
Shih-Ting Fong, Chun-Jung Wang, Chuei-Tin Chang

Abstract—Although diagnostic tests have already been developed in the past for differentiating the originally inseparable fault origins in several simple batch processes, their applicability in realistic systems is still questionable. To address this practical issue, the dynamic behavior of every processing unit involved in a given sequential operation is modeled here by integrating both the generic engineering knowledge and also the ASPEN-generated simulation data into a single automaton. The improved test plans can then be synthesized according to the system model obtained by assembling all such automata. The feasibility of this model building strategy is demonstrated with an example concerning the start-up operation of a flash process.

Keywords—Automata; Diagnostic tests; Dynamic simulation; Sequential operations

I. INTRODUCTION

Unexpected faults and failures in a chemical plant often result in catastrophic consequences. Since the offline hazard assessment methods can limit the total expected loss of accidents only to a certain extent, online fault diagnosis is an alternative means for further improving operational safety. It has been shown by Kang and Chang [1] that one way to improve the diagnostic performance of an existing system is to implement test procedures. Although there have been a few published studies discussing various related issues [2 – 4], each of them dealt with only simple processes. In the present work, a unified automata-based strategy has been developed to synthesize credible operating procedures needed for diagnostic tests [7]. Specifically, the extended finite automata (EFA) have been adopted to facilitate model building and procedure synthesis with the free software SUPREonica [5]. An extensive set of new automaton configuration techniques have been proposed to incorporate both the generic engineering knowledge and also the simulation data generated by ASPEN Dynamics. The feasibility and effectiveness of this hybrid modeling approach are demonstrated with an example concerning the start-up operation of a realistic flash drum.

II. MODEL BUILDING METHODS

A. Hierarchical process structure

All components in a batch process can be classified into a hierarchy of four levels, i.e. the programmable logic controller (PLC), the actuators, the processing units and the online sensors. To illustrate this system framework, let us consider the flash operation described in Figure 1 (process flow diagram) and Figure 2 (normal operating procedure) as an example. In this system, there are a heater, an inlet valve (Vin), two outlet valves (Vvap and Vliq) and four PID controllers (INPUT_FC, HEATER_TC, FLASH_PC and FLASH_LC) for controlling the feed flow rate and temperature, and the vapor pressure and liquid level in the flash drum, respectively. It is also assumed that the raw material is a mixture of 20% H₂O, 60% methanol and 20% N₂, and its temperature is 25°C. It can be observed that, other than PLC in the first level, these units can be classified into the remaining three levels, i.e., (2) the heater, valves and PID controllers, (3) the flash drum, and (4) the flow, temperature, level and pressure sensors.

To facilitate unambiguous discussions, a Nomenclature section is provided at the end of this paper and the initial conditions of all processing units are chosen as follows:
1. Vin, Vvap and Vliq are all closed (A_Vin=0 & A_Vvap=0 & A_Vliq=0); heater is off (A_H=0);
2. All controllers are on manual;
3. Flash drum is empty (PU_Flevel=0) and its temperature is low (PU_Ftemp=0).

In addition, it is assumed that the level and temperature sensors on flash drum can be used respectively for detecting the following states of flash process:
1. “Normal” means the liquid level is kept at 2.5 m and temperature is at 75 °C;
2. “C_Idle” means the flash drum is empty;
3. “C_LL_TH” means the liquid level is lower than 2.5 m and temperature is kept at 75 °C;
4. “C_LL_TL” means the liquid level is lower than 2.5 m and temperature is kept at 75 °C and temperature is low (e.g., at the room temperature);
5. “C_LH_TH” means the liquid level is higher than 2.5 m and temperature is at 75 °C;
6. “C_LH_TL” means the liquid level is higher than 2.5 m and temperature is low (e.g., at the room temperature).

B. Diagnoser synthesis procedure

A “diagnoser” can be synthesized according to the following four steps:
1. Build the unit models based on the process flow diagram;
2. Build the PLC model based on the sequential function chart (SFC).
3. Build the layer model;
4. Assemble the system model and then synthesize its diagnoser accordingly.

For simplicity, let us consider only five failure events:
- F1 (VinSC): Vin sticks at close position;
- F2 (VinSO): Vin sticks at open position;
- F3 (VliqSC): Vliq sticks at close position;
- F4 (heater fails): heater fails;
- F5 (flash leaks): flash drum leaks.

It should be noted that all variables in EFA should be adopted to characterize the component states: (see Nomenclature for details). These variables can be summarized as follows:
1. Controller set points: s_temp, s_level, s_flow, s_press.
2. Valve and heater states: A_Vin, A_Vliq, A_Vvap, A_H.
3. Process states: PU_Ftemp, PU_Flevel, PU_Fin, PU_Fliq, PU_Fvap. These process states can be further divided into two types:
   - The transient values of instantaneous variables can be affected within a short period of time in response to the changes in actuator states. For example, the inlet flow rate can be adjusted by manipulating the opening of inlet valve.
   - The direction of change in each accumulated variables may be dependent upon one or more instantaneous variable. For example, the increase of liquid level in a tank can be confirmed if the difference between input and output flow rates is positive.

Each event in EFA can be activated only when its guarding condition(s) is satisfied and, after completion of this event, one or more designated state should then be updated. Note that such a modeling approach greatly reduces the total number of places in automata.

**B.1 Unit models**

Let us first consider valve Vin. As shown in Figure 3, there is a normal valve state, i.e., “Vin normal”, representing the valve is at either its normally-closed or normally-open position. The self-looping event “Vin_change” denotes that Vin is under automatic control. On the other hand, the event “Vin_p” represents the action of opening Vin more under manual control and its guard (PU_Flevel!=2) requires that the drum must not be full. The event “Vin_n” represents the action of reducing the valve opening of Vin under manual control, and this action can only be carried out when its guard (PU_Flevel!=0) (i.e., the drum is not empty) is satisfied. The failure “fVinSC” is equipped with guards (A_Vin==0 & s_flow>0) to impose the requirement that “fVinSC” only occurred when valve Vin is closed and flow set-point s_flow is positive, while The failure “fVinSO” is equipped with guards (A_Vin==4 & s_flow!=4) to impose the constraint that “fVinSO” can only take place when Vin is open manually and s_flow is irrelevant.
As a second example, let us consider the automata representing the accumulated variables (i.e., temperature and liquid level) of the flash process (see Figure 4). In this case, the corresponding models are formulated according to the general laws of mass and energy balances as follows:

- \((\text{PU}_\text{Flevel}=+1)\) if \(\text{PU}_\text{Fin} > \text{PU}_\text{Fliq} + \text{PU}_\text{Fvap}\)
- \((\text{PU}_\text{Flevel}=-1)\) if \(\text{PU}_\text{Fin} < \text{PU}_\text{Fliq} + \text{PU}_\text{Fvap}\)
- \((\text{PU}_\text{Ftemp}=+1)\) if \(\text{A}_\text{H} > \text{PU}_\text{Ftemp}\)
- \((\text{PU}_\text{Ftemp}=-1)\) if \(\text{A}_\text{H} < \text{PU}_\text{Ftemp}\)

Figure 4. Processing unit model of flash drum.

### B.2 PLC and Layer models

The automaton model of the normal operating procedure is shown in Figure 5. The operation steps are expressed as events in this automaton, while the activation conditions (i.e., see the horizontal bars in Figure 2) are their guards and the actuator actions (i.e., see the rectangles in Figure 2) are represented with the updated variables. Notice that the precedence order of these actions and conditions in this PLC model are exactly the same as that in SFC. On the other hand, a layer model is utilized in this work to ensure all events in the flash process occur in correct sequence (see Figure 6). Firstly, it is assumed that the initial state should be “C_Idle” at layer0. The changes in actuator states, set points or the equipment failures at layer1 may follow which, in turn, cause the changes in the instantaneous variables at layer2. Next, the accumulated variables (i.e., liquid level and temperature) at layer3 could be affected by the variations in instantaneous variables, and subsequently a particular allowed state of the flash drum could then be reached. Finally, since sensor failures are not considered in the present example, the sensor readings are assumed to be the same as the corresponding process states.

Figure 5. Normal controller model of flash drum.

Figure 6. Layer model.

### B.3 Simulation-based modeling approach

Since the conventional knowledge-based modeling approach may not always be feasible in realistic applications, the simulation-based models have been developed in this work to capture the dynamic system features with commercial software. Four steps are needed to generate such models:

1. Generate the state-transition data base. The simulation data are first exported from ASPEN Dynamics, and the variation ranges of all variables are estimated and then discretized accordingly. The output of a state transition should be associated with an accumulated variable at a future time, while the inputs are the current instantaneous variables.

2. Convert the state-transition data base to automata in SUPREMIC [5]. Let us consider the simulation-based model of the flash drum as example. Each transition in this model should be represented as a self-looping event “db_1”, “db_2”, “db_3” or “db_4”. Its inputs are translated as the guards and the output is signified with the updated variable. In the flash example, the inputs are values of process states and actuator states. The output \(\text{db}_\text{Flevel}\) is the difference between the future and present values of \(\text{PU}_\text{Flevel}\), while \(\text{db}_\text{Ftemp}\) is the difference between the corresponding values of \(\text{PU}_\text{Ftemp}\).
3. Add the transitional events “Flevel_db” and “Ftemp_db” created in previous step to layer3 in the layer model.

4. Update the automata for characterizing the accumulated variables (see Figure 8). If the differences db_Ftemp and db_Flevel is not zero, PU_Ftemp and PU_Flevel should be updated accordingly. On the other hand, it will change the state from data base state “m_Ftemp_db” & “m_Flevel_db” to the general state “m_Ftemp_general” & “m_Flevel_general” when equipment failure occurred. The latter model can be generated according to the general laws of mass and energy balances. Thus, the modified model can be created as follows:

(a) Create two states “m_Ftemp_db” and “m_Flevel_db” and each is attached with a self-looping event that updates the accumulated variable PU_Ftemp or PU_Flevel when one or both of the two differences db_Ftemp and db_Flevel is nonzero.

(b) Create another two states with self-looping events that update the accumulated variables according to general engineering knowledge. The states in (a) are then directed to the corresponding states in (b), and both transitions are treated as the same event of “equipment_failing”.

B.4 Diagnoser

The standard operation of parallel composition can be applied in SUPREMICA to integrate all automata mentioned above to synthesize a diagnoser [6]. The resulting seven observable event traces (OETs) in the flash start-up operation can be found in Figure 9. Note that there are three OETs that are not diagnosable, i.e., Tr_{01}-Tr_{03}. Note that multiple fault origins can be implied after observing any of these OETs, and there may be one or more failure in each fault origin. Although the probabilities of these scenarios may be quite different, they can be validated qualitatively with ASPEN Dynamics. It is a brief summary of these scenarios which is given below.

- Tr_{01}: The four possible fault origins in this case are: F1, F1F4, F1F5, and F1F4F5. The corresponding sensor readings after S_1 indicate that the system state is “C_Idle”, because F1 (VinSC) is present in all four scenarios.

- Tr_{02}: The three possible fault origins are: F2, F3, and F2F3, and the sensor readings show “C_LH_TH” after S_2.

- Tr_{03}: The four possible fault origins should be: F4, F2F4, F3F4 and F2F3F4, and the eventual sensor readings show “C_LH_TL” because heater fails after S_2.
III. DIAGNOSTIC TESTS

If an OET of the diagnoser is undiagnosable, a test plan may be devised to further improve the diagnostic resolution by implementing extra actuator actions.

A. Test target

In order to enhance the diagnosability of an undiagnosable OET, an auxiliary automaton A (see Figure 10) should be built to force the search for a test plan that can result in distinct system states for different fault origins. Let us take Tr03 as an example. Because F4 (heater fails) is confirmed in this case, this event should be associated with the transition from S0 to S1 in automaton A. Since the other two failures, i.e., F2 (VinSC) and F3 (VinSO), may or may not be present, they are represented with the self-looping events at S3. Since a set of actuator actions should be manually performed to change process states after failure(s), the corresponding self-looping events must be placed on S2. Finally, based on the assumption that a fault origin may be identified if a unique new system state can be produced with the diagnostic tests, each self-looping event on S1 event is equipped with a guard which represents a unique combination of possible sensor readings except that of the initial conditions. In this case, the initial state is “C LL_TH” (PU_Ftemp!=2 & PU_Flevel!=3). Therefore, if a new state is to be found, the event “diagnostic test” must satisfy the guards (PU_Ftemp!=2 | PU_Flevel!=3).

B. Test plans

After marking all terminal places in automaton A, the test plan of an undiagnosable OET (say Tr1) can be synthesized as follows:

1. Remove the normal PLC model.
2. Set the initial state of every component on the basis of the terminal condition of Tr1. Specifically, there are three possible scenarios to be considered:
   a. If a particular component failure can be confirmed after observing Tr1, i.e., it is present in all fault origins, set the corresponding failed state as the initial condition of this component.
   b. If all failures of a specific component can be excluded by Tr1, i.e., they do not appear in any of the fault origins, the initial condition of this component should be its final state reached in Tr1.
   c. If the presence of a particular component failure is uncertain, i.e., it appears only in some (but not all) fault origins of Tr1, the initial condition of this component should be set at the normal state prior to the failure event.
3. Modify the component models by resetting the initial state of each component according to step (2) and remove the failures and failed states that are not implied by Tr1.
4. Create the auxiliary automaton A.
5. Perform the “Synchronize” function in SUPREMICA for all aforementioned models and search for test paths leading to separate marked states.

C. Examples

Tr01: No feasible test plans can be generated since failure F1 is bound to occur in this scenario. From the facts that the flash drum is empty after Tr02 is fully developed and cannot be refilled again by opening Vin, one can conclude that the liquid level and temperature cannot be further altered and, thus, there are no ways to differentiate the implied four fault origins with diagnostic tests.

Tr02: The fault origins of this trace are F2 (VinSO) and F3 (VliqSC), F2F3 (VinSO and VliqSC). The test plan can be obtained with the following procedure:

1. Remove the original PLC model.
2. Since the states of Vin and Vliq are both uncertain, the initial condition of each valve should be set at the state prior to its failure. Thus, the initial state of the valve Vin is at the half-open position (A_Vin=4), and the initial state of the valve Vliq is at the open position (A_Vliq=0).
3. Let us use valve Vin as an example (see Figure 11) to illustrate the model modification step. After S2, it is uncertain whether the valve is at the half-open position (A_Vin=2) or at a failed state due to F2 (VinSO). Therefore, the initial state should be set at the fully-open position (A_Vin=4) so that both possibilities can be covered. On the other hand, the failure events “f_VinSC” and auto control event should all be removed because these are not used in this situation. Note that the model of Vliq should also be modified with the same approach. Finally, the processing unit models can be simplified by removing the simulation-based state transitions (Figure 12).

4. An auxiliary automaton (see Figure 13) should be built according to the approach described previously.
5. All models are synchronized and the corresponding observable event trace can be obtained (see Figure 14). Note that, after the extra step “A_Vin n” is executed, trace Tr02 splits into 2 groups: “C LL_TH” and “C LL_TH (no change)”.

Figure 10. Structure of auxiliary automaton A for Tr03

Figure 11. Modified actuator models (Vin) for Tr02
IV. CONCLUSIONS

A hybrid modeling strategy has been developed in this work for synthesis of diagnostic test plans for sequential operations. These methods can be used to:

- to construct unit, PLC, layer and system models based on engineering knowledge and simulation results,
- to produce the corresponding diagnoser, and
- to synthesize the test plans for all undiagnosable traces to further enhance resolution.

REFERENCES


NOMENCLATURE

\( \text{Tr}_{03} \): As mentioned before, 4 fault origins are implied by this trace, i.e., F4 (heater fails), F2F4 (VinSO and heater fails), F3F4 (VliqSC and heater fails) and F2F3F4 (VinSO, VliqSC and heater fails). After synchronizing the required automata, two event traces can be found and a sequential functional chart can then be obtained (see Figure 16).

\( s_{\text{temp}} \): Set point of temperature controller, -1 (Initial value) / 0 (Off) / 1 (-) / 2 (75 °C)

\( s_{\text{level}} \): Set point of level controller, -1 (Initial value) / 0 (Close) / 1 (2.5m) / 2 (Open)

\( s_{\text{flow}} \): Set point of flow controller, -1 (Initial value) / 0 (Close) / 1 (-) / 2 (26000 kg/hr) / 3 (-) / 4 (Open)

\( s_{\text{press}} \): Set point of pressure controller, -1 (Initial value) / 0 (Close) / 1 (Open)

\( A_{\text{Vin}} \): Opening of inlet, 0-4: 0 (Close) / 1 (-) / 2 (26000 kg/hr) / 3 (-) / 4 (full Open)

\( A_{\text{Vliq}} \): Opening of bottom outlet, 0 (Close) / 1 (13956 kg/hr) / 2 (full Open)

\( A_{\text{Vvap}} \): Opening of top outlet, 0 (Close) / 1 (Open)

\( A_{\text{H}} \): Energy output of heater (exchanger), 0 (Off) / 1 (-) / 2 (75 °C)

\( \text{PU}_{\text{Ftemp}} \): Temperature in the flash drum, 0 (0 °C) / 1 (25 °C) / 2 (75 °C)

\( \text{PU}_{\text{Flevel}} \): Liquid level in the flash drum, 0 (0 m) / 1 (<2.5 m) / 2 (2.5 m) / 3 (>2.5 m)

\( \text{PU}_{\text{Fin}} \): Inlet flow rate, 0 (0) / 1 (-) / 2 (26000) / 3 (-) / 4 (49000)

\( \text{PU}_{\text{Fvap}} \): Top (vapor) outlet flow rate, 0 (0 kg/hr) / 1 (7000 kg/hr)

\( f_{\text{leak}} \): Flash drum leaks, 0 (No leaking) / 1 (leak)