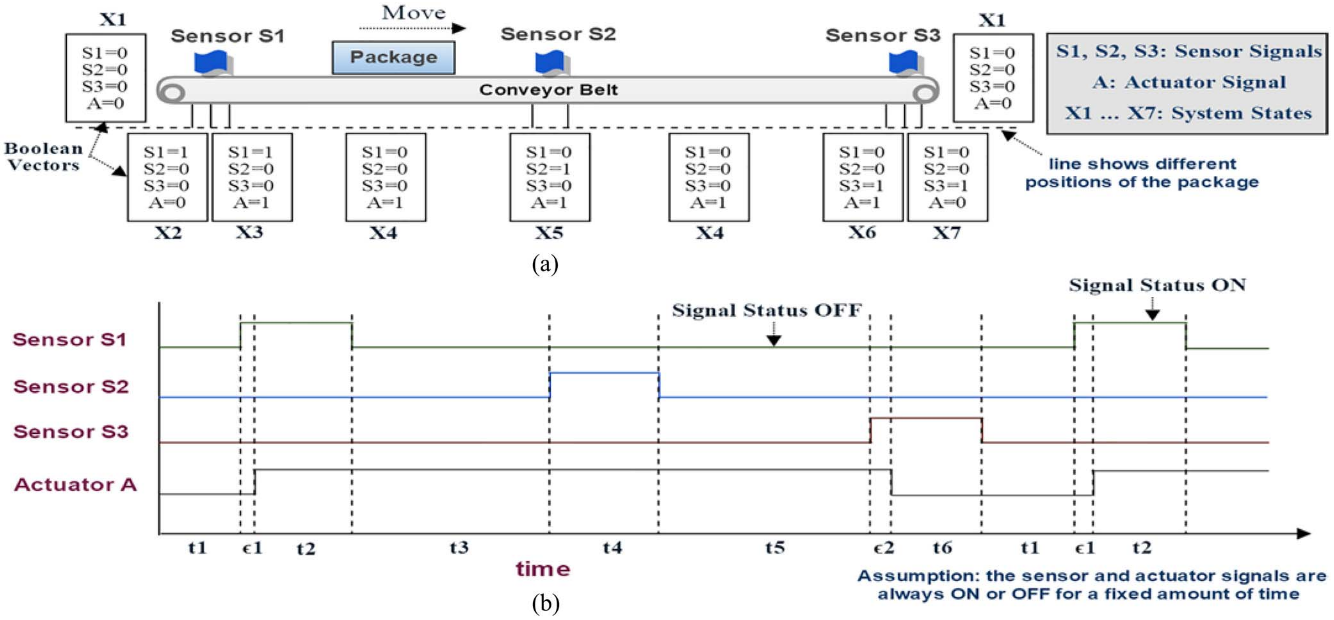


Fig. 3. Conveyor belt system overview. (a) Package carrying process of the conveyor belt. (b) Signal versus time chart of the PLC I/O signals.

*[*will not be included in the signal-status vector].*

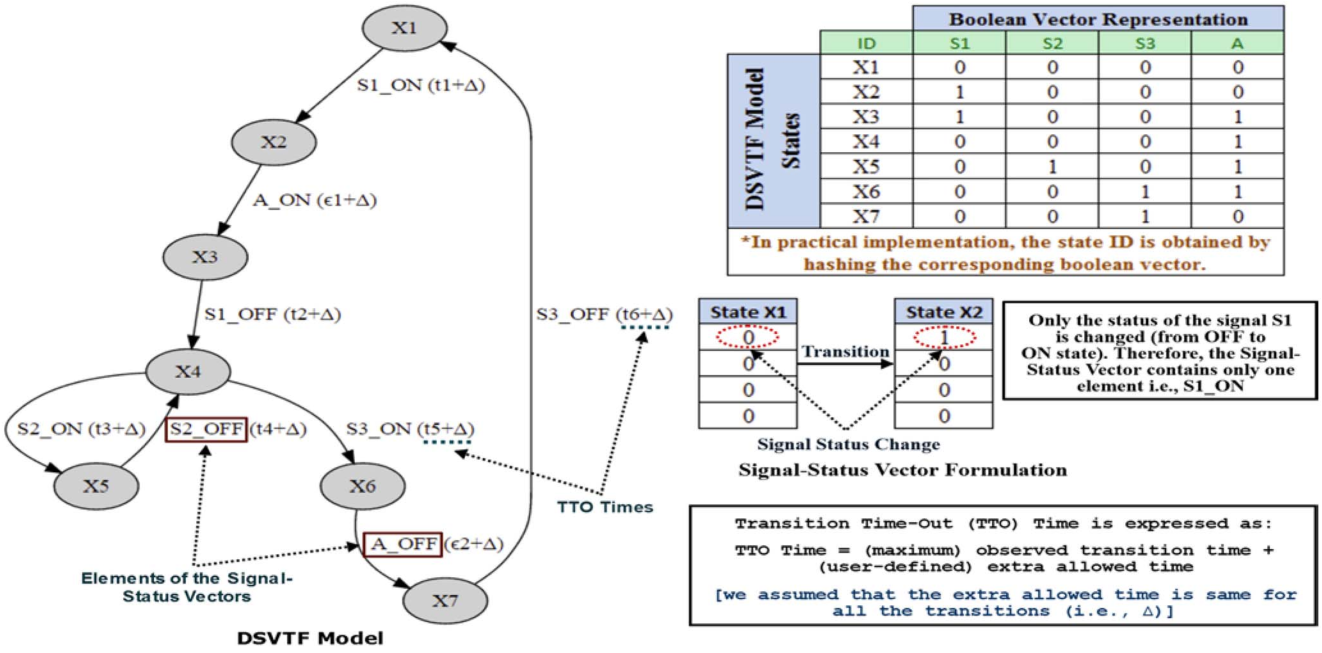
We further discuss the DSVTF model formulation procedure with the help of an example conveyor belt scenario, as shown in Fig. 3 (suppose the manufacturing system contains only one conveyor belt). In Fig. 3(a), the package carrying procedure of the conveyor belt is shown. As we can see, this system has three sensors to sense the location of the package and one actuator to move the conveyor belt (in total four I/O signals). The Boolean vectors (or the DSVTF model states) associated with different package positions depict the system state transitions of the conveyor belt system (in this paper, the terms "state" and "system state" are used interchangeably). As can be seen in Fig. 3(a), initially, all the signals are in OFF state. The package carrying process starts when a package is set on the conveyor belt. So, the sensor at the front end, i.e., S1 detects the package and the signal S1 becomes ON. This turns ON the actuator signal A and hence, the conveyor belt starts to move. When the package goes beyond the range of sensor S1, the signal S1 becomes switched OFF. If the sensor signal S2 (respectively, S3) is turned ON that means the package has reached the middle (respectively, end) position of the conveyor belt. The actuator signal A goes to OFF state when the package reaches the end position of the conveyor belt. This process ends when the package is taken out from the conveyor belt (as a result, the sensor signal S3 becomes switched OFF). This cycle, called the system cycle, starts again when another package is placed on the conveyor belt [see Fig. 3(a)].

In Fig. 3(b), the signal versus time chart of the above-specified I/O signals is presented. In that figure, we assumed that the time period after which a new package is presented to the system, i.e., time $t1$ is always the same; and a small amount of time $\varepsilon 1/\varepsilon 2$ is required before the actuator signal A goes to ON/OFF state. The DSVTF model of the above-stated system is provided in Fig. 4 (also see Definitions 1–4). In practice, the state identification name (state ID) of a DSVTF model state that represents a unique Boolean vector, is obtained by hashing that Boolean vector (in order to achieve the fast execution time and low memory-space requirement). It is easy to see from Fig. 4, a signal-status vector that causes a transition between two states, is actually comprised of the SSC events between those two states (signal name and its status are separated by a "_"). In all our figures, a signal-status vector of a transition is given as a list of space-separated SSC events associated with that transition. The TTO time of a transition (in all our figures, it is presented inside a parentheses associated with that transition) is assumed to be a little more than the actual observed transition times (see Fig. 4). It is easy to see that we are able to incorporate all the necessary information of the above-stated system (such as, SSC event order information, state transition time information, etc.) in the DSVTF model of Fig. 4. This compact representation helps FBMTP to achieve high performance and accuracy during the FBADI phase. Moreover, the DSVTF model is designed based on the system state transitions. For this reason, the data logging module of FBMTP only has to insert into the database the Boolean vectors in which at least one signal status value is changed. This effectively reduces the overall disk I/O operations and hence, increases the performance of the actual data logging task.

B. DSVTF Model Approximation Technique for Large Manufacturing Systems

This section is separated into two sections. In Section IV-B1, the proposed DSVTF model approximation technique for large-sized DESs is explained and justified. In Section IV-B2, the model approximation technique for a special type of log data inaccuracy problem, called the "critical signal miss event," is discussed.

1) Model Approximation by Eliminating the Redundant Transition Paths and the Unstable System States: The previously described DSVTF model (see Figs. 3 and 4) was constructed based on several assumptions. The first and foremost assumption is that all the SSC events can be captured accurately by the data logger, which is not true for large manufacturing systems (recall that in large systems, all the I/O



DSVTF Model States	Boolean Vector Representation				
	ID	S1	S2	S3	A
	X1	0	0	0	0
	X2	1	0	0	0
	X3	1	0	0	1
	X4	0	0	0	1
	X5	0	1	0	1
	X6	0	0	1	1
	X7	0	0	1	0

*In practical implementation, the state ID is obtained by hashing the corresponding boolean vector.

State X1	State X2
0	1
0	0
0	0
0	0

Only the status of the signal S1 is changed (from OFF to ON state). Therefore, the Signal-Status Vector contains only one element i.e., S1_ON

Transition Time-Out (TTO) Time is expressed as:

TTO Time = (maximum) observed transition time + (user-defined) extra allowed time

[we assumed that the extra allowed time is same for all the transitions (i.e., Δ)]

Fig. 4. DSVTF model of the conveyor belt system of Fig. 3.

signal data cannot be acquired in a single PLC scan cycle). Let us assume that only two signals can be accessed in every δ time interval (δ is the PLC scan time). If we apply this constraint to the conveyor belt system of Fig. 3, it can lead to several log data inaccuracy problems, such as the SSC event sequence disorder, the loss of some real SSC events from the SSC event sequence, etc. Due to the limited signal data sampling rate, an actual state or transition can be lost from the DSVTF model; and an incorrect (partially observed) state or transition can appear in the DSVTF model. In Fig. 5(a), an example of such phenomenon is shown. In reality, the SSC event S3_ON (transition of signal S3 from OFF to ON state) is followed by the SSC event A_OFF. However, as can be seen in Fig. 5(a), they are observed to occur at the same time in an observation (which is an incorrect event sequence order). For this reason, all the information associated with state $X6$ (or state transition $X4 \rightarrow X6$) is lost from the log data (for a particular system cycle) and an incorrect transition from state $X4$ to $X7$ takes place in the DSVTF model. It is easy to realize from Fig. 5(a), the transition path times of the two transition paths between state $X4$ and $X7$ (i.e., $X4 \rightarrow X6 \rightarrow X7$ and $X4 \rightarrow X7$) must have to be almost equal because, the PLC scan time δ is a very short time [note that the transition path time of a transition path is equal to the addition of the transition times of all the transitions of that path]. In Fig. 5(b), another example of such DSVTF model (resulted due to the disorder of the sequence of observed SSC events) is presented. This figure is actually taken from a DSVTF model of a real-world manufacturing process (for generality, the SSC event names, time values and state IDs are changed). As can be seen in Fig. 5(b), some incorrect system states and transition paths have appeared in the DSVTF model because of the above-specified problem (the path $A \rightarrow B \rightarrow F$ is actually the correct one). It is impossible for an automated tool or algorithm to identify the correct states and transition paths based only on the observed data. The multiple transition paths between the two states that get generated due to the inaccuracies in the observed log data, can be considered as the duplicate transition paths. As an example, in Fig. 5(b), the transition paths between state A and F can be categorized as the duplicate transition paths (it is easy to realize, they basically represent the same transition path). For large-sized systems, it is impractical as well as meaningless to keep all the duplicate transition paths in the DSVTF model. This is because, it can eventually lead to a memory-space or state-space explosion problem. Moreover, it greatly slows down the execution speed of the approach.

In order to deal with the log data inaccuracy issue, FBMTP takes an approach that removes all the duplicate (or redundant) transition paths and the unstable system states from the DSVTF model, thus the state-space (or memory-space) explosion and the execution speed degradation problem can largely be circumvented. The term “unstable system state” refers to a state, where the process or the execution spends (or stops) less time than the time inaccuracy bound (TIB) time of the data logger.

Definition 5: The TIB time associated with a PLC data logger is defined by using the following equation:

$$\text{TIB time} = (\lceil N/n - 1 \rceil \times \delta) + \partial$$

where

N total number of PLC I/O signals;

n number of signals that can be accessed in each PLC scan cycle (during the communication phase);

δ PLC scan time;

∂ small amount of extra time.

[The value of ∂ depends on the data logging method (example: sequential versus random access), time-stamp protocol, and communication link performance.]

In practice, the TIB time is usually a few milliseconds and an operator generally assigns a very few milliseconds to the

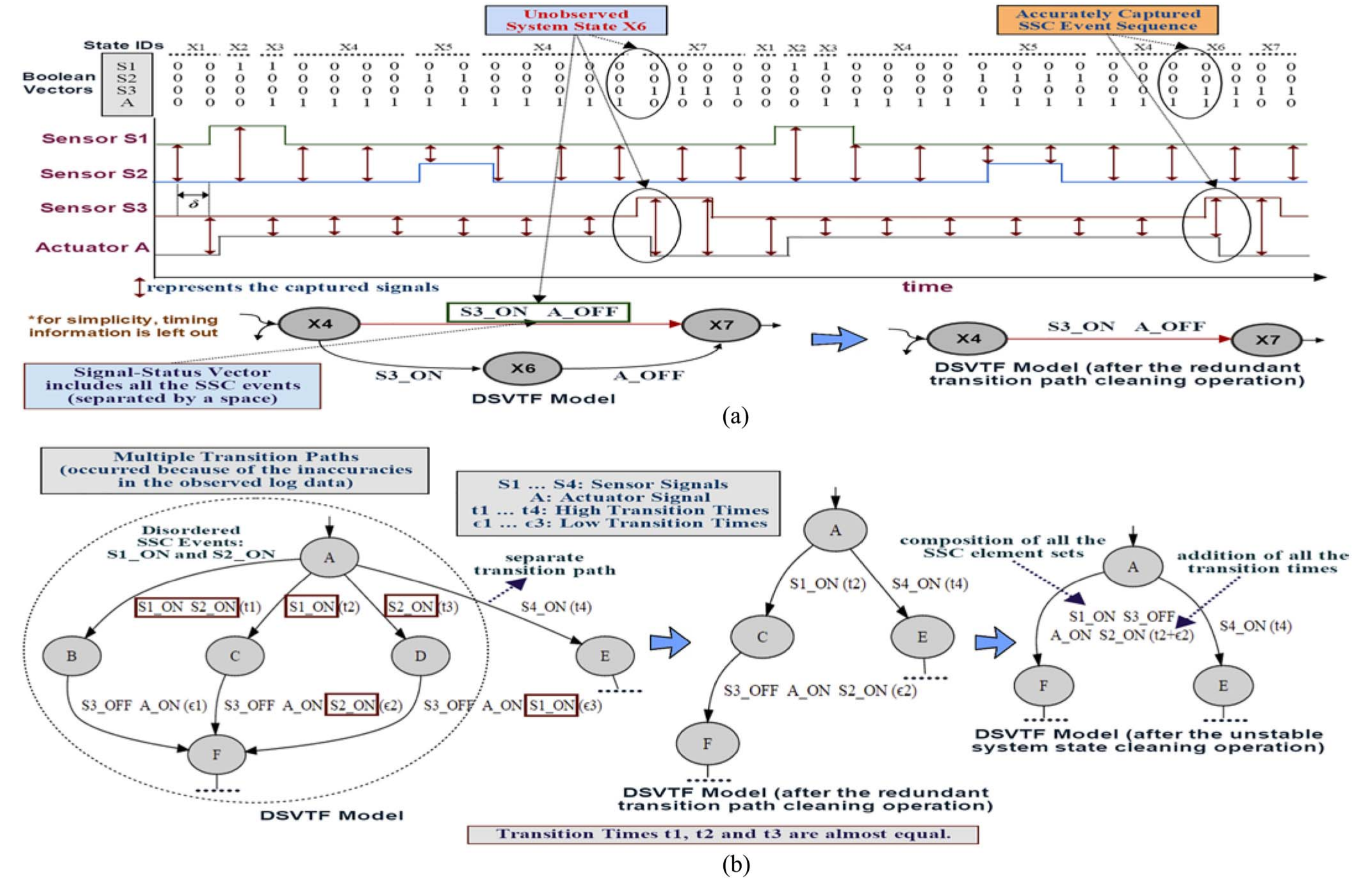

Unobserved System State X6
Accurately Captured SSC Event Sequence
State IDs
X1 X2 X3 X4 X5 X4 X7 X1 X2 X3 X4 X5 X4 X6 X7
Boolean Vectors
S1
S2
S3
A
Sensor S1
Sensor S2
Sensor S3
Actuator A
δ
represents the captured signals
time
*for simplicity, timing information is left out
X4
S3_ON A_OFF
X7
X6
S3_ON
A_OFF
Signal-Status Vector includes all the SSC events (separated by a space)
DSVTF Model
DSVTF Model (after the redundant transition path cleaning operation)
(a)
Multiple Transition Paths (occurred because of the inaccuracies in the observed log data)
S1 ... S4: Sensor Signals
A: Actuator Signal
t1 ... t4: High Transition Times
ϵ1 ... ϵ3: Low Transition Times
Disordered SSC Events: S1_ON and S2_ON
separate transition path
A
B
C
D
E
F
S1_ON S2_ON (t1)
S1_ON (t2)
S2_ON (t3)
S4_ON (t4)
S3_OFF A_ON (ϵ1)
S3_OFF A_ON S2_ON (ϵ2)
S3_OFF A_ON S1_ON (ϵ3)
DSVTF Model
S1_ON (t2)
S4_ON (t4)
S3_OFF A_ON S2_ON (ϵ2)
DSVTF Model (after the redundant transition path cleaning operation)
Transition Times t1, t2 and t3 are almost equal.
composition of all the SSC element sets
addition of all the transition times
S1_ON S3_OFF A_ON S2_ON (t2+ϵ2)
S4_ON (t4)
DSVTF Model (after the unstable system state cleaning operation)
(b)

Fig. 5. Examples of the SSC event sequence disorder problem. (a) Occurrence of an incorrect transition in the DSVTF model. (b) Appearance of some incorrect system states and transition paths in the DSVTF model.

time ∂ (close to the PLC scan time δ) from their experience to avoid the detailed technical calculations. In all our figures, the transition time that falls below the TIB time is represented by the symbol "∊." It is easy to perceive from Definition 5, an actual system state can remain unobserved or partially observed only if that state is an unstable system state. In case of a stable system state, the data logger can primarily produce a partially observed state (or states) while trying to capture it; however, the complete system state is eventually being captured accurately (less than TIB time). Please note that any such partially observed state is ought to be treated as the unstable system state (because, it cannot be decided explicitly weather an unstable system state in the DSVTF model is a partially observed state or an accurately captured unstable system state). It is easy to perceive that the presence of unstable system states in the DSVTF model is the reason behind the redundant transition paths. FBMTP identifies whether a pair of transition paths between any two states are resulted due to an inaccurate data sampling method (the redundant transition paths) or not by using the following rule.

Rule 1: A pair transition paths between two states are redundant or duplicate transition paths if the following three conditions are satisfied**.

1) The SSC element sets of those two transition paths are equal.

   [The SSC element set of a transition path refers to the set of all the SSC elements of all the signal-status vectors of that transition path.]

2) The transition path times of those two transition paths are almost equal (in other words, their difference falls below an user-defined time threshold).

3) All the intermediate states of those two transition paths are the unstable system states (in other words, the transition from any intermediate state to its immediate next state takes less time than the TIB time).

   *[**There exists an exception to rule 1, called the critical signal miss event, where some SSC events can be missing from the SSC element sets (we will discuss more on it later).]*

By applying rule 1 on a pair of transition paths between two states, FBMTP identifies whether they are redundant transition paths (the user-defined threshold value is set to a small number—we will discuss more on it later). If so, then any one of them is selected and the other one is discarded from the DSVTF model (after all the processing, the final DSVTF model will be the same). This process continues until all the redundant transition paths are removed from the DSVTF model [in other words, this process is applied on each pair of transition paths between every pair of states [see Fig. 5(a) and (b)]. By using this procedure, FBMTP basically discards all the transition paths from the DSVTF model that represent the same SSC event and almost same transition path time information with any other transition paths. It is possible that a pair transition paths detected as the redundant transition paths by using rule 1, are actually distinct or separate transition paths (although extremely unlikely, because

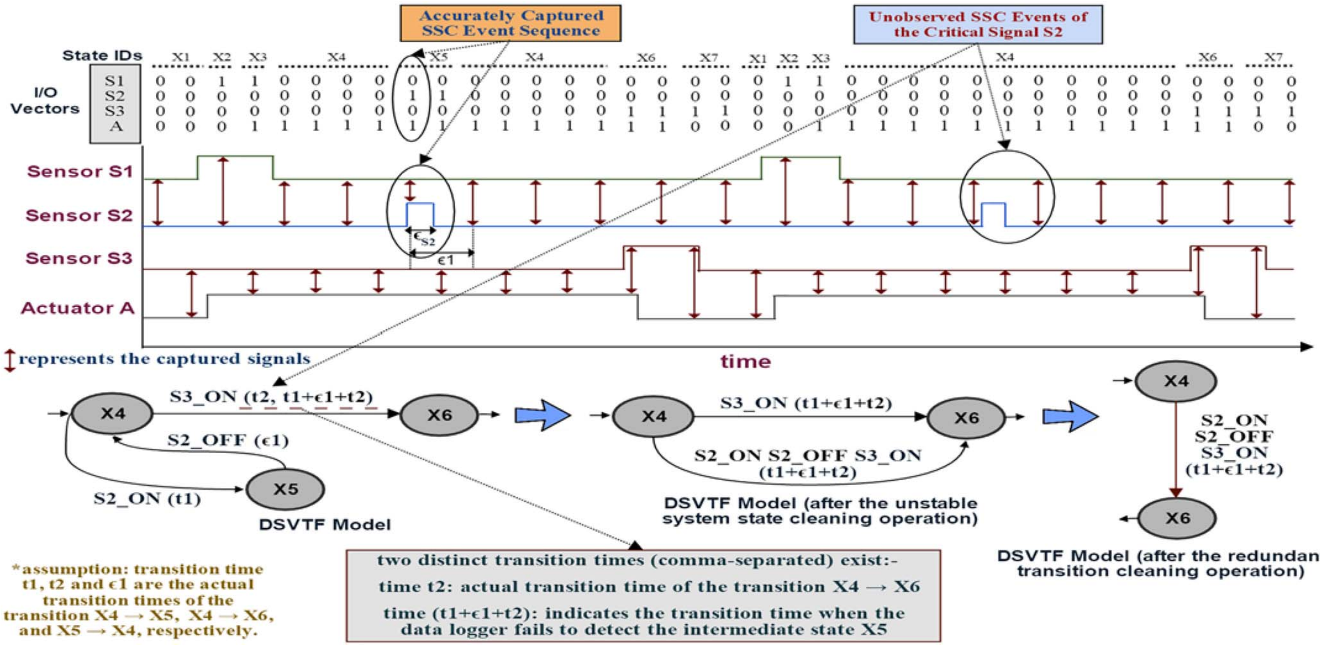

Accurately Captured SSC Event Sequence
Unobserved SSC Events of the Critical Signal S2
State IDs
X1 X2 X3 X4 X5 X4 X6 X7 X1 X2 X3 X4 X6 X7
I/O Vectors
S1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
S2 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
S3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
A 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
Sensor S1
Sensor S2
Sensor S3
Actuator A
ϵS2
ϵ1
represents the captured signals
time
X4
S3_ON (t2, t1+ϵ1+t2)
X6
S2_OFF (ϵ1)
S2_ON (t1)
X5
DSVTF Model
X4
S3_ON (t1+ϵ1+t2)
X6
S2_ON S2_OFF S3_ON (t1+ϵ1+t2)
DSVTF Model (after the unstable system state cleaning operation)
X4
S2_ON S2_OFF S3_ON (t1+ϵ1+t2)
X6
DSVTF Model (after the redundant transition cleaning operation)
*assumption: transition time t1, t2 and ϵ1 are the actual transition times of the transition X4 → X5, X4 → X6, and X5 → X4, respectively.
two distinct transition times (comma-separated) exist:-
time t2: actual transition time of the transition X4 → X6
time (t1+ϵ1+t2): indicates the transition time when the data logger fails to detect the intermediate state X5

Fig. 6. Example scenario of the critical signal miss event.

of such strict criteria). There is no automated way to identify such cases because, the same paths can also get generated due to the log data inaccuracy issues [the SSC event sequence can get disordered because of the presence of the intermediate unstable system states (see rule 1)].

After performing this path cleaning operation, FBMTP starts to eliminate the remaining unstable system states from the DSVTF model. The key idea is to keep only those states (or state transitions) in the model that cannot remain unobserved or partially observed (in other words, cannot be lost from the log dataset) because of the log data inaccuracy issues. As an example, the state C in the transition path $A \rightarrow C \rightarrow F$ of Fig. 5(b) is discarded because, the process spends a very short time ε_2(<TIB) in that state. As we can see from Fig. 5(b), the signal-status vector of the resulted transition (i.e., $A \rightarrow F$) is the aggregation of the signal-status vectors of that path; and the transition time of it is equal to that transition path time. In this paper, a transition that is generated as a result of the above-stated model cleaning operations (for example, transition $A \rightarrow F$), is referred to as the "type I transition"; and the end node of a type I transition is referred to as the "type I system state" (FBMTP deals with them differently during the FBADI phase).

2) Dealing With the Critical Signal Miss Event Issue: In this section, we discuss a special type of log data inaccuracy problem, called the critical signal miss event where, the data logger fails to detect both the SSC events of a "critical signal" in a particular signal cycle. In manufacturing sector terminology, a signal that changes its status very quickly (less than TIB time) is called a critical signal. A critical signal miss event refers to an incident, where the data logger fails to detect both the OFF to ON and the ON to OFF status transformation events of a critical signal in a signal cycle. It is easy to perceive that in any case if the data logger fails to detect an OFF to ON status transformation event of a signal, then it will also fail to detect the successive ON to OFF status transformation event of that signal (and vice versa—this is obvious because no change in the status of the signal will be observed); and this can happen only if that signal is a critical signal (otherwise, both the SSC events will eventually be detected). As an example, let us assume that the sensor signal S2 of Fig. 3(b) goes to ON state for a very short period of time ε_{S2}(<TIB—implies that the signal S2 is a critical signal). In Fig. 6, an example scenario of such critical signal miss event is given. As we can see, the data logger fails to detect both the SSC events of the critical signal S2 (i.e., S2_ON and S2_OFF) in the second signal cycle and for that reason, two transition paths between state $X4$ and $X6$ appear in the DSVTF model. Although they are resulted due to the log data inaccuracy issue, they are not considered as the redundant transition paths by rule 1 (in rule 1, we basically assumed that the SSC events can be disordered; but, none of them can be lost from the SSC event sequence). FBMTP identifies this kind of redundant transition paths after the unstable system state cleaning operation.

The unstable system state cleaning operation transforms a pair of redundant transition paths (between any two states) that are actually resulted due to the critical signal miss events into a pair of redundant transitions between two stable system states (because, all the intermediate unstable system states have already been eliminated). However, both the SSC events of the critical signals must have to be missing from the signal-status vector of any one of those two transitions. For example, as can be seen in Fig. 6, after the unstable system state cleaning operation, there exists two transitions between the stable system state $X4$ and $X6$; and both the SSC events of the critical signal S2 are missing from the signal-status vector of the transition $X4 \rightarrow_{\text{S3\_ON}(t1+\varepsilon 1+t2)} X6$ (the intermediate state $X4$ of the transition path $X4 \rightarrow X5 \rightarrow X4 \rightarrow X6$ is also discarded along with the unstable system state $X5$; otherwise, a self-loop gets created which is invalid according to the DSVTF model definition). By using the above-discussed intuition, FBMTP can easily identify whether a pair of transitions between two stable system states are generated due to the critical signal miss events.

Rule 2: Suppose, after the unstable system state cleaning operation, a pair of transitions between the stable system state x_a and x_b are: $x_a \rightarrow_{e1(t1)} x_b$ and $x_a \rightarrow_{e2(t2)} x_b$; and the SSC element sets of those two transitions (say, E_1 and E_2) are: $E_1 = <e1>$ and $E_2 = <e2>$ [where, $<e>$ represents all the SSC elements of the signal-status vector e].

The transitions are generated because of the critical signal miss events if the following three conditions are satisfied (i, Sig_i and N have the same meaning as in Definition 4).

1) $E_1 \neq E_2$.
2) If $\text{Sig}_i\text{\_ON} \in ((E_1 \cup E_2)\backslash(E_1 \cap E_2))$ then, $\text{Sig}_i\text{\_OFF} \in ((E_1 \cup E_2)\backslash(E_1 \cap E_2))$ $(\forall i = 1, 2, \ldots, N)$.
3) $|t1 - t2| <$ user-defined (small) time threshold. [If the transitions are resulted due to the critical signal miss events, then the signal Sig_i for which $\text{Sig}_i\text{\_ON} \in ((E_1 \cup E_2)\backslash(E_1 \cap E_2))$ is a critical signal $(\forall i = 1, 2, \ldots, N)$.]

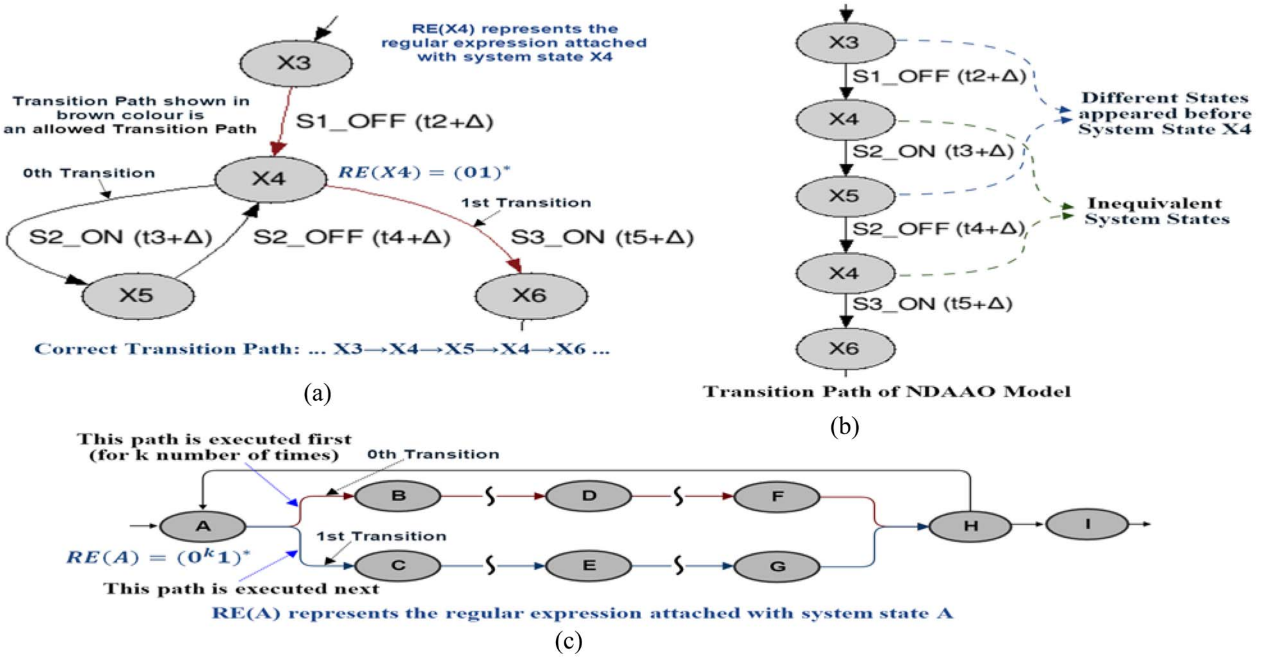
By applying rule 2 on a pair of transitions between two stable system states, FBMTP identifies whether they are generated because of the critical signal miss events (redundant transitions). If so, then both the transitions are discarded and a new transition is created in place of them. The SSC element set of the new transition is the aggregation of the SSC element sets of those two transitions (see Fig. 6); and the transition time of it is equal to the minimum transition time of those two transitions (the reason will be clear in the next paragraph). This process continues until all the redundant transitions are eliminated from the DSVTF model. In this paper, this type of transition (resulted due to the above-stated model cleaning operation) is referred to as the "type II transition" (its signal-status vector contains the SSC events of the critical signals); and the end node of a type II transition is referred to as the "type II system state." Please note that the above-stated model cleaning operations do not affect the accuracy of the FBADI procedure much because, in reality, the TIB time is a very short amount of time. So, a very small number of transitions and the unstable system states between any two stable system states are actually eliminated (even so, the overall elimination will be very high). Actually, the effect of log data inaccuracy problem on the accuracy of FBADI procedure is unavoidable; and keeping the redundant transitions (or transition paths) and the unstable system states in the DSVTF model cannot help an approach in that regard. This is because, it cannot be decided categorically whether such cases have occurred because of the log data inaccuracy problems or some unexpected control process situations.

We must mention here that in all our signal versus time chart examples [see Figs. 3(b), 5(a), and 6], it is shown that the signals are always ON or OFF for a fixed amount of time. In practice, this is not always the case. For example, as can be seen in Fig. 3(b), the actuator signal A is ON until the package reaches the end position of the conveyor belt. This time depends on the speed of the physical device and hence, cannot always be the same. It also varies depending on the timing accuracy of the data logger (i.e., TIB time). The transition time instances of a particular transition also vary because of these reasons. In most of the cases, the transition time instances of a transition do not vary much because of the almost fixed speed of the physical device and a very short TIB time period. However, in some cases, the transition time instances can fluctuate immensely, especially if they are dependent on the operators' inputs. FBMTP considers the minimum value of transition time instances of a transition as the actual transition time of that transition during the model cleaning operations. This explicitly ensures that no unstable system state can remain in the DSVTF model because of the variation in transition time instances. The user-defined time threshold value of rule 1 (respectively, rule 2) is set in such a way that a larger threshold value is assigned if the transition path (respectively, transition) has high-variance or high-value transition path times (respectively, transition times). The user-defined extra allowed time Δ (see Fig. 4) is also set following the same procedure. The TTO times of the transitions are calculated after all the model cleaning operations and in that case, the maximum value of the transition time instances of a transition is considered as the actual transition time of that transition (the rationale is easy to understand). Anyhow, the final DSVTF model can be further manually reviewed and all the necessary corrections can be carried out to eliminate the possibility of any structural flaw.

The overall characteristics of the small and the large manufacturing systems are the same. The only difference is that a large manufacturing system can have some unstable system states. So, a small manufacturing system can be thought of as a large manufacturing system (incidentally, having only the stable system states). Hence, all the methods proposed here for large manufacturing systems are equally applicable to the small manufacturing systems. In some articles (see for instance [10] and [13]), it is argued that a centralized FDI approach can lead to a state explosion problem for large-sized PLC-controlled systems. In theory, it is possible to have 2^N number of states for N number of I/O signals (the value of N typically ranges from a few dozen to a very few thousands depending on the I/O capacity of the physical PLC hardware). However, we have empirically observed that the state complexity of the DSVTF model of a PLC-controlled manufacturing process is usually O(kN), where k is a very small positive number. The main reason of it is that a PLC-controlled manufacturing process is a repetitive process (implies that the same set of operations are repeated over and over) and each operation is carried out by only a fixed set of I/O signals (implies that the other signal values remain unchanged during that operation). FBMTP puts a more restrictive limit on it by eliminating the redundant transitions, redundant transition paths and unstable system states from the DSVTF model (these model cleaning operations are executed periodically in order to ensure that the size of the DSVTF model does not grow unbounded). The authors of this paper assume that some approaches have encountered the state explosion problem because, they cannot handle the log data inaccuracy issues. However, as discussed above, FBMTP can easily overcome that fundamental problem.

C. Incorporating Missing Information Associated With Multiple Transitions

In previously described DSVTF models, the transition execution pattern information associated with the multiple


RE(X4) represents the regular expression attached with system state X4
X3
Transition Path shown in brown colour is an allowed Transition Path
S1_OFF (t2+Δ)
0th Transition
X4
RE(X4) = (01)*
1st Transition
S2_ON (t3+Δ)
S2_OFF (t4+Δ)
S3_ON (t5+Δ)
X5
X6
Correct Transition Path: ... X3→X4→X5→X4→X6 ...
X3
S1_OFF (t2+Δ)
X4
S2_ON (t3+Δ)
X5
S2_OFF (t4+Δ)
X4
S3_ON (t5+Δ)
X6
Different States appeared before System State X4
Inequivalent System States
Transition Path of NDAAO Model
This path is executed first (for k number of times)
0th Transition
A
B
D
F
H
I
1st Transition
C
E
G
RE(A) = (0^k1)*
This path is executed next
RE(A) represents the regular expression attached with system state A

(a)

(b)

(c)

Fig. 7. Example scenarios of the multiple transitions from a state. (a) Relevant part of the DSVTF model of Fig. 4. (b) NDAAO model representation proposed by Roth *et al.* [23]. (c) More complex scenario of the multiple transitions from a state.

transitions from a state is not included. So, the model cannot explicitly determine which transition the system should execute at a given time. For example, consider the DSVTF model of Fig. 4. The relevant part of it is shown in Fig. 7(a). As we can see, this DSVTF model classifies the transition path $X3 \rightarrow X4 \rightarrow X6$ as a valid transition path, which is incorrect. Most of the previous works do not address this issue. In [23], a method was taken that generates a NDAAO model based on the concept of the "equivalent states" to avoid this kind of problems. It determines if the same states are equivalent or not by their preceding state sequence of length, say, L_{ps}; and if they are not equivalent, then are kept in the model as separate states. As an example, in the original state sequence [see Fig. 3(a)], state $X3$ appeared before state $X4$ in one case and in another case, state $X5$ appeared before state $X4$. So, those two state $X4$ are not categorized as equivalent states (suppose, $L_{\text{ps}} = 1$) and hence, the NDAAO model generates only one transition path as shown in Fig. 7(b). This strategy is able to fix the above-mentioned issue. However, in practice, it is hard to determine the exact value of L_{ps} for large-sized systems. In addition, the multiple transitions from a state are not always as simple as depicted in Fig. 7(a). For example, consider the scenario of Fig. 7(c). If we apply the above strategy in this case, it generates a large number of redundant states and transitions (assume that the value of L_{ps} is set very high). Moreover, if the k value is very high then it can lead to a state or space explosion problem.

FBMTP takes a novel regular expression based strategy to deal with this kind of situations. Let us assume that all the transitions from a state are numbered, starting from zero according to their appearance [see Fig. 7(a) and (c)]; and a buffer is maintained at each state (from which multiple transitions are possible) to keep track of the transition occurrences (the buffer holds the identification numbers of the observed transitions). Then, the element set of the buffer maintained at state $X4$ of Fig. 7(a) over a period of time can be expressed as: $\text{ESB}(X4) = \{\varepsilon, 0, 01, 010, 0101, \ldots\}$ where, the symbol "ε" denotes an empty buffer and the comma separated items represent the distinct element sets of that buffer at various time points. For example, the buffer is empty at the beginning and hence, the first item of $\text{ESB}(X4)$ is ε. After the occurrence of 0th transition, the buffer holds only one element, i.e., 0 which is the second item of $\text{ESB}(X4)$. Then, the first transition takes place and the buffer becomes a set of two elements, i.e., 0 and 1, represented by "01," which is the third item of $\text{ESB}(X4)$ (and so on). The distinct element sets of the buffer observed over a period of time at state $X4$, i.e., $\text{ESB}(X4)$ can be thought of as a regular language over an alphabet $\Sigma = \{0, 1\}$ where, Σ denotes the set of transition identification numbers and the comma separated items denote the finite length strings over Σ [5], [33], [34]. It is easy to see, the strings of the language $\text{ESB}(X4)$ follow a repetitive pattern and hence, the language can easily be represented by a simple regular expression, i.e., $(01)^*$ where, the symbol "$*$" denotes the Kleene closure operator [5], [33], [34]. Similarly, in case of the multiple transitions from state A of Fig. 7(c), the regular expression that defines the language ESB(A) is $(0^k1)^*$.

FBMTP determines this kind of regular expression for each state from which multiple transitions are possible and attaches them to their corresponding DSVTF model nodes for future use [see Fig. 7(a) and (c)]. By using the regular expressions, FBMTP can detect precisely whether the system has executed any incorrect transition. As an example, suppose, the transition path $X3 \rightarrow X4 \rightarrow X6$ is wrongly executed during the second conveyor belt system cycle [see Figs. 3 and 7(a)]. The string generated by that event (i.e., the element set of the buffer at that particular time point) is "011." This causes a mismatch with its corresponding regular expression [i.e., $(01)^*$] by which

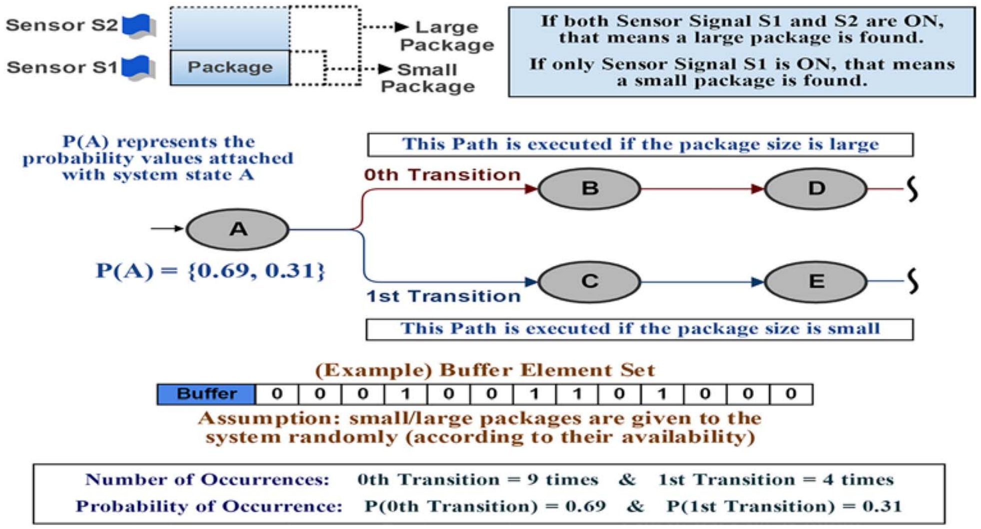

Sensor S2
Sensor S1
Package
Large Package
Small Package
If both Sensor Signal S1 and S2 are ON, that means a large package is found.
If only Sensor Signal S1 is ON, that means a small package is found.
P(A) represents the probability values attached with system state A
This Path is executed if the package size is large
0th Transition
A
B
D
C
E
P(A) = {0.69, 0.31}
1st Transition
This Path is executed if the package size is small
(Example) Buffer Element Set
Buffer 0 0 0 1 0 0 1 1 0 1 0 0 0
Assumption: small/large packages are given to the system randomly (according to their availability)
Number of Occurrences: 0th Transition = 9 times & 1st Transition = 4 times
Probability of Occurrence: P(0th Transition) = 0.69 & P(1st Transition) = 0.31

Fig. 8. Example scenario of the multiple transitions from a state without any execution pattern.

FBMTP can easily detect the fault [see Fig. 7(a)]. Here, we assumed that the strings of the language ESB over the alphabet Σ always follow a repetitive pattern and hence, FBMTP can easily define that language by using a simple regular expression. Unfortunately, this is not always the case. For example, consider a scenario where a robot picks a package and places it into a container in between two containers according to its size (say, small or large). An example of such scenario is presented in Fig. 8. It is easy to perceive from the element set of the buffer, the strings do not follow any repetitive pattern. This is because, the transitions from state A are executed dynamically based on the package size. In this type of situations, FBMTP calculates the probability of occurrence for each such transitions by using the buffer element set and attaches them to their corresponding node (see Fig. 8). The transition probabilities must be calculated based on a large sample size (or based on a big log data) thus they can truly represent the transition execution behavior. Moreover, FBMTP may infer some incorrect rules (or regular expressions) if the sample size is too small.

With these transition probability values, it cannot be determined exactly which transition path the system should execute at a given time (because of the dynamic path selection mechanism) and hence, some soft faults can remain undetected. For example, suppose, the sensor signal S2 of Fig. 8 goes to OFF state permanently due to a fault. In this case, the control process automatically categorizes any package as a small package, irrespective of its size (implies that the first transition from state A is executed repeatedly); and FBMTP cannot detect that fault using the probability values. There is no algorithmic way to identify this type of soft faults because, the control process can perfectly cope with such faults (the process behavior does not perceptibly change). However, FBMTP can identify several behavioral anomalies associated with the transition execution behavior using the probability values. As an example, suppose in normal behavior, the number of small packages given to the system is much lower than the number of large packages. If the number of small packages becomes high then that can be seen as an anomaly in the control process behavior. FBMTP can easily identify this kind of anomalies from the changes in the transition probability values. The above-mentioned failure in sensor S2 (a soft fault) can also be detected using the probability values. However, FBMTP detects it as a behavioral anomaly during the periodical checking of transition probability values. This is because, it causes a very high increment in the probability value corresponding to the first transition from state A. If this type of soft faults do not bring any changes in short or long term behavior of the control process, then they may remain undetected. However, the probability of occurrence of such incidents is significantly low because, usually, a PLC-controlled manufacturing process is a highly ordered and sequential process.

V. Fault and Behavioral Anomaly Detection and Isolation Procedure of FBMTP

This section is divided into three sections. In Section V-A, the BADI procedure of FBMTP is presented. The fault detection and the fault isolation procedure of FBMTP are discussed in Sections V-B and V-C, respectively.

A. Behavioral Anomaly Detection and Isolation Procedure of FBMTP

A DSVTF model is basically a set of transition rules. From the computer science point of view, the FBADI problem can be thought of as a mapping of two databases: one database, i.e., reference database from which the DSVTF model is built and another database, i.e., query database in which FBMTP has to search the faults and behavioral anomalies. It is easy to realize from the DSVTF model layout, FBMTP can detect two types of behavioral anomalies: 1) *transition time error:* this type of anomaly occurs if a transition (found in the query database) takes more time than its corresponding TTO time (see Fig. 4) and 2) *transition probability error:* this type of anomaly occurs if significant changes (user-defined) in the transition probabilities are observed. The transition probabilities are inspected periodically after processing a large number of log data records from the query database (this ensures that the system has already moved into a steady state of operation). Recall from Section IV-B2, that in our DSVTF model, some information about transition times can be lost. This is because,

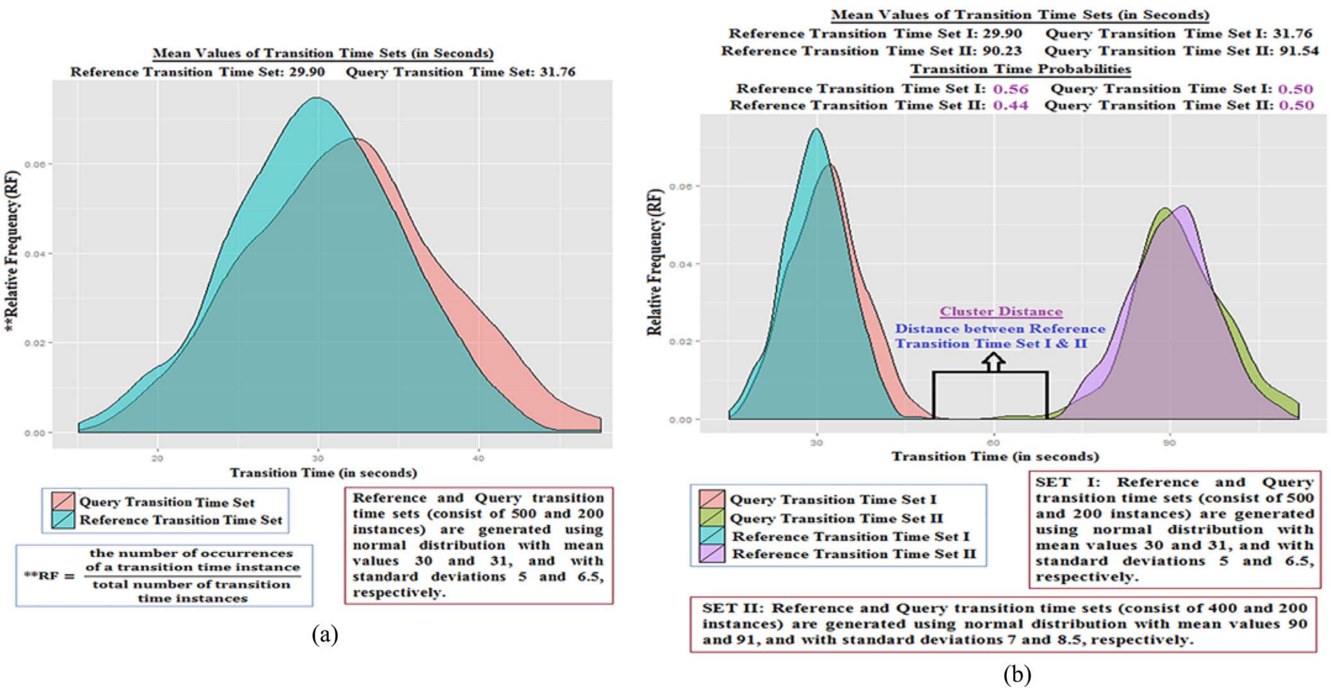
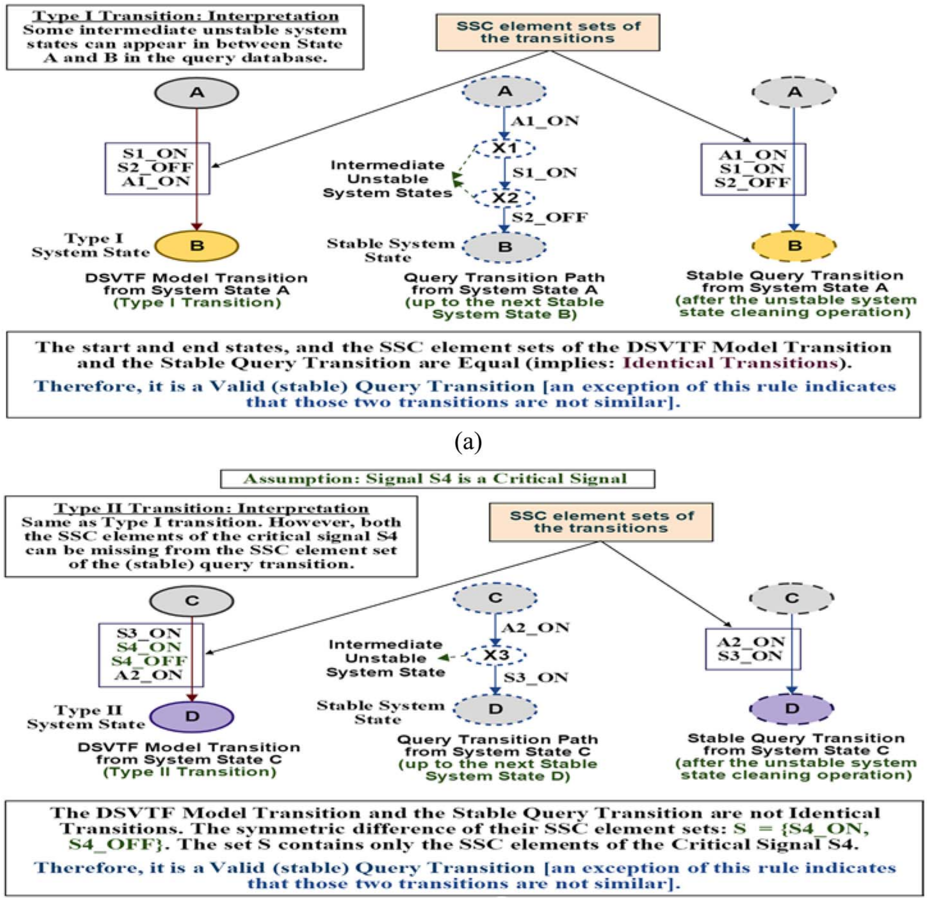


Fig. 9. Graphical plotting of the transition time sets. (a) Transition with a single transition time. (b) Transition with multiple (two) transition times.

the transition times of a particular transition are represented by a single TTO time (for the sake of speed of the fault and behavioral anomaly searching procedure) and hence, if the transition times vary immensely then a single TTO time cannot able to depict that behavior. The transition time instances of a transition can fluctuate greatly mainly because of two reasons: first, in cases where the transition times are dependent on the operators' inputs and second, if the transition actually has more than one transition time. As an example, consider a steel production plant scenario, where a bridge crane is being used to carry two types of steel billets (say, light and heavy) from one location to another location. It is easy to perceive, this state transition can have two highly varied transition times: one for light and another for heavy billets. However, this information will be lost from the DSVTF model because of the single TTO time. So, the transitions with high-variance (user-defined) transition times are needed to be inspected separately in order to overcome this shortcoming.

After processing a large number of log data records from the query database, the reference and query transition time set (transition times found in the reference and query database) of a transition with high-variance transition times are plotted in a relative frequency versus transition time graph; thus the user can find out visually whether any significant changes in the transition time behavior has occurred. An example of such graph plotting is shown in Fig. 9(a). It is easy to see, an user can easily identify any significant variations in those two time sets from this graph and its associated mean values. As argued previously, a transition can have multiple transition times. An example of such scenario is presented in Fig. 9(b) (note that the transition time probabilities are also used to define the behavior). In a large-sized manufacturing system, it is difficult to know how many transition times a particular transition actually has and hence, a transition time clustering approach is needed (a low-variance transition time set is basically considered as a single cluster). FBMTP uses a variation of k-means algorithm, called "Ckmeans.1d.dp" [35] for this purpose. The Ckmeans.1d.dp algorithm is used because, it is particularly developed for 1-D data and it guarantees optimality. The complete clustering algorithm is set in such a way that it starts with a little high number of clusters (obtained using Ckmeans.1d.dp) and the clusters are merged subsequently if their distances are not so high (in an user-defined way). Please note that the reference and query transition time set of a transition are plotted in the graph only if the corresponding cluster means or transition time probabilities vary significantly (this checking is done periodically). Some "true" transition time clusters can be wrongly merged into a single cluster if they are not well-separated. Similarly, a single transition time cluster can be wrongly split up into different clusters if its instances vary significantly. There is no way for an automated tool to identify such cases without any domain knowledge. However, the incorrect transition time clusters do not have much effect on the behavioral anomaly detection accuracy. This is because, the purpose of clustering is to place a strict time bound on each transition time thus any significant deviation in the behavior of the time instances can be identified. As the clusters are built based on the proximity of the time instances, the incorrect clusters do not have much impact on that purpose. If a behavioral anomaly is detected at the end of a state transition, then the device components (i.e., the sensors and actuators) linked with that transition operation (in other words, the device components that carry out the corresponding mechanical operation) are inspected closely to find out the exact defective device component/s. As an example, if the transition from state $X4$ to $X6$ (see Figs. 3 and 4) takes longer time than $(t5 + \Delta)$ (implies that a transition time error has occurred), then the actuator A, sensors S2 and S3 are needed to be examined by


Type I Transition: Interpretation
Some intermediate unstable system states can appear in between State A and B in the query database.
SSC element sets of the transitions
A
S1_ON
S2_OFF
A1_ON
Type I System State
B
DSVTF Model Transition from System State A (Type I Transition)
A1_ON
X1
S1_ON
X2
S2_OFF
Intermediate Unstable System States
Stable System State
Query Transition Path from System State A (up to the next Stable System State B)
A1_ON
S1_ON
S2_OFF
Stable Query Transition from System State A (after the unstable system state cleaning operation)
The start and end states, and the SSC element sets of the DSVTF Model Transition and the Stable Query Transition are Equal (implies: Identical Transitions).
Therefore, it is a Valid (stable) Query Transition [an exception of this rule indicates that those two transitions are not similar].
Assumption: Signal S4 is a Critical Signal
Type II Transition: Interpretation
Same as Type I transition. However, both the SSC elements of the critical signal S4 can be missing from the SSC element set of the (stable) query transition.
SSC element sets of the transitions
C
S3_ON
S4_ON
S4_OFF
A2_ON
Type II System State
D
DSVTF Model Transition from System State C (Type II Transition)
A2_ON
X3
S3_ON
Intermediate Unstable System State
Stable System State
Query Transition Path from System State C (up to the next Stable System State D)
A2_ON
S3_ON
Stable Query Transition from System State C (after the unstable system state cleaning operation)
The DSVTF Model Transition and the Stable Query Transition are not Identical Transitions. The symmetric difference of their SSC element sets: S = {S4_ON, S4_OFF}. The set S contains only the SSC elements of the Critical Signal S4.
Therefore, it is a Valid (stable) Query Transition [an exception of this rule indicates that those two transitions are not similar].

(a)

(b)

Fig. 10. Query transition matching procedure. (a) Query transition matching procedure for type I and normal DSVTF model transitions. (b) Query transition matching procedure for type II DSVTF model transitions.

the engineers in order to find out the exact defective device component/s.

B. Fault Detection Procedure of FBMTP

A fault occurs in a PLC-controlled system if the state transition behavior of the control process model found in the query database is not exactly similar to the state transition behavior of the corresponding DSVTF model (see for instance [21]–[25]). More specifically, two types of faults can take place at a particular system state. The first category of faults occurs if the system falls in an infinite loop at a state. As an example, suppose, the sensor signal S1 of Fig. 3(a) goes to OFF state permanently at the beginning of a system cycle due to a fault. It is easy to see from Fig. 4, the system falls in an infinite loop at state $X1$ for that reason. FBMTP can easily detect such faults by inspecting the TTO times of the corresponding transitions. For example, suppose, there exists three transitions from a state with TTO times $t1$, $t2$, and $t3$. We can safely assume that the system has fallen in an infinite loop if it has already spent more time than (maximum value of TTO times $t1$, $t2$, and $t3+\ell$) in that state where, ℓ is an user-defined high-valued time parameter [in this paper, this time is referred to as the "state expiry time"; and the value of ℓ is set following the same procedure as described for the TTO times (see Section IV-B2)]. The second category of faults occurs if an invalid transition from a state takes place. For example, let us assume that the sensor signal S2 of Fig. 3(a) goes to ON state at the beginning of a system cycle due to a fault. In this case, we observe a transition $X1 \rightarrow_{S2\_\mathrm{ON}} F$ (suppose, the resulted faulty state is F) in the query database, which is not similar to any transition from state $X1$ of the DSVTF model of Fig. 4 (implies that it is an invalid transition). So, a fault is detected at state $X1$.

A query transition (transition found in query database) is similar to any DSVTF model transition (i.e., a valid transition) or not depends on the type of the possible transitions from the corresponding state of the DSVTF model. Recall from Section IV-B that the DSVTF model has three types of transitions, i.e., type I, type II, and normal transition (a normal transition refers to any transition except the type I and type II transitions; and the end node of a normal transition is referred to as the "normal system state"). The query transition matching procedure for type I and type II DSVTF model transitions are shown in Fig. 10(a) and (b), respectively. Recall from Section IV-B, that some intermediate unstable system states can appear in between the start and end states of a type I or type II transition. So, FBMTP always has to wait until a stable system state is found in the query database [this also ensures that the start and end states are not partially observed states—see Fig. 10(a) and (b)]. Then, FBMTP has to apply the unstable system state cleaning operation in order to get a

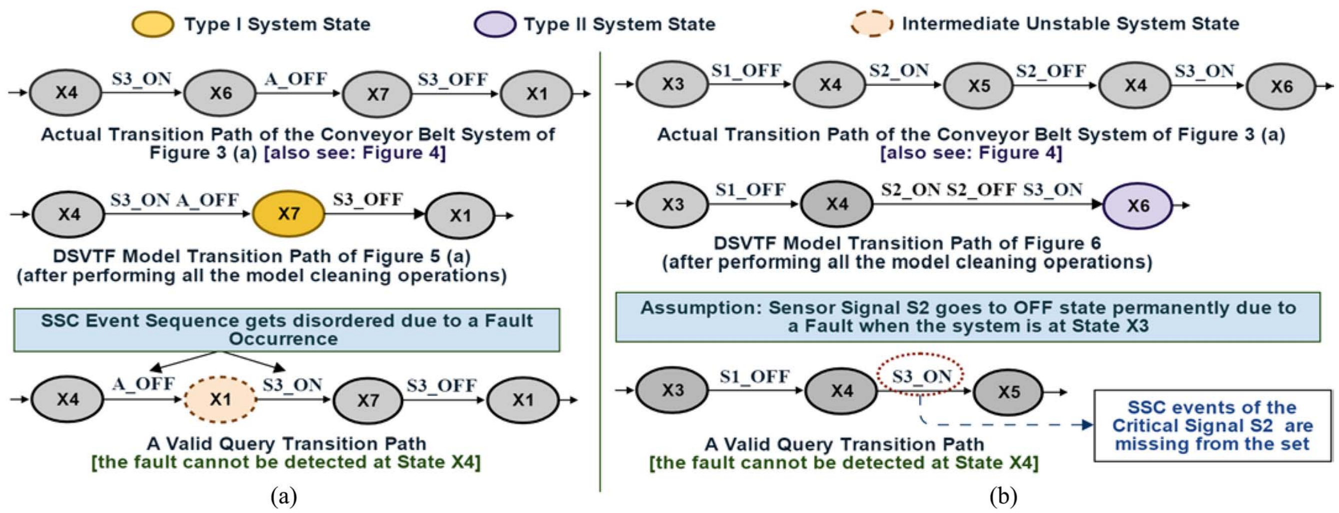

Type I System State
Type II System State
Intermediate Unstable System State
X4
S3_ON
X6
A_OFF
X7
S3_OFF
X1
Actual Transition Path of the Conveyor Belt System of Figure 3 (a) [also see: Figure 4]
X4
S3_ON A_OFF
X7
S3_OFF
X1
DSVTF Model Transition Path of Figure 5 (a) (after performing all the model cleaning operations)
SSC Event Sequence gets disordered due to a Fault Occurrence
X4
A_OFF
X1
S3_ON
X7
S3_OFF
X1
A Valid Query Transition Path [the fault cannot be detected at State X4]
(a)
X3
S1_OFF
X4
S2_ON
X5
S2_OFF
X4
S3_ON
X6
Actual Transition Path of the Conveyor Belt System of Figure 3 (a) [also see: Figure 4]
X3
S1_OFF
X4
S2_ON S2_OFF S3_ON
X6
DSVTF Model Transition Path of Figure 6 (after performing all the model cleaning operations)
Assumption: Sensor Signal S2 goes to OFF state permanently due to a Fault when the system is at State X3
X3
S1_OFF
X4
S3_ON
X5
SSC events of the Critical Signal S2 are missing from the set
A Valid Query Transition Path [the fault cannot be detected at State X4]
(b)

Fig. 11. Examples of the undetectable soft faults. (a) SSC event sequence is disordered due to a fault occurrence. (b) Failure occurs in the sensor S2.

stable query transition (a query transition between two stable system states), in the same way as described in Section IV-B1 [see Fig. 10(a) and (b)]. The stable query transition is then compared with the corresponding DSVTF model transitions to determine their similarities. As can be seen from Fig. 10(a), in case of type I DSVTF model transitions, the stable query transition and the DSVTF model transition must have to be identical in order to be "similar." However, in case of type II DSVTF model transitions, the SSC elements of the critical signals can be missing from the SSC element set of the stable query transition [as stated earlier—also see Fig. 10(b)]. The normal transitions are compared in the same way as type I transitions are compared. Please note that in case of normal transitions, while processing the reference database, FBMTP has not found any intermediate unstable system states. However, it does not necessarily signify that FBMTP will not encounter such intermediate states in the query database (recall from Section IV-B1 that if the signal data sampling rate is low then the data logger may produce some partially observed states while trying to capture a stable system state). If a stable query transition from a particular state is not similar to any transition from that state of the DSVTF model, then we can easily conclude that a fault has occurred at that state. In case of the multiple transitions from a state, if the transition execution behavior is represented by a regular expression, then an exception of that rule also indicates that a fault has taken place at that state (see Section IV-C for details). It is easy to perceive that this query transition matching process follows exactly the same principles as outlined in Sections IV-B and IV-C.

It is easy to understand, a fault that has significant impact on system operations, i.e., a hard fault cannot remain undetected in FBMTP (provided that the TIB time is reasonable). This is because, a system cannot execute all the state transitions correctly even after the occurrence of a hard fault (in other words, a hard fault will eventually be detected since the observed faulty behavior will eventually be distinguishable from the fault-free nominal behavior) [22], [23]. However, it is possible that FBMTP may fail to detect some of the soft faults (false negatives) and as argued previously, the TIB time has a direct effect on it. If the TIB time is high then some stable system states become unstable and hence, are discarded from the DSVTF model. This increases the number of SSC events attached with the corresponding transitions. In addition, some normal signals can become critical signals because of the high TIB time. Some soft faults can remain unidentified as a consequence of it. More specifically, if a soft fault causes disorder in the SSC event sequence in such a way that it can also get generated because of some log data inaccuracy issues, then that soft fault can go undetected. An example of such incident is given in Fig. 11(a). Similarly, if the SSC events of the critical signals associated with a type II transition do not get executed due to the occurrence of a soft fault, then that soft fault can also remain unidentified. An example of such scenario is presented in Fig. 11(b). These types of false negative cases take place because, it cannot be determined, based only on the available data, whether such incidents have occurred due to the control process faults or due to the log data inaccuracy issues. However, the probability of occurrence of such incidents is extremely low in practice. This is because, first, most of the state transitions take very long time to execute in a real-world manufacturing system. So, only a very few SSC events are attached with each transition and only a very few signals actually become critical signals. Second, a system component failure occurs randomly and independently (not necessarily in between a specific short time range that makes the soft fault undetectable). Third and most importantly, (usually) a PLC-controlled manufacturing system is a strongly coupled system and its corresponding control process is a step-by-step structured process. So, if the SSC events are really disordered due to a fault occurrence or some critical signals actually become faulty, then it has serious consequences on the rest of the process [in Fig. 11(b), the fault remains undetected because, the sensor S2 is functionally a standalone component and therefore, the corresponding signal does not have any impact on the rest of the process [also see Fig. 3(a)]; and the fault given in Fig. 11(a) simply cannot occur in reality].

FBMTP does not produce any false positives if the DSVTF model is fully converged. The model convergence problem can largely be avoided if the DSVTF model is built on a large reference database. However, sometimes, the DSVTF model cannot converge completely because of the system alarms. In the manufacturing systems, the system alarms are often used to handle the irregular situations [25]. As an example, if a stepper motor gets overheated, the corresponding system alarm is enabled thus the protective operations (such as, the coolant circulation) can be carried out. As this type of situations does not occur frequently, its corresponding transition path may not appear in the reference database. FBMTP classifies such missing true transitions as the invalid transitions or faults. When the operations associated with the system alarms are executed, the system returns back to the normal transition path after some time. Moreover, it switches ON/OFF a new set of signals which have never been turned ON/OFF previously. For this reason, FBMTP does not halt its execution immediately after identifying a fault. It waits for some user-defined time period to determine whether the transition path is generated because of the execution of the operations associated with the system alarms (using the above-defined criteria). If so, then that transition path is included in the DSVTF model (with operator confirmation). So, the model building procedure of FBMTP is an incremental procedure (not a one-time task).

C. Fault Isolation Procedure of FBMTP

Once a fault is detected at a state, the next step is to isolate the PLC I/O signal set that could be the reason for the fault. FBMTP uses a stricter and more accurate version of the residual-based fault isolation procedure proposed in [21]–[23] for this purpose (actually, the stated fault isolation method is slightly modified in order to deal with the log data inaccuracy issues and the infinite looping problem). Recall from previous section that two types of faults can actually take place at a particular system state: 1) a fault with a faulty transition (the term "faulty transition" refers to an invalid stable query transition) and 2) a fault without a faulty transition (implies that the system gets stuck in an infinite loop). FBMTP handles each of them differently. We discuss the fault isolation procedure of FBMTP with the help of an example manufacturing system and its corresponding DSVTF model, as shown in Fig. 12 [It should be noted that in the following discussion (also in Fig. 12), we use the generic term "transition time" in order to keep the discussion simple and easier to follow. At the time of practical application, the maximum transition time (or for that matter, TTO time, average transition time, etc.) can be used as the actual transition time (only the time threshold parameters are required to be tuned accordingly)]. The rest of this section is organized into three sections. The first two sections describe the fault isolation procedure for the above-defined two types of faults and the last section discusses the effect of the TIB time on the accuracy of the fault isolation procedure.

1) Fault Isolation Technique for Fault With Faulty Transition: FBMTP determines the initial fault candidate set in case of a fault with a faulty transition by using following residual-based formula (also see [21]–[23]).

Rule 3: Suppose, a fault is detected at state x_a and the faulty transition is $\boldsymbol{x}_a \rightarrow_{e_{ab}} \boldsymbol{x}_b$ where, state x_b is the immediate next stable system state of state x_a; then

$$\text{initial fault candidate set} = <\boldsymbol{e}_{ab}> \bigcup \bigcup_{\forall e_{ai} \in \Gamma(x_a)} <\boldsymbol{e}_{ai}>.$$

[<e> represents the SSC element set of the signal-status vector e; and $\Gamma(x)$ represents the set of all signal-status vectors of all the possible DSVTF model transitions from state x (see Definitions 1 and 2 in Section IV-A).]

Let us assume that the sensor signal S2 of Fig. 12 goes to OFF state permanently at the beginning of a system cycle due to a fault. In this case, we observe a faulty transition at state $X3$ as shown in Fig. 13(a). As can be seen in Fig. 13(a), the SSC event of the faulty signal S2 is correctly included in the initial fault candidate set. Actually, if a faulty transition occurs at a state, say x_a, it can happen only because of the following two reasons (see for instance [21]–[23]).

1) *Cause I:* Some unexpected SSC events have occurred in the system that lead the control process to execute the faulty transition (can broadly be represented by the SSC element set of the faulty transition, i.e., $<e_{ab}>$).
2) *Cause II:* The next expected (stable) DSVTF model states cannot be reached (or are skipped) because, some of the SSC events associated with those expected state transitions are not executed (can broadly be represented by the union of all the SSC element sets of all the possible DSVTF model transitions from state x_a, i.e., $\cup_{\forall e_{ai} \in \Gamma(x_a)} <e_{ai}>$).

As can be seen from rule 3, the initial fault candidate set is simply the union of all the above-mentioned SSC element sets (see causes I and II) and hence, the SSC event/s of the faulty signal/s can definitely be found in the initial fault candidate set (see [21], [22]). In practice, the initial fault candidate set contains a very small number of SSC elements and hence, an operator can easily locate the actual broken device/s by inspecting the signal names and the corresponding physical devices. In case of the multiple transitions from a state, if the transition execution behavior is characterized by using the probability values, then the SSC element sets associated with all the DSVTF model transitions from that state are included in the initial fault candidate set [see Figs. 12 and 13(a)]. If the transition execution behavior is defined by using the regular expression, then only the SSC element set of the expected DSVTF model transition is included in the initial fault candidate set (basically, the expected DSVTF model transition is considered as only possible transition from that state—this is equally applicable to the other fault candidate sets described later in this section). FBMTP always places the actuator SSC elements at the end of a fault candidate set [see Fig. 13(a)] because, in practice, the occurrence of an actuator failure is much more unlikely than the occurrence of a sensor failure (the actuators are in general very expensive devices).

After identifying the initial fault candidate set, FBMTP tries to determine a more specific fault candidate subset of the initial fault candidate set, called the "exact fault candidate set." Actually, the initial fault candidate set is too broadly defined and hence, can be reduced substantially. The exact fault candidate set is determined based on the most probable cause of the fault, i.e., causes I or II. If a faulty transition occurs due

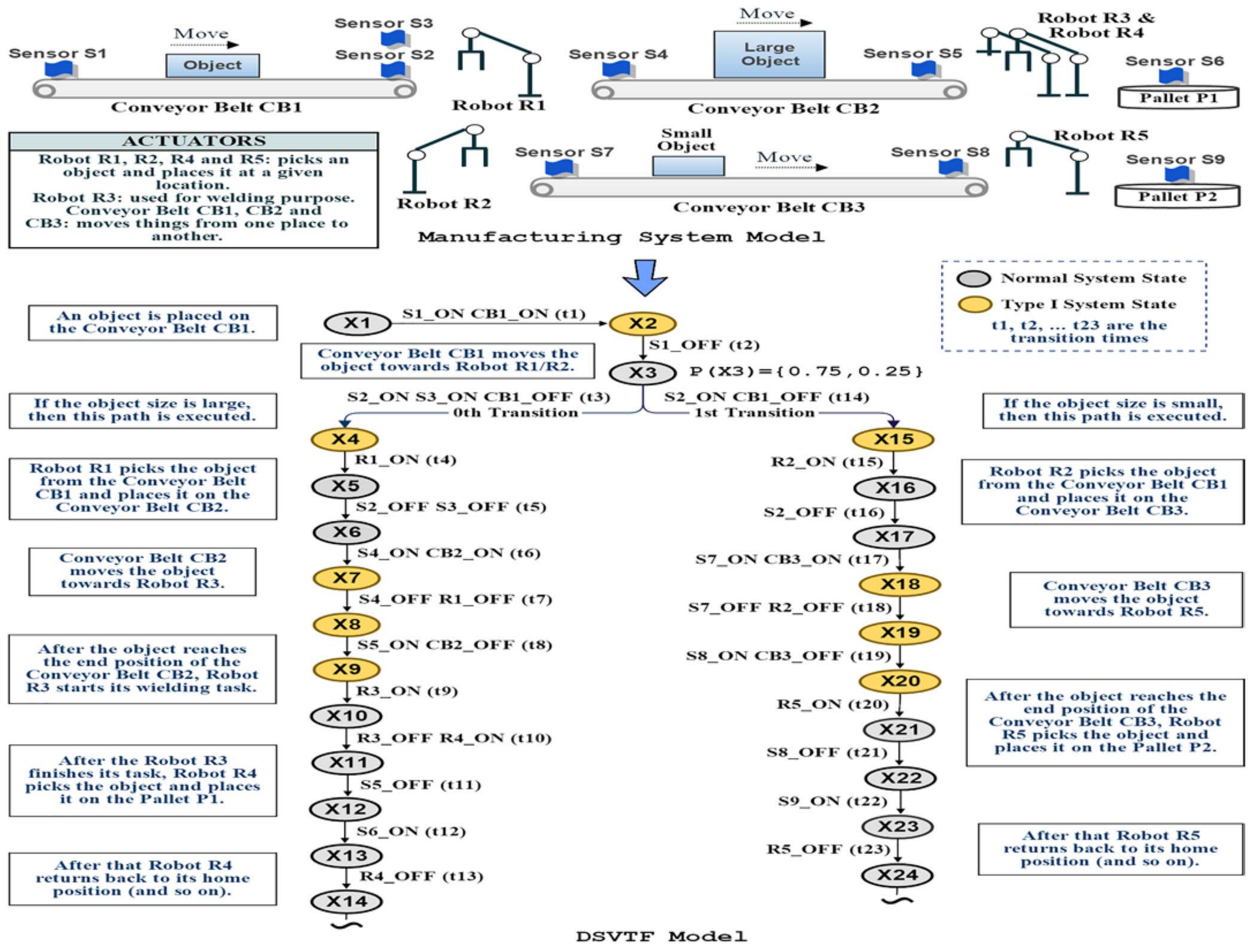

Sensor S1
Move
Object
Sensor S3
Sensor S2
Conveyor Belt CB1
Robot R1
Move
Large Object
Sensor S4
Sensor S5
Conveyor Belt CB2
Robot R3 & Robot R4
Sensor S6
Pallet P1
ACTUATORS
Robot R1, R2, R4 and R5: picks an object and places it at a given location.
Robot R3: used for welding purpose.
Conveyor Belt CB1, CB2 and CB3: moves things from one place to another.
Robot R2
Sensor S7
Small Object
Move
Sensor S8
Robot R5
Sensor S9
Pallet P2
Conveyor Belt CB3
Manufacturing System Model
Normal System State
Type I System State
t1, t2, ... t23 are the transition times
An object is placed on the Conveyor Belt CB1.
X1
S1_ON CB1_ON (t1)
X2
S1_OFF (t2)
Conveyor Belt CB1 moves the object towards Robot R1/R2.
X3
P(X3)={0.75,0.25}
S2_ON S3_ON CB1_OFF (t3)
S2_ON CB1_OFF (t14)
0th Transition
1st Transition
If the object size is large, then this path is executed.
If the object size is small, then this path is executed.
X4
R1_ON (t4)
X5
S2_OFF S3_OFF (t5)
X6
S4_ON CB2_ON (t6)
X7
S4_OFF R1_OFF (t7)
X8
S5_ON CB2_OFF (t8)
X9
R3_ON (t9)
X10
R3_OFF R4_ON (t10)
X11
S5_OFF (t11)
X12
S6_ON (t12)
X13
R4_OFF (t13)
X14
X15
R2_ON (t15)
X16
S2_OFF (t16)
X17
S7_ON CB3_ON (t17)
X18
S7_OFF R2_OFF (t18)
X19
S8_ON CB3_OFF (t19)
X20
R5_ON (t20)
X21
S8_OFF (t21)
X22
S9_ON (t22)
X23
R5_OFF (t23)
X24
Robot R1 picks the object from the Conveyor Belt CB1 and places it on the Conveyor Belt CB2.
Conveyor Belt CB2 moves the object towards Robot R3.
After the object reaches the end position of the Conveyor Belt CB2, Robot R3 starts its wielding task.
After the Robot R3 finishes its task, Robot R4 picks the object and places it on the Pallet P1.
After that Robot R4 returns back to its home position (and so on).
Robot R2 picks the object from the Conveyor Belt CB1 and places it on the Conveyor Belt CB3.
Conveyor Belt CB3 moves the object towards Robot R5.
After the object reaches the end position of the Conveyor Belt CB3, Robot R5 picks the object and places it on the Pallet P2.
After that Robot R5 returns back to its home position (and so on).
DSVTF Model

Fig. 12. Example manufacturing process scenario and its corresponding DSVTF model.

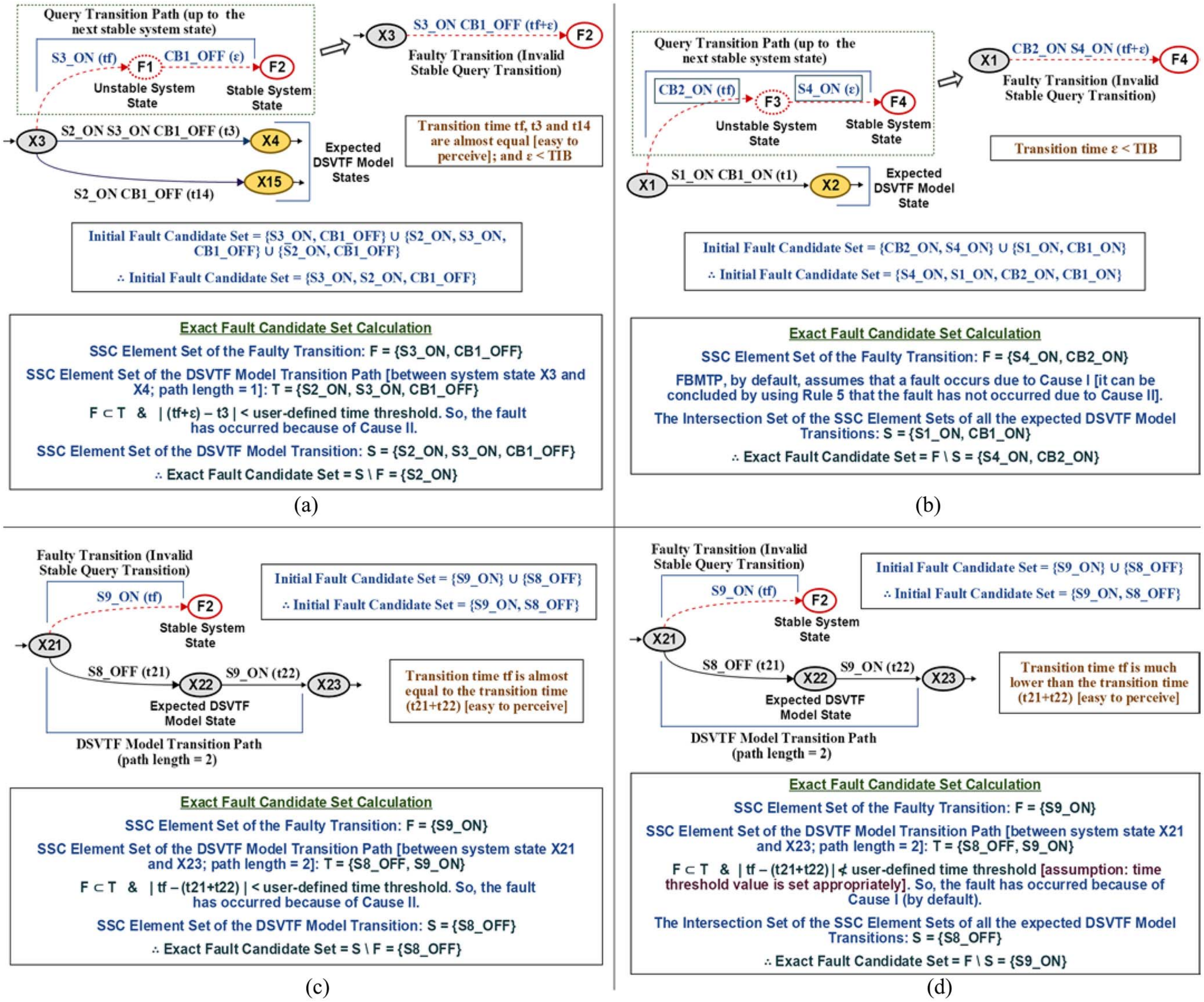
to cause I, then the exact fault candidate set is computed by using the following residual-based formula.

Rule 4: The meaning of the symbols are the same as in rule 3,

$$\text{initial fault candidate set} = <e_{ab}>\setminus \bigcap_{\forall e_{ai}\in\Gamma(x_a)} <e_{ai}>.$$

The second component of rule 4 (i.e., $\cap_{\forall e_{ai}\in\Gamma(x_a)} < e_{ai} >$) characterizes the SSC elements that must appear in the SSC element set of every possible DSVTF model transition from state x_a, irrespective of which transition path is taken to exit the state. So, only the SSC elements of the faulty transition that do not belong to all the DSVTF model transitions (in other words, occur unexpectedly with respect to any DSVTF model transition) are kept in the exact fault candidate set. If a faulty transition really occurs due to cause I, then the SSC event/s of the faulty signal/s must reside within the above-defined exact fault candidate set (also see cause I). As an example, suppose, the sensor signal S4 of Fig. 12 abruptly goes to ON state at the beginning of a system cycle due to a fault. This cause I fault case scenario is presented in Fig. 13(b). As can be seen, the SSC event of the faulty signal S4 is rightly included in the exact fault candidate set (derived by using rule 4). In practice, most of the times, a faulty transition occurs due to cause I because, as stated earlier, a device component failure occurs randomly and independently in a physical manufacturing plant. For this reason, FBMTP by default assumes that a fault occurs due to cause I.

Whether a faulty transition from a state is resulted due to cause II or not can be determined by inspecting the SSC element sets and the transition path times of all possible DSVTF model transition paths from that state. As an example, let us assume that the sensor signal S8 of Fig. 12 gets stuck into ON state permanently due to a fault at the moment when the robot R5 is starting its task. This cause II fault case scenario is given in Fig. 13(c) [recall that the example presented in Fig. 13(a) is also a cause II fault case scenario]. If a faulty transition from a state is resulted due to cause II, then the following two properties must hold (see cause II): 1) the SSC element set of the faulty transition has to be an absolute subset of the SSC element set of one of the DSVTF model transition path of length, say L, from that state and 2) the transition path times of the faulty transition and the corresponding DSVTF model transition path of length L have to be almost equal [they can vary a little because of the timing inaccuracy problem—see Fig. 13(a) and (c)]. By using this intuition, FBMTP decides whether a faulty transition has occurred due to cause II (the rationale is easy to understand). If so, then the exact fault candidate set is determined by using the following residual-based formula.


Query Transition Path (up to the next stable system state)
S3_ON (tf)
F1
CB1_OFF (ε)
F2
Unstable System State
Stable System State
X3
S3_ON CB1_OFF (tf+ε)
F2
Faulty Transition (Invalid Stable Query Transition)
X3
S2_ON S3_ON CB1_OFF (t3)
X4
X15
S2_ON CB1_OFF (t14)
Expected DSVTF Model States
Transition time tf, t3 and t14 are almost equal [easy to perceive]; and ε < TIB
Query Transition Path (up to the next stable system state)
CB2_ON (tf)
F3
S4_ON (ε)
F4
Unstable System State
Stable System State
X1
CB2_ON S4_ON (tf+ε)
F4
Faulty Transition (Invalid Stable Query Transition)
Transition time ε < TIB
X1
S1_ON CB1_ON (t1)
X2
Expected DSVTF Model State
Faulty Transition (Invalid Stable Query Transition)
S9_ON (tf)
F2
Stable System State
X21
S8_OFF (t21)
X22
S9_ON (t22)
X23
Expected DSVTF Model State
DSVTF Model Transition Path (path length = 2)
Transition time tf is almost equal to the transition time (t21+t22) [easy to perceive]
Faulty Transition (Invalid Stable Query Transition)
S9_ON (tf)
F2
Stable System State
X21
S8_OFF (t21)
X22
S9_ON (t22)
X23
Expected DSVTF Model State
DSVTF Model Transition Path (path length = 2)
Transition time tf is much lower than the transition time (t21+t22) [easy to perceive]

Initial Fault Candidate Set = {S3_ON, CB1_OFF} ∪ {S2_ON, S3_ON, CB1_OFF} ∪ {S2_ON, CB1_OFF}

∴ Initial Fault Candidate Set = {S3_ON, S2_ON, CB1_OFF}

Exact Fault Candidate Set Calculation

SSC Element Set of the Faulty Transition: F = {S3_ON, CB1_OFF}

SSC Element Set of the DSVTF Model Transition Path [between system state X3 and X4; path length = 1]: T = {S2_ON, S3_ON, CB1_OFF}

F ⊂ T & | (tf+ε) – t3 | < user-defined time threshold. So, the fault has occurred because of Cause II.

SSC Element Set of the DSVTF Model Transition: S = {S2_ON, S3_ON, CB1_OFF}

∴ Exact Fault Candidate Set = S \ F = {S2_ON}

(a)

Initial Fault Candidate Set = {CB2_ON, S4_ON} ∪ {S1_ON, CB1_ON}

∴ Initial Fault Candidate Set = {S4_ON, S1_ON, CB2_ON, CB1_ON}

Exact Fault Candidate Set Calculation

SSC Element Set of the Faulty Transition: F = {S4_ON, CB2_ON}

FBMTP, by default, assumes that a fault occurs due to Cause I [it can be concluded by using Rule 5 that the fault has not occurred due to Cause II].

The Intersection Set of the SSC Element Sets of all the expected DSVTF Model Transitions: S = {S1_ON, CB1_ON}

∴ Exact Fault Candidate Set = F \ S = {S4_ON, CB2_ON}

(b)

Initial Fault Candidate Set = {S9_ON} ∪ {S8_OFF}

∴ Initial Fault Candidate Set = {S9_ON, S8_OFF}

Exact Fault Candidate Set Calculation

SSC Element Set of the Faulty Transition: F = {S9_ON}

SSC Element Set of the DSVTF Model Transition Path [between system state X21 and X23; path length = 2]: T = {S8_OFF, S9_ON}

F ⊂ T & | tf – (t21+t22) | < user-defined time threshold. So, the fault has occurred because of Cause II.

SSC Element Set of the DSVTF Model Transition: S = {S8_OFF}

∴ Exact Fault Candidate Set = S \ F = {S8_OFF}

(c)

Initial Fault Candidate Set = {S9_ON} ∪ {S8_OFF}

∴ Initial Fault Candidate Set = {S9_ON, S8_OFF}

Exact Fault Candidate Set Calculation

SSC Element Set of the Faulty Transition: F = {S9_ON}

SSC Element Set of the DSVTF Model Transition Path [between system state X21 and X23; path length = 2]: T = {S8_OFF, S9_ON}

F ⊂ T & | tf – (t21+t22) | ≮ user-defined time threshold [assumption: time threshold value is set appropriately]. So, the fault has occurred because of Cause I (by default).

The Intersection Set of the SSC Element Sets of all the expected DSVTF Model Transitions: S = {S8_OFF}

∴ Exact Fault Candidate Set = F \ S = {S9_ON}

(d)

Fig. 13. Example scenarios of a fault with a faulty transition. Failure occurs in the sensor (a) S2, (b) S4, (c) S8, and (d) S9.

Rule 5: Suppose, a faulty transition detected at state x_a is $\boldsymbol{x}_a \rightarrow_{\boldsymbol{e}_{ab}(\boldsymbol{t}_{ab})} \boldsymbol{x}_b$ and the corresponding DSVTF model transition path of length L is

$$\boldsymbol{x}_a \rightarrow_{\boldsymbol{e}_{ak_1}(\boldsymbol{t}_{ak_1})} \boldsymbol{x}_{k_1} \rightarrow_{\boldsymbol{e}_{k_1k_2}(\boldsymbol{t}_{k_1k_2})} \boldsymbol{x}_{k_2} \cdots \rightarrow_{\boldsymbol{e}_{k_{L-1}k_L}\left(\boldsymbol{t}_{k_{L-1}k_L}\right)} \boldsymbol{x}_{k_L}$$

such that: $<\boldsymbol{e}_{ab}> \subset (<\boldsymbol{e}_{ak_1}> \cup <\boldsymbol{e}_{k_1k_2}> \cdots \cup <\boldsymbol{e}_{k_{L-1}k_L}>)$ and $|\boldsymbol{t}_{ab} - (\boldsymbol{t}_{ak_1} + \boldsymbol{t}_{k_1k_2} \cdots + \boldsymbol{t}_{k_{L-1}k_L})| <$ (user-defined) time threshold;

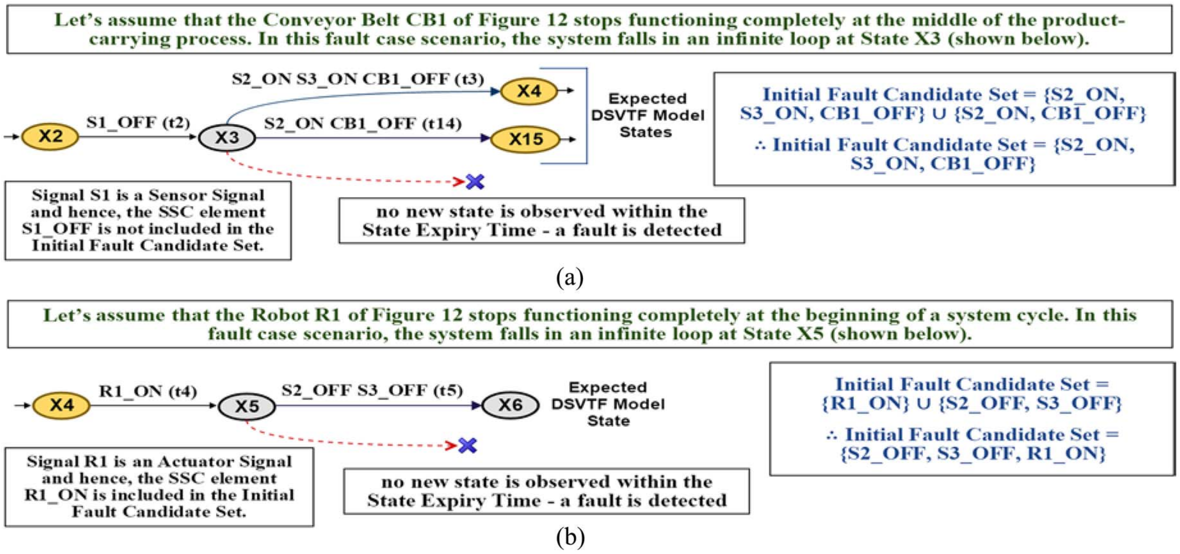
[implies that the fault is generated because of cause II]

then

$$\text{exact fault candidate set} = <\boldsymbol{e}_{ak_1}> \backslash <\boldsymbol{e}_{ab}>$$

[the symbol <e> has the same meaning as in rule 3.]

The set-theoretic difference of the above two sets (i.e., $<e_{ak_1}>$ and $<e_{ab}>$) basically characterizes the expected but missing SSC elements from the SSC element set of the faulty transition. If a faulty transition is actually resulted due to cause II, then the set-theoretic difference must contain the expected SSC events that are not yet executed [in other words, the SSC event/s of the faulty signal/s—see Fig. 13(a) and (c)]. The exact fault candidate set finding method may not always exhibit the correct result mainly because, the appropriate transition path length and the user-defined threshold value are hard to determine in some real-world scenarios [see the differences in Fig. 13(a) and (c)]. As an example, let us assume that the sensor signal S9 of Fig. 12 goes to ON state unexpectedly due to a fault at the moment when the robot R5 has just started its operation. In this cause I fault case scenario [given in Fig. 13(d)], we observe the same faulty transition as in Fig. 13(c). It is easy to realize from Fig. 13(d), the exact fault candidate set finding method may produce an incorrect result if the time threshold value is set too high. FBMTP uses a small time threshold value and examines up to a small path length starting from path length one in order to roughly estimate the most probable cause of the fault (i.e., causes I or II). If the SSC event of any faulty signal is not found in the exact fault candidate set, then the engineer has to examine the remaining SSC elements of the initial fault candidate set. However, we have empirically


Let's assume that the Conveyor Belt CB1 of Figure 12 stops functioning completely at the middle of the product-carrying process. In this fault case scenario, the system falls in an infinite loop at State X3 (shown below).
X2
S1_OFF (t2)
X3
S2_ON S3_ON CB1_OFF (t3)
X4
S2_ON CB1_OFF (t14)
X15
Expected DSVTF Model States
Initial Fault Candidate Set = {S2_ON, S3_ON, CB1_OFF} ∪ {S2_ON, CB1_OFF}
∴ Initial Fault Candidate Set = {S2_ON, S3_ON, CB1_OFF}
Signal S1 is a Sensor Signal and hence, the SSC element S1_OFF is not included in the Initial Fault Candidate Set.
no new state is observed within the State Expiry Time - a fault is detected

(a)

Let's assume that the Robot R1 of Figure 12 stops functioning completely at the beginning of a system cycle. In this fault case scenario, the system falls in an infinite loop at State X5 (shown below).
X4
R1_ON (t4)
X5
S2_OFF S3_OFF (t5)
X6
Expected DSVTF Model State
Initial Fault Candidate Set = {R1_ON} ∪ {S2_OFF, S3_OFF}
∴ Initial Fault Candidate Set = {S2_OFF, S3_OFF, R1_ON}
Signal R1 is an Actuator Signal and hence, the SSC element R1_ON is included in the Initial Fault Candidate Set.
no new state is observed within the State Expiry Time - a fault is detected

(b)

Fig. 14. Example scenarios of a fault without a faulty transition. Failure occurs in the actuator (a) CB1 and (b) R1.

found that most of the times, FBMTP is able to capture the SSC event/s of the faulty signal/s in the exact fault candidate set (provided that the parameter values are set properly).

2) Fault Isolation Technique for Fault Without Faulty Transition: In case of a fault without a faulty transition, the initial fault candidate set is calculated in the same way as described above. The residual-based formula for it is given below.

Rule 6: Suppose, the system gets stuck in an infinite loop at a stable system state x_b and the immediate previous DSVTF model state of it is state x_a; then

$$\text{initial fault candidate set} = \mathrm{ACC}(<e_{ab}>) \cup \bigcup_{\forall e_{bi} \in \Gamma(x_b)} <e_{bi}>$$

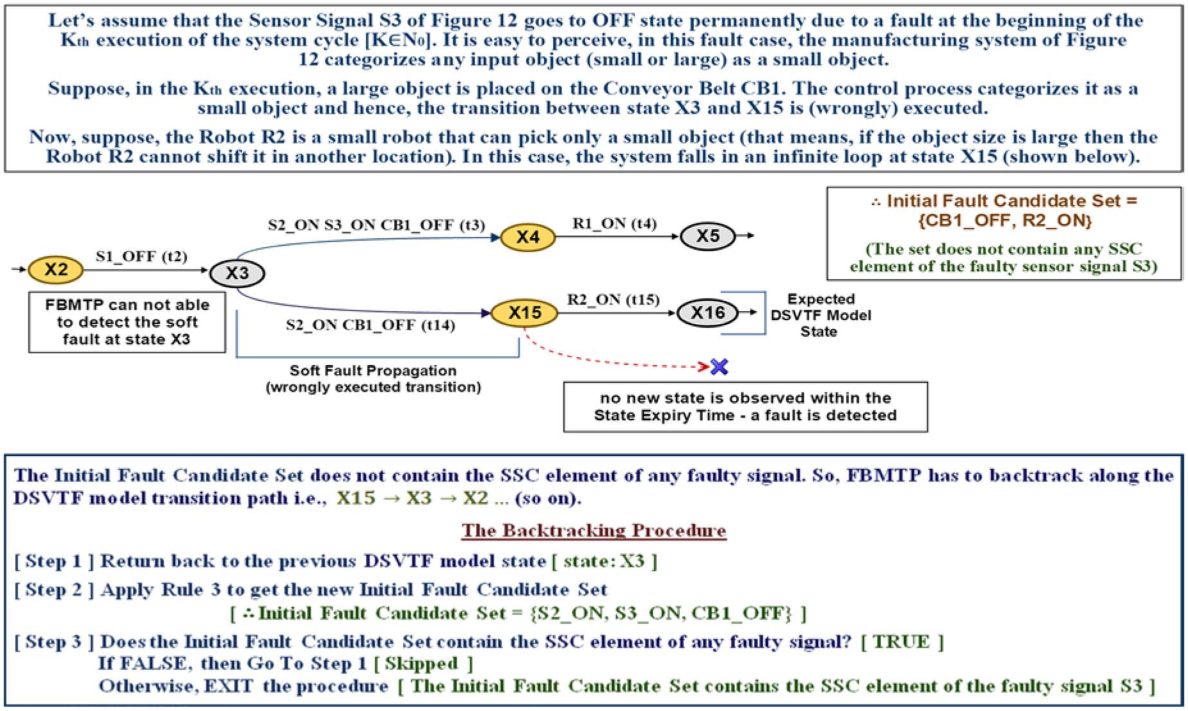
[<e> and Γ(x) have the same meaning as in rule 3; and ACC(<e>) represents the set of all the actuator SSC elements of the signal-status vector e].

Actually, if a system falls in an infinite loop at a stable system state, say x_b, it can happen only because of the following two reasons.

1) *Reason I:* The next expected DSVTF model states cannot be reached because, some of the SSC events associated with those expected state transitions cannot be executed (represented by: $\cup_{\forall e_{bi} \in \Gamma(x_b)} <e_{bi}>$). An example of such scenario is shown in Fig. 14(a).
2) *Reason II:* The state x_b is supposedly reached; however, all the actuator operations associated with the transition $\boldsymbol{x}_a \rightarrow_{\boldsymbol{e}_{ab}} \boldsymbol{x}_b$ are not physically executed correctly (where, state x_a is the immediate previous DSVTF model state of state x_b). An example of such case is given in Fig. 14(b). In this case, the system assumes that the state $X5$ is already reached because, the control process has already switched ON the actuator signal R1. However, technically, the system is at state $X4$ at that moment because of the breakdown of the actuator R1 (an unfinished physical operation). It is easy to perceive, this kind of situations can only happen if some actuator operations associated with the previous DSVTF model transition are not properly executed and hence, the actuator SSC elements associated with the transition $\boldsymbol{x}_a \rightarrow_{\boldsymbol{e}_{ab}} \boldsymbol{x}_b$ are included in the initial fault candidate set [represented by Acc($<e_{ab}>$)]. As can be seen in Fig. 14(b), the SSC event of the faulty actuator signal R1 is rightly included in the initial fault candidate set.

FBMTP calculates only the initial fault candidate set in case of a fault without a faulty transition. Actually, it cannot be determined based only on the log data records whether such faults are generated because of Reasons I or II (that means, an optimal fault candidate set is already derived). Please note that the fault case presented in Fig. 14(a) can also take place if the sensor signals S2 and S3 permanently go to OFF state together at the beginning of a system cycle. Recall from Section III that unlike the sensor breakdown incidents, the actuator breakdown incidents are not directly reflected in the PLC memory table (because, the actuator signals are the output signals). In general, if an actuator failure occurs, the system gets stuck in an infinite loop because, the rest of the system operations cannot be executed without completing that task. So, the fault can easily be detected and isolated (also see Reasons I and II). If an actuator does not have any impact on the rest of the system operations, then a fault in that actuator (that means, a soft fault) can remain unidentified. However, as stated earlier, an actuator in a real-world manufacturing system is usually strongly connected with the rest of the system and hence, an occurrence of such undetectable soft faults is extremely rare in practice.

3) Effect of the TIB Time on the Accuracy of the Fault Isolation Procedure and Its Corresponding Corrective Actions: If the TIB time is high, then it can cause addition of some extra SSC elements into the fault candidate sets because, the high TIB time can increase the number of unstable system states in the model. However, the number of such extra SSC elements will not be so high in practice because, in a real-world manufacturing system, most of the state transitions take much longer time than the TIB time. It is easy to understand from the discussions in Sections V-C1 and V-C2 (also see [21]–[23]), if a fault is detected at the correct state,



Let's assume that the Sensor Signal S3 of Figure 12 goes to OFF state permanently due to a fault at the beginning of the K_{th} execution of the system cycle [K∈N₀]. It is easy to perceive, in this fault case, the manufacturing system of Figure 12 categorizes any input object (small or large) as a small object.

Suppose, in the K_{th} execution, a large object is placed on the Conveyor Belt CB1. The control process categorizes it as a small object and hence, the transition between state X3 and X15 is (wrongly) executed.

Now, suppose, the Robot R2 is a small robot that can pick only a small object (that means, if the object size is large then the Robot R2 cannot shift it in another location). In this case, the system falls in an infinite loop at state X15 (shown below).

X2
S1_OFF (t2)
X3
S2_ON S3_ON CB1_OFF (t3)
X4
R1_ON (t4)
X5
S2_ON CB1_OFF (t14)
X15
R2_ON (t15)
X16
Expected DSVTF Model State
FBMTP can not able to detect the soft fault at state X3
Soft Fault Propagation (wrongly executed transition)
no new state is observed within the State Expiry Time - a fault is detected

∴ Initial Fault Candidate Set = {CB1_OFF, R2_ON}

(The set does not contain any SSC element of the faulty sensor signal S3)

The Initial Fault Candidate Set does not contain the SSC element of any faulty signal. So, FBMTP has to backtrack along the DSVTF model transition path i.e., X15 → X3 → X2 ... (so on).

The Backtracking Procedure

[Step 1] Return back to the previous DSVTF model state [state: X3]

[Step 2] Apply Rule 3 to get the new Initial Fault Candidate Set
[∴ Initial Fault Candidate Set = {S2_ON, S3_ON, CB1_OFF}]

[Step 3] Does the Initial Fault Candidate Set contain the SSC element of any faulty signal? [TRUE]
If FALSE, then Go To Step 1 [Skipped]
Otherwise, EXIT the procedure [The Initial Fault Candidate Set contains the SSC element of the faulty signal S3]

Fig. 15. Transformation of an undetected soft fault into a detectable hard fault.

then the fault isolation procedure of FBMTP always provides an accurate initial fault candidate set (provided that the data logger is able to capture all the faulty SSC events correctly—we will discuss on it in the next paragraph). Most of the times, FBMTP is able to detect a fault (especially, a hard fault) at the correct state because, it is quite exceptional that a query transition exactly similar to the corresponding DSVTF model transition gets generated even after the fault occurrence (assuming that the TIB time is reasonable—that means, the system may at most have a limited number of critical signals). However, sometimes, an undetected fault subsequently gets transformed into a detectable fault along the transition path. In such a case, FBMTP can produce an inaccurate fault candidate set because, the exact state where the fault has actually taken place, is not correctly identified. An example of such scenario is given in Fig. 15. As can be seen, an undetectable soft fault that occurred at state $X3$, subsequently gets transformed into a detectable hard fault along the transition path. FBMTP is able to detect that (hard) fault at state $X15$, which is not the exact state where the core issue lies. For this reason, an initial fault candidate set is produced that does not contain any SSC event of the faulty signal S3. So, if the SSC event of any faulty signal is not found in the initial fault candidate set, then FBMTP has to backtrack along the DSVTF model transition path; and has to apply rule 3 (the formula to identify the initial fault candidate set in case of a fault with a faulty transition) on each state in order to get the new initial fault candidate set (the $<e_{ab}>$ component of rule 3 is considered to be an empty set). This backtracking procedure is repeated until the SSC event/s of the faulty signal/s is found in the initial fault candidate set (in other words, repeated until the exact state where the fault has actually taken place, is discovered—see Fig. 15).

As argued previously, an occurrence of an undetectable fault particularly which later transforms into a detectable fault, is extremely unlikely. In addition, even if such incident occurs, FBMTP has to explore a very few states during the backtracking procedure. This is because, first, it is nearly impossible that a long query transition sequence exactly similar to the corresponding DSVTF model transition sequence gets generated even after the occurrence of a fault (especially, a hard fault). Second, even if a PLC-controlled manufacturing system is physically large in size, operationally it is divided into several small partitions and a fault cannot propagate beyond its corresponding operational-level partition. So, the propagation of an undetected fault has limited impact on the performance of the overall fault isolation procedure of FBMTP. Until now, we constantly assumed that the data logger is able to capture all the faulty SSC events correctly. Unfortunately, this is not always the case. In some complex fault cases (although quite unlikely), a sensor failure causes its corresponding signal to be wrongly turned ON/OFF for less than TIB time. In such a case, the faulty SSC elements can be missing from the SSC element set of the (faulty) query transition because, the data logger may fail to detect such fast status transformation events of the faulty signal/s (recall the notion of the critical signal miss event from Section IV-B2). If such faulty SSC event miss incidents occur, then FBMTP may produce a fault candidate set that does not contain any SSC event of the faulty (critical) signal/s. Please note that this kind of situations cannot happen due to an actuator failure because, the status value of an actuator signal is determined based on the status values of the input sensor signals. Unfortunately, there is no algorithmic way to identify such faulty SSC event miss incidents-based only on the available log data records (as argued previously). From our practical experience, we have observed that a normal (sensor) signal turns into a faulty critical signal mainly because of the loose cable connection and when such failure occurs, the corresponding signal changes its status value too frequently. For this reason, even after the detection of a fault, FBMTP does not immediately halt its execution. It continuously inspects whether any signal is changing its status value repeatedly (there is a high probability that the data logger will able to detect some of the SSC events of the faulty critical signal/s). If any such signal is found, then it is also reported (along with the fault candidate sets) to the file. If the faulty critical signal does not change its status value frequently, then FBMTP may fail to localize the actual faulty signal (depending on whether the data logger is able to capture the faulty SSC events or not). However, generally, if a sensor breakdown occurs, the corresponding signal goes to ON/OFF state permanently (or for a very long period of time) and hence, all the erroneous SSC events are captured accurately by the data logger.

We have tested FBMTP on various real-world as well as simulated manufacturing systems (ranging from small to large systems) to find out its practical effectiveness. The results show significant improvement in the accuracy of the FBADI method compared to the previous works. We chose not to report those results here since the advantages of FBMTP are evident from the discussions throughout this paper (and also to economize space). For the detailed experimental results and further analyses, the interested readers are kindly asked to visit the project website http://www.udmtek.com/ksub05_03/articles/view/tableid/ksub05_03/id/40.

VI. Conclusion

In this paper, we have presented an automated tool called FBMTP that can detect and isolate the faults and behavioral anomalies associated with PLC-controlled manufacturing systems effectively. The main strength of this tool is that it can handle the log data inaccuracy issues associated with large manufacturing systems efficiently. In FBMTP, the information corresponding to the transition time, system alarm, transition execution pattern, undetected fault propagation, infinite looping at a state, etc., have been taken into consideration; and sufficient actions have been implemented in order to handle the log data inaccuracy issues accordingly. These measures obviously make FBMTP much more accurate and effective compared to the other existing methods. Our experimental results show that, FBMTP can accurately identify and isolate most of the faults and behavioral anomalies present in the manufacturing system; and provides better performance results than the previous works. In addition, FBMTP largely scales down the possibility of the state or space explosion. In some extreme situations, a fault or a behavioral anomaly that does not have much impact on the system operations can remain unidentified; and FBMTP may also produce an incorrect fault candidate set. However, the probability of occurrence of such incidents is extremely low in practice and hence, can easily be neglected. Future work planned will be to

1) investigate models with predictive and preventive fault finding features (using motor current signature analysis);
2) incorporate more information from various other sources (such as, PLC program, circuit diagrams, etc.) into the DSVTF model.

Acknowledgment

The authors would like to thank UDMTEK Company, Ltd. for the use of its research facilities during this paper.

References

[1] F. Basile, P. Chiacchio, and D. Gerbasio, "On the implementation of industrial automation systems based on PLC," *IEEE Trans. Autom. Sci. Eng.*, vol. 10, no. 4, pp. 990–1003, Oct. 2013.

[2] W. Hu, A. G. Starr, and A. Y. T. Leung, "Two diagnostic models for PLC controlled flexible manufacturing systems," *Int. J. Mach. Tools Manuf.*, vol. 39, no. 12, pp. 1979–1991, Dec. 1999.

[3] S. Qin and G. Wang, "A study of fault detection and diagnosis for PLC controlled manufacturing system," in *Proc. Int. Conf. Syst. Simulat. Sci. Comput. (ICSC)*, Shanghai, China, Oct. 2012, pp. 373–382.

[4] W. Hu, M. Schroeder, and A. G. Starr, "A knowledge-based real-time diagnostic system for PLC controlled manufacturing systems," in *Proc. IEEE Int. Conf. Syst. Man Cybern. (SMC)*, Tokyo, Japan, Oct. 1999, pp. 499–504.

[5] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. New York, NY, USA: Springer, 2008.

[6] M. Sayed-Mouchaweh, A. Philippot, and V. Carre-Menetrier, "Decentralized diagnosis based on Boolean discrete event models: Application on manufacturing systems," *Int. J. Prod. Res.*, vol. 46, no. 19, pp. 5469–5490, Oct. 2008.

[7] W. Qiu and R. Kumar, "Decentralized failure diagnosis of discrete event systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 36, no. 2, pp. 384–395, Mar. 2006.

[8] H. Chakib and A. Khoumsi, "Multi-decision diagnosis: Decentralized architectures cooperating for diagnosing the presence of faults in discrete event systems," *Discrete Event Dyn. Syst.*, vol. 22, no. 3, pp. 333–380, Sep. 2012.

[9] R. Debouk, S. Lafortune, and D. Teneketzis, "Coordinated decentralized protocols for failure diagnosis of discrete event systems," *Discrete Event Dyn. Syst. Theory Appl.*, vol. 10, no. 1, pp. 33–86, Jan. 2000.

[10] A. Philippot, M. Sayed-Mouchaweh, V. Carre-Menetrier, and B. Riera, "Decentralized approach to diagnose manufacturing systems," in *Proc. IMACS Mulitconf. Comput. Eng. Syst. Appl. (CESA)*, Beijing, China, Oct. 2006, pp. 912–918.

[11] S. Inagaki, T. Suzuki, M. Saito, and T. Aoki, "Local/global fault diagnosis of event-driven controlled systems based on probabilistic inference," in *Proc. 46th IEEE Conf. Decis. Control (CDC)*, New Orleans, LA, USA, Dec. 2007, pp. 2633–2638.

[12] Y. Pencolé and M.-O. Cordier, "A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks," *Artif. Intell.*, vol. 164, nos. 1–2, pp. 121–170, May 2005.

[13] A. Philippot, P. Marangé, V. Carré-Ménétrier, and B. Riera, "Implementation of diagnosis approach for discrete event systems," presented at the *Int. Symp. Security Safety Complex Syst. (2SCS)*, Agadir, Morocco, May 2012, pp. 1–6.

[14] R. Kumar and S. Takai, "Inference-based ambiguity management in decentralized decision-making: Decentralized diagnosis of discrete-event systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 6, no. 3, pp. 479–491, Jul. 2009.

[15] S. Genc and S. Lafortune, "Distributed diagnosis of place-bordered Petri nets," *IEEE Trans. Autom. Sci. Eng.*, vol. 4, no. 2, pp. 206–219, Apr. 2007.

[16] W. Hu, A. G. Starr, and A. Y. T. Leung, "Operational fault diagnosis of manufacturing systems," *J. Mat. Process. Technol.*, vol. 133, nos. 1–2, pp. 108–117, Feb. 2003.

[17] J. Fan, Y. Xie, and M. Ding, "Research on embedded PLC control system fault diagnosis: A novel approach," in *Proc. Int. Conf. Intell. Syst. Res. Mechatronics Eng. (ISRME)*, Zhengzhou, China, Apr. 2015, pp. 1876–1879.

[18] T. Alenljung, M. Skoldstam, B. Lennartson, and K. Akesson, "PLC-based implementation of process observation and fault detection for discrete event systems," in *Proc. IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, Scottsdale, AZ, USA, Sep. 2007, pp. 207–212.

[19] S. Klein, L. Litz, and J.-J. Lesage, “Fault detection of discrete event systems using an identification approach,” in *Proc. 16th Triennial World Congr. Int. Federation Autom. Control (IFAC)*, Prague, Czech Republic, Jul. 2005, pp. 92–97.

[20] M. Roth, J.-J. Lesage, and L. Litz, “An FDI method for manufacturing systems based on an identified model,” in *Proc. 13th IFAC Symp. Inf. Control Problems Manuf. (INCOM)*, Moscow, Russia, Jun. 2009, pp. 1406–1411.

[21] M. Roth, J.-J. Lesage, and L. Litz, “A residual inspired approach for fault localization in DES,” in *Proc. 2nd IFAC Workshop Depend. Control Discrete Syst. (DCDS)*, Bari, Italy, Jun. 2009, pp. 305–310.

[22] M. Roth, J.-J. Lesage, and L. Litz, “The concept of residuals for fault localization in discrete event systems,” *Control Eng. Pract.*, vol. 19, no. 9, pp. 978–988, Sep. 2011.

[23] M. Roth, S. Schneider, J.-J. Lesage, and L. Litz, “Fault detection and isolation in manufacturing systems with an identified discrete event model,” *Int. J. Syst. Sci.*, vol. 43, no. 10, pp. 1826–1841, Oct. 2012.

[24] S. Schneider, L. Litz, and M. Danancher, “Timed residuals for fault detection and isolation in discrete event systems,” in *Proc. 3rd Int. Workshop Depend. Control Discrete Syst. (DCDS)*, Saarbrücken, Germany, Jun. 2011, pp. 35–40.

[25] J. Bao, H. Wu, and Y. Yan, “A fault diagnosis system-PLC design for system reliability improvement,” *Int. J. Adv. Manuf. Technol.*, vol. 75, no. 1, pp. 523–534, Oct. 2014.

[26] J.-S. Lee and C.-C. Chuang, “Development of a Petri net-based fault diagnostic system for industrial processes,” in *Proc. 35th Annu. Conf. IEEE Ind. Electron. (IECON)*, Porto, Portugal, Nov. 2009, pp. 4347–4352.

[27] R. Alur and D. L. Dill, “A theory of timed automata,” *Theor. Comput. Sci.*, vol. 126, no. 2, pp. 183–235, Apr. 1994.

[28] M. G. Merayo, M. Núñez, and I. Rodríguez, “Implementation relations for stochastic finite state machines,” in *Proc. 3rd Eur. Perform. Eng. Workshop (EPEW)*, Budapest, Hungary, Jun. 2006, pp. 123–137.

[29] O. Kondratyeva and M. Gromov, “To the parallel composition of timed finite state machines,” in *Proc. 5th Spring/Summer Young Res. Colloquium Softw. Eng. (SYRCoSE)*, Yekaterinburg, Russia, May 2011, pp. 94–99.

[30] K. El-Fakih, N. Yevtushenko, and A. Simao, “A practical approach for testing timed deterministic finite state machines with single clock,” *Sci. Comput. Program.*, vol. 80, pp. 343–355, Feb. 2014.

[31] M. M. Abdelhameed and F. A. Tolbah, “A recurrent neural network-based sequential controller for manufacturing automated systems,” *Mechatronics*, vol. 12, no. 4, pp. 617–633, May 2002.

[32] D. Thapa, S. C. Park, C. M. Park, and G.-N. Wang, “Controller logic design and implementation using t-MPSG,” in *Proc. Int. Conf. Control Autom. Syst. (ICCAS)*, Seoul, South Korea, Oct. 2007, pp. 1540–1544.

[33] S. Wintner, “Formal language theory,” in *The Handbook of Computational Linguistics and Natural Language Processing*, A. Clark, C. Fox, and S. Lappin, Eds. Chichester, U.K.: Wiley, 2010, pp. 11–42.

[34] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 2nd ed. Reading, MA, USA: Addison-Wesley, 2001.

[35] H. Wang and M. Song, “Ckmeans.1d.dp: Optimal *k*-means clustering in one dimension by dynamic programming,” *R J.*, vol. 3, no. 2, pp. 29–33, Dec. 2011.

Arup Ghosh was born in Barasat, India, in 1989. He received the bachelor’s degree in computer science and engineering from the West Bengal University of Technology, Kolkata, India, in 2011, and the master’s degree in computer science from the University of Trento, Trento, Italy, in 2012. He is currently pursuing the Ph.D. degree in industrial engineering with Ajou University, Suwon, South Korea.

His research interests include manufacturing process modeling, motor current signature analysis, fault detection and isolation in industrial processes, PLC log data analysis, intelligent process control, and virtual manufacturing.

Mr. Ghosh was a recipient of the Erasmus Mundus External Cooperation Window Scholarship in 2010, and many grants from different ministries of the Government of the Republic of Korea.

Shiming Qin was born in Heze, China, in 1986. He received the master’s degree in industrial engineering from Ajou University, Suwon, South Korea, in 2013, where he is currently pursuing the Ph.D. degree in industrial engineering.

He is also a Senior Researcher with UDMTEK Company, Ltd., Gyeonggi, South Korea. His current research interests include automatic manufacturing system optimization, PLC control system design, modeling and simulation, and fault detection and diagnosis.

Jooyeoun Lee received the M.S. degree in management information system from the Ajou University, Suwon, South Korea in 1993, and the Ph.D. degree in management information system from the Inha University, Incheon, South Korea in 2004.

He was with Green Business Division, POSCO ICT, Seongnam, Gyeonggi, South Korea as an Executive Vice President, and Strategic Marketing Division, SK C&C, Seongnam, Gyeonggi, South Korea as a Vice President. Since 2014, he has been an Associate Professor in the Department of Industrial Engineering, Ajou University. He is also an Ombudsman for Industrial Convergence, Ministry of Trade, Industry and Energy, South Korea. His current research interests include business intelligence, convergence technology, energy management system, smart manufacturing, and cyber-physical systems.

Gi-Nam Wang received the M.S. degree in industrial engineering from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 1984, and the Ph.D. degree in industrial and system engineering from the Texas A&M University, College Station, TX, USA, in 1993.

Since 1993, he has been a Faculty Member with Ajou University, Suwon, South Korea, where he teaches courses on advanced information system design and practice, neural networks, and intelligent manufacturing systems. He is currently a Full Professor with the Department of Industrial Engineering, Ajou University. He has authored 16 patents, and has published over 50 original research papers in international journals, book chapters, and international conference proceedings. His current research interests include digital manufacturing, cyber-physical systems, virtual manufacturing, intelligent process control, PLC control system design, and modeling and simulation.