A structural decomposition-based diagnosis method for dynamic process systems using HAZID information

A. Tóth

Department of Electrical Engineering and Information Systems, University of Pannonia, Veszprém, Hungary

Á. Werner-Stark

Department of Electrical Engineering and Information Systems, University of Pannonia, Veszprém, Hungary

K.M. Hangos

Institute for Computer Science and Control, Budapest, Hungary

Department of Electrical Engineering and Information Systems, University of Pannonia, Veszprém, Hungary

Abstract

A novel diagnosis method is proposed in this paper that uses the results of the blended HAZID analysis extended to the dynamic case of process systems controlled by operational procedures. The algorithm is capable of finding fault root causes in process systems using nominal and observed possible faulty operational procedure execution traces. The algorithm uses the structural decomposition of the process system and its component-level dynamic HAZID (P-HAZID) tables and executes the diagnosis component-wise by first decomposing the observed execution traces, and then assembling the diagnosis results. The exact structure of the algorithm is also discussed, followed by two case studies on which its operation is demonstrated.

Email addresses: atezs82@gmail.com (A. Tóth), werner.agnes@virt.uni-pannon.hu (Á. Werner-Stark), hangos@scl.sztaki.hu (K.M. Hangos)

1. Introduction

Loss prevention in process systems is a task of crucial importance in avoiding fatal accidents and plant breakdowns. An important cornerstone of preventing losses is the early mitigation and isolation of faults which can be achieved by applying various fault detection procedures to the system. Because of its vital importance, the literature of various diagnostic approaches is enormously wide, but the model-based methods for fault detection and isolation are the most widespread.

Depending on the a priori and measured information that is available for diagnosis, the fault diagnostic approaches for process systems are categorized into quantitative, qualitative and process history based approaches in [15]. Model-based qualitative diagnostic approaches, to which our method also belongs, apply approximate "qualitative" information for both the models and the measured signatures or symptoms.

Hazard identification has been long taken as an independent activity from diagnosis, but the information they built on has a lot of common elements. The HAZOP (Hazard and Operability, see [4]) and FMEA (Failure Effect and Mode Analysis, see [5]) are two fundamentally different analysis methods for hazard identification, where HAZOP is deviation-driven and FMEA is process component-driven. Due to the complexity of real process systems, the time-consuming and error-prone manual construction of HAZOP and FMEA tables has been identified as a major bottleneck in hazard identification of process systems. Fortunately, there have been results for automated generation of them (for HAZOP, in [15] together with a concrete application in [14]). Another possible way is to use qualitative models in such an analysis (again, for HAZOP, see [6] with a batch process system application in [10]). An attempt to unite the two different diagnostic information stored in HAZOP and FMEA analysis results, called the blended HAZID methodology was described in [9] together with its use for process system diagnosis tasks.

The domain for the usual HAZID analysis is static in the sense, that devi-

ations from the normal, usually steady-state operation are recorded and used. It is, however, possible to extend the blended HAZID diagnostic idea to the dynamic case, when the execution of an operational procedure drives the process system from one state to another, and use it for diagnostic purposes. This method can be used for finding component faults based on the deviation(s) between the observable inputs and outputs of planned (nominal) and actual (characteristic) event sequences driven by an operational procedure. In order to achieve this, the method uses a novel dynamic procedure HAZID (P-HAZID) information as described in [16]. The use of these P-HAZID tables for diagnostic purposes had been formalized in [12], and it had been extended by taking the structural similarities of process system components into account in [13].

The aim of this paper is to describe a diagnostic approach based on the P-HAZID methodology which can be used effectively for process systems. By decomposing the process system into smaller and individually better diagnosable components, the inevitable combinatorial explosion associated with the amount of diagnostic information can be managed better. For this purpose, a similar object-oriented process system topology modeling approach was used as in [8], but here the decomposed diagnostic reasoning is developed and utilized.

In the first part of the paper, the basic notions of operational procedural diagnosis, and the single-component diagnostic procedure is described. Based on these fundamental concepts, the second part discusses the idea of structural decomposition in process systems based on P-HAZID information and how a research prototype for the idea was implemented. Finally, in the third part the operation of the algorithm is shown on two simple case studies.

2. The blended HAZID approach and its extension to event sequences

In this section the basic notions are developed for use later in the article during the description of the algorithm. These are the same notions used in [13].

2.1. Blended HAZID ([9])

The blended HAZID approach (BLHAZID, described thoroughly in [9]) combines the system-driven HAZOP and the component-driven FMEA into one methodology, attempting to minimize their weaknesses and utilize their strengths at the same time. Unlike the original HAZOP and FMEA, three main steps in the initial analysis are needed (based on [9]):

- Decompose the system into subsystems - the analysis is done in subsystem level onwards.
- Find deviations from intended functions, with their causes and implications.
- Elicit the causes and effects of each fault per component in every subsystem on the function of the system.

As a result of the analysis, a cause-implication directed graph (see [9] for an example) can be drawn for each identified failure to visualize casual relationships between failures and components. The nodes in this graph are either components or functional failures and each edge represents a causal relationship between them. This graph is a powerful visualization tool for plant operators. The table used for diagnosis (described in section 2.3) is analogous to this graph, however, it describes causal relationships between operational procedure deviations and failure root causes.

2.2. Diagnostics based on event sequences

Qualitative range spaces. Current values of real-valued scalar outputs in process systems can be described using a properly selected qualitative range space. For example, to describe the value of a level sensor in a tank, the following range space can be used:

$$Q_e = \{\mathbf{e}-, 0, L, N, H, \mathbf{e}+\} \tag{1}$$

Here 0 means an empty tank, L, N and H means low, normal and high fluid level respectively, while $\mathbf{e}-$ and $\mathbf{e}+$ refer to unmeasurably low and high fluid

levels (this might mean a failure in the level sensor itself). This range space will be used to describe system outputs during operation.

Similarly, binary-valued valve actuator states can be described using the following range space (an extended qualitative range space, such as Q_e, for example, can also be used for the input values in the general case):

$$Q_b = \{cl, op\} \tag{2}$$

Here cl means a closed state, while op means an open state.

Traces. Operational procedures in process systems can be seen as detailed lists of instructions for the plant operator personnel to perform certain operations on the plant. Procedures can be formally described using finite input-output event sequences where a single event describes a state of the inputs and outputs of the system at a specific time instant. Therefore the syntax of a an input-output event (at time instant t of an n-input m output system) is the following:

$$event_t = (t; input_1, ..., input_n; output_1, ..., output_m) \ .$$

The time instant of an event is always a natural number, defined as the sequence number of the discrete event. The inputs in an event always refers to a state of an actuator component in the process system (e.g. in the case of a valve it can be open (op) or closed (cl)). On the other hand, the outputs in an event refer to a value of an output of the process system in the qualitative range space using the qualitative set defined in (1). Examples of events in a system with a single binary-valued input and a single real-valued output are (1;cl;N) or (5;op;0).

Sequences formed from these events are called **traces** and defined as:

$$T(t_1, t_n) = event_{t_1}, ..., event_{t_n}$$

Separate events in the same trace always contain the same inputs and outputs.

Nominal and characteristic traces. For every operational procedure there exists a trace (called the **nominal trace**) which describes its behavior

under fault-free conditions. Comparing this to other traces which may have been executed under faulty conditions (called **characteristic traces**), the differences (called deviations) can be used to find possible malfunctions of components in the system.

Deviations. Nominal and characteristic traces can be compared by comparing their corresponding event fragments. The difference between two corresponding event fragments (where the time instant is the same) is described by a single deviation. Deviations are formed from a deviation guide word and the nominal event from which the corresponding characteristic trace event is deviating from. Two deviation types can be distinguished, a deviation might be **chronological** or **quantitative**. Chronological deviations describe that the event in the nominal trace did not occur at the correct time, while quantitative deviations denote a difference in output values between events of the same time instant. A quantitative deviation is always tied to an output variable in the event, while a chronological deviation always relates to the event itself.

The following types of chronological deviations are used:

- **never-happened**: When the particular event never happened in the characteristic trace.
- **later**: When the event happened in the characteristic trace, but at a later time instant.
- **earlier**: When the event happened in the characteristic trace, but at an earlier time instant.

The following types of quantitative deviations are used:

- **greater**: When a particular output's qualitative value (see Q_e in (1)) was higher in the characteristic event that of the nominal one.
- **smaller**: When a particular output's qualitative value was lower in the characteristic event.

For example, the chronological deviation **earlier**(3;op,cl;N) "means" that the third nominal event, where the inputs were "open" and "closed" and the output value was "normal" happened at an earlier time instant in the characteristic trace. As an other example, the deviation **smaller**(4;op,op;N) denotes that at the fourth characteristic event the value of the output was smaller than the nominal.

2.3. The P-HAZID table ([16])

As a combination and extension of the widely used FMEA and HAZOP analyses (for details, refer to [12] or [7] and partly [11]), the procedure HAZID (abbreviated as P-HAZID) analysis result can be used for fault diagnosis during operational procedures in a given process system. The result of this P-HAZID analysis is given in the form of a table, and it consists of deviations leading to possible root causes ordered in columns. Using the initial set of differences (deviations) between the characteristic trace and the nominal trace, the set of possible root causes can be found by simple reasoning taking the connections between rows into account. Two rows are said to be connected in the table if the corresponding values in the (Deviation,Implication) columns and (Cause,Deviation) columns are equal. This is used during reasoning to find possible root causes from a set of initial deviations by chaining rows after each other. Therefore, from an initial set of starting deviations a reasoning tree can be drawn connecting them to the possible root causes based on the P-HAZID table. The reasoning tree describes causal relationships between operational procedure deviations and root causes for a particular process system. A simple example of a P-HAZID table can be found in Table 1.

2.4. Simple diagnosis based on the P-HAZID table ([12], [13])

Using a nominal operational procedure with its corresponding P-HAZID table and a possibly faulty characteristics trace the diagnostic algorithm operates as follows:

Cause	Deviation	Implication
TANK-LEAK	NH(2;op,cl;L)	NH(3;op,cl;N)
NH(2;op,cl;L)	NH(3;op,cl;N)	NH(4;op,op;N)
TANK-LEAK	SML(2;op,cl;L)	SML(3;op,cl;N)
SML(2;op,cl;L)	SML(3;op,cl;N)	SML(4;op,op;N)
NEG-BIAS	SML(1;op,cl;0)	SML(2;op,cl,L)
SML(1;op,cl;0)	SML(2;op,cl,L)	SML(3;op,cl,N)
SML(2;op,cl;0)	SML(3;op,cl,L)	SML(4;op,op,N)
POS-BIAS	GRE(1;op,cl;0)	GRE(2;op,cl,L)
GRE(1;op,cl;0)	GRE(2;op,cl,L)	GRE(3;op,cl,N)
GRE(2;op,cl;0)	GRE(3;op,cl,L)	GRE(4;op,op,N)

Table 1: A fraction of a simple **P-HAZID** table.
Inputs: op=open, cl=closed. Outputs: 0=no, L=low, N=normal. Deviations: NH=**never-happened**, SML=**smaller** and GRE=**greater**. Root causes: **TANK-LEAK** is the leak of the tank and **NEG-BIAS**/**POS-BIAS** is the negative/positive bias failure of the tank level sensor.

1. All deviations are collected between the nominal and characteristic traces. The deviations at the final and preceding time instant are calculated, and assigned to the set of final deviation pairs by increasing order in the time instant. The next step is executed for each final deviation pair.
2. All the rows in the P-HAZID table are selected where exactly the first element of the pair is found in the Deviation column, and exactly the second element is found in the Implication column.
3. For all selected rows: if the selected row's Cause column contains a root cause, or a deviation which is not present in the original set of deviations (called terminal deviation), then this is added to the set of root causes and the reasoning finishes for this single path by continuing with step 5, otherwise it continues with step 4.
4. If the selected row's deviation is contained in the collected set of deviations then continue with step 2 by taking the deviation in the Cause column as the first element of the pair and the Deviation element as the second element of the pair (to find rows where the content in Cause equals to the Deviation

and the content is Deviation equals to implication).

5. Return the set of found root causes and the set of deviations which were not present at the initial set of deviations.

This diagnostic algorithm attempts to construct a set of reasoning trees in the P-HAZID table (rows are represented as nodes) using the method mentioned above from the set of initial deviations. The roots of the tree are the final deviation pairs from where the diagnostic procedure is initiated, while the leaves are the identified root causes or the deviations from which the diagnosis cannot proceed forward (because they were not contained in the set of initial deviations). The junctions (nodes with degree more than 2) denote rows in the P-HAZID table which are connected to multiple rows in the P-HAZID by their event values in the Deviation and Implication columns. If the diagnosis cannot proceed forward from a deviation because it was missing from the set of initial deviations, this might mean a failure in the P-HAZID table (thus the algorithm might be used for validating the P-HAZID table).

Such a reasoning tree can be found for the P-HAZID table of Table 1 and the set of deviations:

- **smaller**(2;op,cl;N)
- **smaller**(3;op,cl;N)
- **smaller**(4;op,cl;N)

In this case - due to the negative bias of the level sensor - the leak of the tank and the negative bias fault of the sensor cannot be distinguished by the algorithm, so both are added to the set of found root causes.

3. Diagnosis using structural decomposition

Using the already discussed method of simple diagnosis, root causes can in principle be found, but in the case of complex process systems creating an appropriate P-HAZID table might be a laborious and highly complex task. Because

most process systems are constructed from similar elements (e.g. tanks, pipes, valves), called components, the diagnostic task can also be decomposed accordingly, and then performed using component-level diagnosis. In this case only the component level P-HAZID tables and nominal trace need to be constructed, which is a more simple task compared to assembling the global P-HAZID table and nominal traces. This is the basis of the structural diagnostic procedure proposed in this paper.

3.1. Components

A component in the diagnostic system consists of a component-specific P-HAZID table, a nominal trace, a start condition and a transformation function. The P-HAZID table describes all root causes (or component failure modes) and deviations leading to them the same way as in the case of single component diagnosis. The nominal trace defines the failure-free operation of the component, and is always a subset of the system-level nominal trace which relates to the whole system. Events in the nominal trace and P-HAZID table only contains input and output values which are related to the particular component, making the creation of the table easier: the creators can focus on the local events and does not need to deal with all the inputs and outputs. During diagnosis, a component-related characteristic trace fragment is generated from the observed global characteristic trace, and local deviations are generated by comparing that characteristic trace fragment to this nominal trace.

The component start condition describes a global system-level condition on inputs an outputs under which the particular component can be reasonably diagnosed (so that - in a case of a tank, for example - diagnosis will not start if the preceding component was broken, and no liquid at all can flow to the tank).

The transformation function of the component maps system-level events to component-level (by trimming inputs and outputs not relevant to the particular component and reordering them so that the resulting events can be used in local P-HAZID diagnosis). In that way the components might get diagnosed in isolation and might use the same P-HAZID diagnostic information. The trans-

formation function of the component is unique for every component instance in the system.

All components are instances of specific equipment classes (eg. tank, pipe, valve), process systems are built from them in an object-oriented manner as in [10]. That means that similar parts (eg. tanks) in process systems can use the same P-HAZID information (tables) and nominal traces, in that way decreasing the redundancy in the diagnostic system.

The majority of the diagnostic information (equipment specific P-HAZID table and nominal trace part) belongs to the equipment class, so it can be reused among similar components in the system (with the same physical behavior and role in the system-level operation procedure). Component instances shall only contain their own input/output transformation function (to convert the system-level input and output names to the ones "locally" used in the equipment specific P-HAZID table and nominal trace) and start-condition (to stop the diagnostic procedure if a system is in a state which makes the component diagnosis obsolete - for example due to a severe fault in an other component).

3.2. Breaking the complete P-HAZID along components

Based on the system's structure and the events in the system-level P-HAZID table it is possible to break the system up into individual components connected by boundary elements (valves or pipes). Note that an operational procedure usually deals with one part of the system at a time, these parts can be identified from the nominal trace and can be converted into components with inputs (using the boundary elements of the parts) and measurable internal states or outputs.

3.3. Setting up the component graph

Based on the physical characteristic of the system and the system-level nominal trace, components can be identified in the system and the connection between them can be modeled by a non-directed graph (for example there is an edge in the graph between two tank components which are connected with a pipe physically). Using this graph, if we have a starting component (where

the operational procedure starts at) all possible execution paths (**component paths**) of the system can be determined. This determination also gives edge directions to the graph (as the operational procedure operates on the system), forming a directed graph. The reason for the graph based representation is that diagnosis only progresses until the last component with a true starting condition, but if there are multiple components connected in parallel, then the failure in one component should not affect the other component directly (provided they are on different paths) - so they shall be diagnosed separately. In such a component graph, parallelly connected components belong to different component paths, and all component paths are diagnosed until the last component with a true start condition.

For example, considering the process system depicted in Figure 2, and the operational procedure in Table 2 (which fills up all the three tanks with liquid and then opens up the output valves), the simple component graph is seen on the bottom part of the figure.

3.4. Distributing trace fragments among components

From the nominal trace the start component can always be determined. Knowing the component graph, the characteristic trace and the start component, the corresponding parts of the characteristic trace can be distributed among components. The resulting fragments only refer to a single component, and they can be compared with the component-level nominal trace to generate deviations and find possible faults (for the component). If there are multiple components in the system, then there are inputs which directly relate to neighboring components (like a valve on a pipe which is between two tanks, feeding one from the fluid in the other). These inputs are included in the characteristic traces of both components. Moreover, if there is a specific time instant in which this valve is manipulated (switched to open from closed state for instance) then this whole event need to be present in both characteristic trace fragments for the neighboring components (at the end of the first, and at the beginning of the next one). These events are not the same (due to the possible component-level

input/output mapping) but nevertheless they are created from a global event in which the valve was manipulated. Such events which relate to neighboring components are called **boundary events** and inputs manipulated in them (input between the components, like the valve in the previous example) are the boundary inputs.

The simple nominal trace in Table 2 shows the component level distribution of a system-level nominal trace, the result of which is shown in the component graph next to the nodes in Figure 2 as inscriptions. It is important to note that every component-level trace begins with the first time instant - unlike its source event in the system-level trace it was created from. Also, as mentioned before, the last event of component TA became the first event of the component TB, and all of the component-level events in the traces contain only component-local inputs and outputs (VA,VB,VC,TA for TA; VB,VE,TB for TB and VC,VD,TC for TC).

A serial connection between the components results in a simple component graph where components are chained after each other, therefore the trace need to be broken up - and different components are assigned different parts in the trace. On the other hand, in a parallel connection in the component graph (as in Figure 3) the same trace fragments are distributed among the parallelly connected components (in that case some of the components might get events from the same system-level event but with different inputs and outputs).

3.5. Component based diagnosis

Given a system-level characteristic trace and a system broken up into separate components, the component-level diagnosis uses the original single component diagnostic idea to search for potential failures or root causes. First the trace is sliced and distributed among separate components of the system along the possible component paths as described in section 3.4. In this step, all inputs and outputs in the original trace are also transformed to inputs and outputs local to the component by using the component's transformation function.

After that, the algorithm takes all the component paths and calculates the

last component where the start condition is not fulfilled (based on the appropriate preceding event in the characteristic trace), then at this component the path is truncated (so that components does not get diagnosed in vain). It is also possible that all start conditions are fulfilled, in this case no truncation is done.

3.6. Algorithm

The complete diagnostic algorithm takes an already decomposed system and a system-level characteristic trace as an input and outputs the diagnosed root causes per component and the start component failures (if there were any). The execution has the following major steps:

1. The characteristic trace is decomposed by components and its fragments are distributed based on the decomposition of the system. This step, described in Section 3.4 in detail, takes the whole characteristic trace as an input and outputs a sequence of pairs of components and corresponding trace fragments.
2. Based on the component pair sequences of the preceding step and the structure of the system, all possible component paths are created and trimmed adequately based on their start conditions as described in Section 3.5.
3. The single component diagnostic procedure described in Section 2.4 is executed for all valid (non-trimmed) components in every component path (local deviations are determined and based on them the reasoning procedure is initiated on the local P-HAZID table). The diagnosed root causes, and the set of terminal deviations (from which the diagnosis cannot proceed forward) are collected per component.
4. The algorithm outputs the root causes, terminal deviations and the list of components which were not diagnosed due to start condition failures.

As its single component equivalent, this component-based diagnosis also has P-HAZID table validation capabilities (the terminal deviations are also returned as root causes).

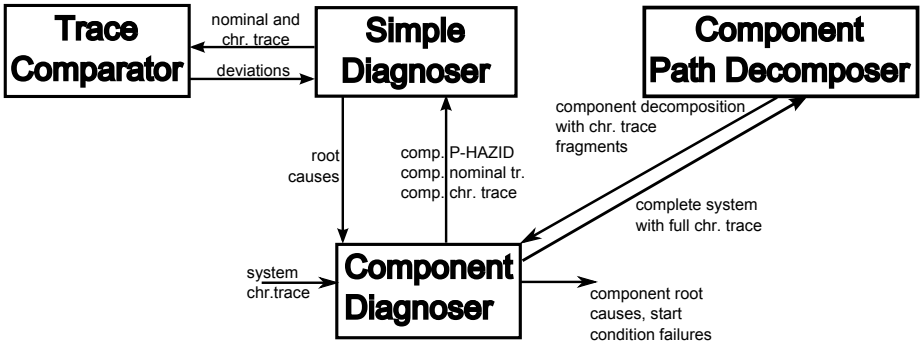

Figure 1: Connections between major parts of the algorithm. Abbreviations: chr=characteristic, comp.=component, tr.=trace .

Based on this, the major components of the algorithm and the connections between them can be seen in Figure 1.

The algorithm had been implemented in Scala [2], which is an object oriented functional language with very similar syntax to Java [1]. Only the core parts of the algorithm were developed in a "proof of concept" manner, and they were just executed by a properly configured unit test environment (using [3]) to demonstrate the functionality.

4. Case studies

In this section the operation of the algorithm is demonstrated in two simple process systems both consist of similar tanks connected with pipes and valves. In both cases the identified components in the process systems are these tanks.

4.1. A simple serial case study

The first case study is a serially connected process system of three similar tanks (seen in Figure 2) and a simple operational procedure to fill up all three tanks (TA, TB and TC), described in Table 2 in detail. The first three events of the operational procedure relate to tank TA, the events from the third to the fifth relate to the tank TB, while the last three events relate to tank TC (the

third and the fifth event overlaps between neighboring components). The component graph of the system with the component-level nominal trace fragments as inscriptions to the component nodes is shown in the bottom part of Figure 2.

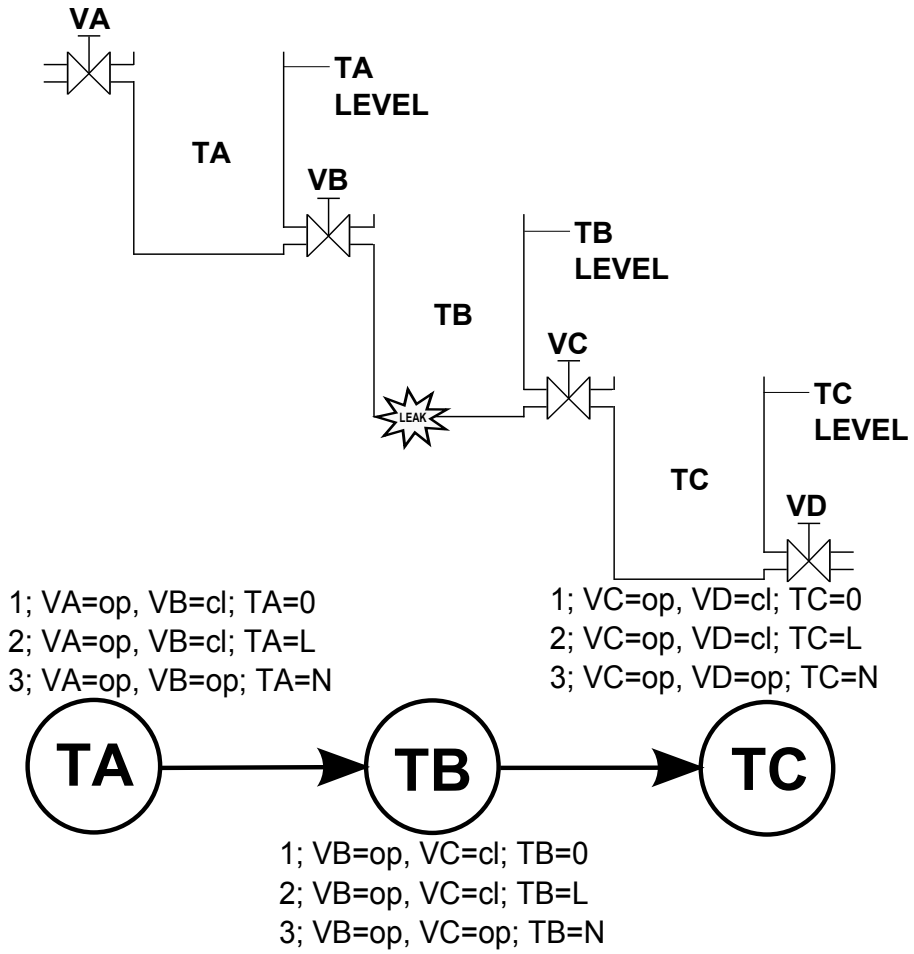

Figure 2: Serial process system with its component graph for the case study trace in Table 2.

At the beginning of the procedure the input valve is opened, and after the tank fills, its output valve is opened. Finally the output valve is opened on the last tank so that the fluid might exit the system. The considered fault in this case is the rupture of the second tank, which is preventing it from containing, so

the sensor never shows any fluid inside the tank. The faulty trace is described in Table 3.

The diagnostic algorithm identified one single component path of the system starting from the first tank TA, and assigned fragments of the input trace (with aligned time) to each of them (this can be also seen in Figure 2). This assignment was done in a serial manner, so every component got a different part from the trace, apart from the boundary events which were distributed to neighboring component's traces. The start conditions of the components were the normal level of fluid ("N") in of the preceding tank (so, in case of TB, the level of TA should have been N), due to this, TC was not diagnosed at all (there was no fluid detected in preceding tank TB, regardless the open state of its input valve).

In this case the algorithm could correctly identify the leak during the diagnosis of TB also using the P-HAZID table in Table 1 (but instantiated in three copies for the three tanks as components, respectively) from the following deviation set:

- **smaller**(2;op,cl;L)
- **smaller**(3;op,op;N)
- **never-happened**(2;op,cl;L)
- **never-happened**(3;op,op;N)

Here the first and second input refer to the input and output valve, respectively, and the only output is the level of the tank.

Due to the small resolution of the sensor, a **later** and a **smaller** deviation was also found, because the algorithm started to reason on the path to the "negative level sensor bias" in the table (but could not proceed with this because these deviations were not found in the list of deviations).

Time	Input values				Output values		
	VA	VB	VC	VD	TA	TB	TC
1	op	cl	cl	cl	0	0	0
2	op	cl	cl	cl	L	0	0
3	op	op	cl	cl	N	0	0
4	op	op	cl	cl	N	L	0
5	op	op	op	cl	N	N	0
6	op	op	op	cl	N	N	L
7	op	op	op	op	N	N	N

Table 2: The nominal trace of a simple operational procedure for filling up all tanks in the serial process system of Figure 2.

Time	Input values				Output values		
	VA	VB	VC	VD	TA	TB	TC
1	op	cl	cl	cl	0	0	0
2	op	cl	cl	cl	L	0	0
3	op	op	cl	cl	N	0	0
4	op	op	cl	cl	N	0	0
5	op	op	op	cl	N	0	0
6	op	op	op	cl	N	0	0
7	op	op	op	op	N	0	0

Table 3: A faulty trace of the simple operational procedure for the serial process system (Figure 2) for a rupture in the second tank (TB).

4.2. A simple parallel case study

In the second case study the algorithm execution is demonstrated on a parallel process system, here fluid is fed into a bigger aggregator tank and then it is distributed into two smaller tanks, where it leaves the system through the two outputs at the bottom of the tanks. The objective of the operational procedure is similar to the one in the previous case, namely to fill up all three tanks with liquid and then open the outputs. The aggregator tank's dimensions are different, because the input valve's diameter is increased, so that it can fill up twice as fast as the other tanks, and it has two output valves leading to the other two tank components. The nominal trace of the system is given in Table 2.

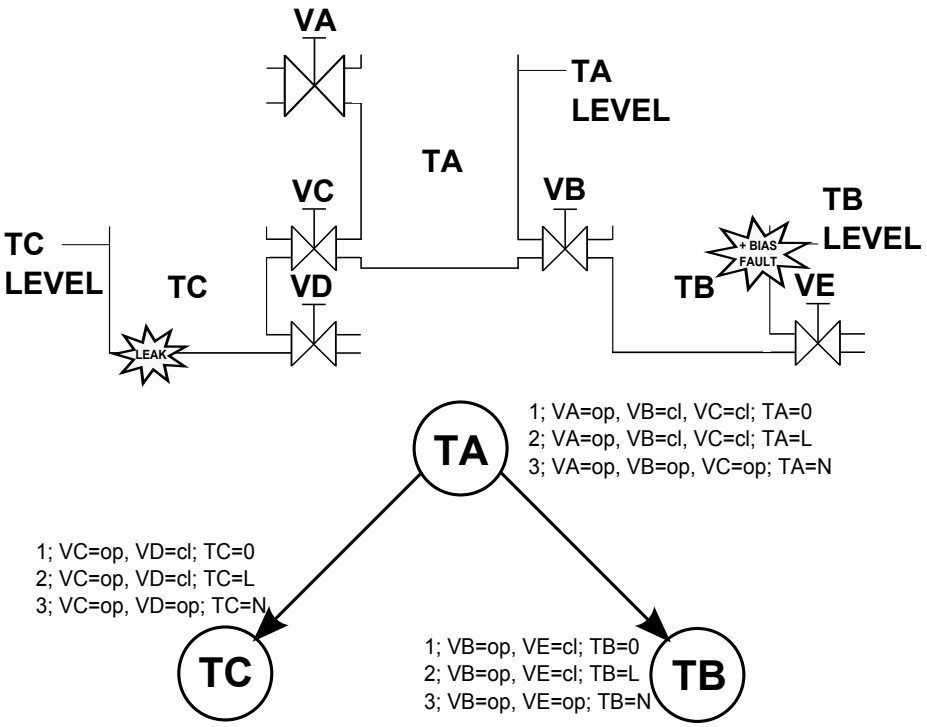

Figure 3: Parallel process system with its component graph for the case study trace available in Table 4.

In this example scenario we added to both output tanks simulated failures: TB had a positive sensor bias failure, while TC had a rupture. The correspond-

Time	Input values					Output values		
	VA	VB	VC	VD	VE	TA	TB	TC
1	op	cl	cl	cl	cl	0	0	0
2	op	cl	cl	cl	cl	L	0	0
3	op	op	op	cl	cl	N	0	0
4	op	op	op	cl	cl	N	L	L
5	op	op	op	op	op	N	N	N

Table 4: Nominal trace of the simple operational procedure to fill up all tanks in the parallel process system of Figure 3.

ing faulty trace is given in Table 5. Due to the parallel nature of the system, these failures shall be diagnosed separately, they are independent of each other. Because of the parallel nature of the components, the algorithm found two different component paths in this scenario (see at the bottom of Figure 3) using the aggregator tank, TA, as a starting component. It should also be noted that in contrast with the serial process system, TB and TC got the same event fragments here (only with different inputs and outputs).

After executing the diagnostic algorithm, the following root causes were found:

- **For Tank TB.** Here the positive bias failure could be identified, this is the only root cause or deviation which had been found as a result of diagnosing this component.
- **For Tank TC.** Here - just like in the case of the first case study - the tank leak failure could be identified, along with a false report of a negative bias failure - this was diagnosed due to the simplicity of the P-HAZID table.

This case study demonstrated that multiple faults can also be correctly identified using the proposed algorithm.

Time	Input values					Output values		
	VA	VB	VC	VD	VE	TA	TB	TC
1	op	cl	cl	cl	cl	0	L	0
2	op	cl	cl	cl	cl	L	L	0
3	op	op	op	cl	cl	N	L	0
4	op	op	op	cl	cl	N	N	0
5	op	op	op	op	op	N	H	0

Table 5: A faulty trace of the operational procedure for the parallel process system (Figure 3) for a positive level sensor bias fault in tank TB and a tank rupture in tank TC.

5. Conclusion and Further work

A novel diagnosis method was described in this paper that uses the component-wise constructed P-HAZID tables and nominal traces of a process system driven by an operational procedure, together with its component graph. The algorithm is capable of finding fault root causes in complex process systems using nominal and observed possible faulty operational procedure execution traces. The algorithm uses the structural decomposition of the complex process system and its component-level dynamic HAZID (P-HAZID) tables and executes the diagnosis component-wise by first decomposing the observed characteristic traces, and then assembling the diagnosis results.

A structural decomposition of the process system and its diagnostic components, similar to the ones in [8] were used for the diagnosis, but in our work the decomposed operational procedure is executed on the results of the structural decomposition, while in [8] the physical connections and fault propagation through them were emphasized.

Two simple case study demonstrated the operation of the diagnostic algorithm and its effect to decrease the redundancy in a complex process system with similar components. The capability of the method for identifying multiple faults has also been demonstrated.

Two possible directions for further work have been identified. Firstly, an

on-line version of the diagnostic procedure is aimed at. Currently the algorithm is working on off-line data, i.e. it is assumed that the diagnosable operational procedure had been completed before. Extending the algorithm with capabilities to support run-time operational procedure events (to diagnose them under execution, and process events as they happen) might add a considerable benefit to the use cases of the diagnostic method. In that way, fault detection could be faster, operational procedure execution could be halted in order to enable plant personnel to begin possible fault mitigation operations before the procedure had been finished.

The other possible further research direction targets the refinement of the P-HAZID tables for improved diagnostic resolution. With a slight modification, arbitrary P-HAZID tables can be validated by the single-component reasoning part by using it to explore all possible reasoning paths in the table. This might help experts during the setup of a P-HAZID table for a complicated component (missing events or faulty paths can be found easier), or the P-HAZID table can be extended in order to provide more accurate diagnostics results.

6. References

[1] http://www.oracle.com/technetwork/java/index.html.

[2] http://www.scala-lang.org/.

[3] http://www.scalatest.org/.

[4] Standards Australia. *Hazard and operability studies (HAZOP studies) Application Guide.* 2003. AS IEC 61882-2003.

[5] Standards Australia. *Analysis techniques for system reliability - Procedure for failure mode and effects analysis (FMEA).* 2008. AS IEC 60812-2008.

[6] V. Bartolozzi, L. Castiglione, A. Piciotto, et al. Qualitative models of equipment units and their use in automatic hazop analysis. *Reliability Engineering and Systems Safety*, 70:49–57, 2000.

[7] I. T. Cameron and R. Raman. *Process Systems Risk Management*, volume 6 of *Process Systems Engineering*. Elsevier Academic Press, San Diego, CA, 2005.

[8] H. A. Gabbar. Improved qualitative fault propagation analysis. *Journal of LossPrevention in the Process Industries*, 20:260–270, 2007.

[9] E. Németh and I. T. Cameron. Cause-implication diagrams for process systems: Their generation, utility and importance. *Chemical Engineering Transactions*, 31:193–198, 2013.

[10] C. Palmer and P. W. H. Chung. An automated system for batch hazard and operability studies. *Reliability Engineering and System Safety*, 94:1095–1106, 2009.

[11] B. J. Seligmann, E. Németh, K. Hangos, and Ian T. Cameron. A blended hazard identification methodology to support process diagnosis. *Journal of Loss Prevention in the Process Industries*, 25:746–759, 2012.

[12] A. Tóth, K. M. Hangos, and Á. Werner-Stark. Hazid information based operational procedure diagnosis method. In *12th International PhD Workshop on Systems and Control*, pages 1–6, Veszprém, Hungary, August 2012.

[13] A. Tóth, K. M. Hangos, and Á. Werner-Stark. A model based diagnosis method for discrete dynamic processes using event sequences. In *Factory Automation 2013 Conference*, pages 114–119, Veszprém, Hungary, `http://daedalus.scl.sztaki.hu/PCRG/works/publications/Toth2013.pdf`, May 2013.

[14] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri. A review of process fault detection and diagnosis part I: Qualitative model-based methods. *Computers and Chemical Engineering*, 27:293–311, 2003.

[15] V. Venkatasubramanian, J. S. Zhao, and S. Viswanathan. Intelligent systems for hazop analysis of complex process plants. *Computers and Chemical Engineering*, 24:2291–2302, 2000.

[16] Á. Werner-Stark, Erzsébet Németh, and K. M. Hangos. Knowledge-based diagnosis of process systems using procedure hazid information. In *15th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 385–394, 2011.