2020 17th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)
Mexico City, Mexico. November 11-13, 2020

Removing Omissions and Inconsistencies from the ISA 5.1 Narratives of Industrial Processes

1st D. Rozo-Ibañez
Dept. of Electrical Engineering
CINVESTAV-Guadalajara
Zapopan, Jalisco, México
durvvin.rozo@cinvestav.mx

2nd J. Ruiz-León
Dept. of Electrical Engineering
CINVESTAV-Guadalajara
Zapopan, Jalisco, México
javier.ruiz@cinvestav.mx

3rd D. Guevara-Lozano
Dept. of Electrical Engineering
CINVESTAV-Guadalajara
Zapopan, Jalisco, México
daniel.guevara@cinvestav.mx

4th C.R. Vázquez
Escuela de Ingeniería y Ciencias
Tecnologico de Monterrey
Zapopan, Jalisco, México
cr.vazquez@tec.mx

5th Y. Yuliana Rios
Dept. of Mechatronics Engineering
Universidad Tecnológica de Bolívar, Campus Tecnológico
Bolívar, Colombia
yrios@utb.edu.co

***Abstract*—This work is concerned with the modelling of industrial processes, described by Pipe and Instrumentation Diagrams (P&ID) and narratives as specified in the norm ISA 5.1, by Interpreted Petri Nets (IPN). In a previous work, a modeling methodology was introduced in order to translate a P&ID representation into an IPN. Now, herein the proposed methodology is enriched with the inclusion of errors detection and recovery stages that allow to remove errors involuntarily introduced by engineers in the P&ID and the process and operation narratives. In particular, the detection stage searches for omitted or inconsistent information involuntarily introduced in the narratives. Every omitted or inconsistent information is referred as an error. The errors are detected by evaluating logical predicates given in a list of predicates, which can be extended to include new type of errors. The recovery stage is then implemented by the execution of functions that focus on removing the detected errors. These functions fix the information in some tables, this information is then propagated to the narratives and the derived IPN models, in order to make consistent the information in the model.**

***Index Terms*—P&ID, Petri nets, ISA, industrial processes**

I. INTRODUCTION

An industrial process consists of sub-processes or unit operations, commonly represented by international standards such as the ANSI ISA S5.1. This standard describes graphically the elements (sensors, actuators, transmitters), and the relation among them. It also captures the process and operation behaviors (narratives), where the first one describes the system operation and the latter describes the required system behavior. Frequently, both behaviors are described in natural language, and although it can be easily understood, it is commonly a source of potential errors such as ambiguities, inconsistencies, and omissions in the information [1], [2]. From the information provided by the P&ID, the control program is designed based on the expertise of the programmer, who may introduce involuntary errors in the ladder code, requiring further time consuming debugging stages. In order to face these problems, formal tools and modeling methodologies have been proposed [3]–[8], mainly based on Finite Automata or Petri nets [9]. Unfortunately, none of these methodologies focused on capturing and removing the errors introduced in the narratives.

It is known that the information on industrial processes given in natural language can be represented in structured language through tables; it is estimated that from 1% to 5% of the information in the cell is wrong [10], bringing serious consequences for the process and the industry. Detection of omissions, ambiguities and inconsistencies allows their reduction by identifying requirement statements, which can be interpreted in multiple ways. For requirements written in natural language, different reading techniques are used, such as inspection-based reading [11], [12], scenario-based reading [13], or object-oriented inspections [14]. Anda and Sjøberg apply the reading technique to use case models and propose a checklist-based to detect inconsistencies and ambiguities [15]. Unfortunately, these techniques are rarely used in the industrial environment, with the consequence that an error in the process information can cause major failures.

In a previous work [16], a methodology for translating a system represented as a P&ID into a Petri net model was introduced. In that methodology, the information provided by the P&ID and the process and operation narratives is codified into standard tables. Next, those tables are translated into two Petri nets, one representing the system behavior (the model of the system) and another representing the required behavior (the specification to be imposed by the controller). Although the resulting system and specification models are derived to be used in the regulation control framework, any other, such as supervisory control theory can be used.

Now we assume that the system is modeled using the methodology reported in [16]. Based on this model and tables,

978-1-7281-8987-1/20$31.00

the problem herein addressed deals with the detection of omissions and inconsistencies in the translation of process and operation narratives into tables, considering in particular the removal of errors that might be included during the filling of the tables. The approach herein presented assumes that the potential error types are a priori known, representing omissions and inconsistencies in the tables (named potential errors). The proposed methodology consists of two stages. In a first stage, the tables are reviewed in order to detect potential errors, and in a second stage, an action is performed for each detected error in order to remove it from the tables. Once the tables are error-free, the translation methodology can be applied to generate the Petri net models.

This paper is organized as follows: Section II provides basic definitions of *PN* and the fundamentals of industrial processes. Section III recalls the translation methodology of a P&ID representation to a *IPN* model and an example is presented in order to illustrate it. Section IV describes the methodology to detect the errors introduced in the table structure and to remove them. Then Section V illustrates the proposed methodology using an example. Finally, some conclusions and future work are presented in Section VI.

II. Basic concepts

This section recalls fundamentals on Petri nets and industrial processes.

A. Petri nets

Definition 1. *A Petri net (PN) structure is a bipartite digraph represented by the 4-tuple $G = \langle P,T,I,O\rangle$, where $P = \{p_1, p_2, ..., p_n\}$ is a finite set of places, $T = \{t_1, t_2, ..., t_o\}$ is a finite set of transitions, $I : P \times T \to \mathbb{Z}_{\geq 0}$ is a function representing the weighted arcs connecting places to transitions, and $O : P \times T \to \mathbb{Z}_{\geq 0}$ is a function representing the weighted arcs connecting transitions to places.*

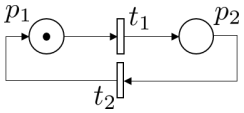

Fig. 1: Petri net example.

Pictorially, places are represented by circles, transitions by rectangles, and arcs by arrows as depicted in Fig. 1. The *incidence matrix* of a *PN* is a $|P| \times |T|$ matrix $\mathbf{C}$ defined such that $\mathbf{C}[i, j] = O(p_i,t_j) - I(p_i,t_j)$. The symbol ${}^\bullet t_j$ (resp. ${}^\bullet p_i$) denotes the set of all places p_i (resp. transitions t_j) such that $I(p_i,t_j) \neq 0$ (resp. $O(p_i,t_j) \neq 0$). Similarly, $t_j^\bullet$ (resp. $p_i^\bullet$) denotes the set of all places p_i (resp. transitions t_j) such that $O(p_i,t_j) \neq 0$ (resp. $I(p_i,t_j) \neq 0$).

Definition 2. *Given a PN structure, the* marking *distribution is defined as a function $M : P \to \mathbb{Z}_{\geq 0}$, where $M(p_i)$ represents the number of tokens residing inside the place p_i (depicted as dots). The marking distribution is expressed as a column vector $\mathbf{M}$ of length $|P|$, where $\mathbf{M}[i] = M(p_i)$, $\forall p_i \in P$. A PN* system *is a pair $\langle G, \mathbf{M}_0\rangle$, where G is a PN structure and $\mathbf{M}_0$ is the initial marking distribution. The marking distribution evolves according to the firing of transitions. A transition t_j is* enabled *at a marking $\mathbf{M}_k$ if $\forall p_i \in {}^\bullet t_j$, $\mathbf{M}_k[i] \geq I(p_i,t_j)$, this is denoted as $\mathbf{M}_k \xrightarrow{t_j}$. A transition t_j can fire if it is enabled. The firing of an enabled transition t_j leads to a new marking $\mathbf{M}_{k+1}$ that can be computed with the so-called PN* fundamental equation *$\mathbf{M}_{k+1} = \mathbf{M}_k + \mathbf{C}\mathbf{v}_k$, where $\mathbf{v}_k[i] = 0$ for $i \neq j$ and $\mathbf{v}_k[j] = 1$.*

Definition 3. *An Interpreted Petri net (IPN) system is a tuple $Q = \langle G, \mathbf{M}_0, \Sigma_I, \Sigma_O, \lambda, \varphi\rangle$, where $\langle G, \mathbf{M}_0\rangle$ is a PN system; Σ_I is the input alphabet, where each element of the set Σ_I is an input symbol; Σ_O is the output alphabet, where each element of the set Σ_O is an output symbol; $\lambda : T \to 2^{\Sigma_I}$ is the input-labeling function of transitions (a single transition can be associated with more than one symbol from the input alphabet Σ_I); $\varphi : P \to 2^{\Sigma_O}$ is the labeling function of places (a single place can be associated with more than one output symbol).*

The evolution of an *IPN* is similar to that of the *PN* system with the addition of the following rules: 1) a symbol $a \in \Sigma_I$ is said to be *indicated* if it is activated by an external device (e.g., a controller or a engineer) or $\exists p \in P$ such that $a \in \varphi(p)$ and $M(p) > 0$; 2) a transition t_j can fire iff t_j is enabled and $\forall a \in \lambda(t_j)$ it holds that a is indicated.

B. Industrial processes

An industrial process is usually described by a *Piping and Instrument Diagram* (P&ID), a *Process narrative* ($\mathcal{PN}$), and an *Operation narrative* ($\mathcal{ON}$).

A P&ID is a graphic representation that provides important information for the manufacturing and interconnection of the equipment and machinery, piping, control and safety elements, essential for the correct operation of the process [17]. This diagram includes symbols and an identification code to establish a uniform means of designating instruments used for measurement and control. This is based on the standards issued by the Instrumentation Systems and Automation Society (ISA). Each variable appearing in a P&ID is named *controlled variable* and the values that this variable can take are its *variable states*. In a similar way, the *actuator states* are the values that an actuator that appears in a P&ID can take. We denote the number of the possible states in a controlled variable or actuator as its *range*. Both the controlled variables and the actuators are referred as the process *elements*. The elements controlling, acting or sensing a controlled variable form a *control loop* in the P&ID; control loops are gathered to form *Sub-processes*. In this work, we consider processes in which the set of actuators and variables states are discrete.

In addition to the graphic representation, a process and control narratives are used to describe the behavior of the process. The *Process narrative* ($\mathcal{PN}$) describes in natural language the process functionality. This narrative describes the evolution of the controlled variables caused by the actuators action.

The *Operation narrative* ($\mathcal{ON}$) is a set of sentences describing the required behavior of the system. In particular,

this narrative indicates ordered sequences of state values that the actuators and controlled variables must reach in order to produce goods as expected.

III. Modelling methodology

As mentioned in the Introduction, this work is based on the modelling methodology presented in [16] to obtain IPN models from the P&ID and process narratives, see [16] for details. The application of the methodology is illustrated in the following example.

Example 1. *The P&ID depicted in Fig. 2 represents a bottle filling system. This process is described as follows.*

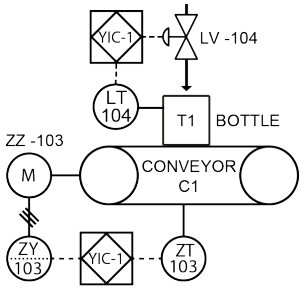

Fig. 2: *P&ID* of a bottle filling system.

Process narrative. The system has two sub-processes. The first one is the bottle transfer ($C-1$) that is composed of a position transmitter ZT-103, and a motor ZZ-103. The transmitter ZT-103 is used to detect bottle positions over the conveyor, defining the controlled variable of the bottle position $BP-103$ with a range two, where it can be either at a presence state (BP^p) or at an absence state (BP^a). The motor ZZ-103 of range two can be either at the on state or at the off state. If the motor ZZ-103 is at the state on the conveyor belt conveys bottles to the filling position (a physical barrier is at the end of the conveyor); when the motor ZZ-103 is at the state off the conveyor stops. The second sub-process ($T-1$) is the bottle filling which consists of a level transmitter (LT-104) and a level valve (LV-104). The level transmitter LT-104 is used to detect the liquid level defining the controlled variable of the bottle level $BL-104$ with a range two, it can be at an empty state (BL^e) or $full$ state (BL^f); and the valve $LV-104$ with range two can be either at a closed state or at an open state. If the valve LV-104 is at open state, then the state of $BL-104$ can change from BL^e to BL^f. The complete process is controlled by the controller YIC-1.

Operation narrative. The initial condition of ZT-103 is BP^a, and the initial condition of BL-104 is BL^e. The complete system must perform the following task: at the initial conditions, take a bottle to the filling position of the conveyor (BP^p) and fill it (BL^f), then retrieve the bottle (BP^a).

A. Deriving IPN models

By following the methodology proposed in [16], a structure $\mathcal{T}$ of tables consisting of Tables I to VI is derived.

The *element description table* (Table I) lists the controlled variables and actuators with their characteristics (range, states, and initial conditions). The *element behavior tables* (Table II) capture the element behavior described in the process narrative. A check-mark in the entry i,j represents that the element can change from the $i-th$ state to the $j-th$ state in a single step. The *permissive relation table* (Table III) captures the case when controlled variables evolve only if certain actuators are at some particular states. The *Synchronous relation table* indicates when two or more controlled variables evolve simultaneously. The *Operation states table* (Table V) lists the elements and their operation states as given by the P&ID and the operation narrative. The *Operation conditions table* describes each required sub-process as a sequence of operations, whose occurrence may be conditioned by the states of the controlled variables. A few errors were intentionally included in these tables to illustrate the application of an error-detection methodology in a forthcoming example.

Once the tables are filled, the methodology continues translating tables into the *Process Graph (P_G)* (Fig. 3) and later

TABLE I: Element Description ($\mathcal{T}_1$)

Controlled variables & Actuators	Range	State	Initial State
ZZ-103(Motor)	2	off, on	off
LV-104 (Valve)	2	closed	closed
Bottle-Position ($BP-103$)	2	absence (BP^a), presence (BP^p)	absence
Bottle-level (BL-104)	2	empty (BL^e), full (BL^f)	empty

TABLE II: Element Behavior tables ($\mathcal{T}_2$)

ZZ-103	off	on
off		✓
on	✓	

LV-104		closed
closed		

BP-103	BP^a	BP^p
BP^a		✓
BP^p	✓	

BL-104	BL^e	BL^f
BL^e		✓
BL^f	✓	

TABLE III: Permissive relation ($\mathcal{T}_3$)

		Controlled variables	
		$BP-103$	$BL-104$
Actuator	**Actuator state**	**Events**	
ZZ-103	off	-	-
	on	BP^a to BP^p BP^p to BP^a	-
LV-104	closed	-	-
		-	BL^e to BL^f

TABLE IV: Synchronous relation ($\mathcal{T}_4$)

Variables and events		$BP-103$		$BL-104$	
		BP^a to BP^p	BP^p to BP^a	BL^e to BL^f	BL^f to BL^e
$BP-103$	BP^a to BP^p				
	BP^p to BP^a				X
$BL-104$	BL^e to BL^f				
	BL^f to BL^e		X		

TABLE V: Operation states ($\mathcal{T}_5$)

Elements	Operation states
$BP-103$	BP^a, BP^p
$BL-104$	BL^e, BL^f

TABLE VI: Operation conditions ($\mathcal{T}_6$)

	Pre-condition states	Conditions	Target states
Bottle filling	$\{BP^a, BL^e\}$		$\{BP^p, BL^f\}$
	$\{BP^p, BL^f\}$		$\{BP^a, BL^e\}$

into an IPN (Fig. 4). Each vertex in P_G represents a process element, solid arcs represent the permissive relation among them, and dashed arcs represent the synchronous relation among controlled variables. In the IPN, the permissive relation is captured as self-loop arcs, and the synchronous relation is captured as join transitions. The IPN of Fig. 4 represents the system model.

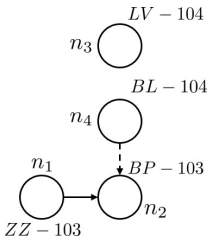

Fig. 3: Process graph for the bottle filling process.

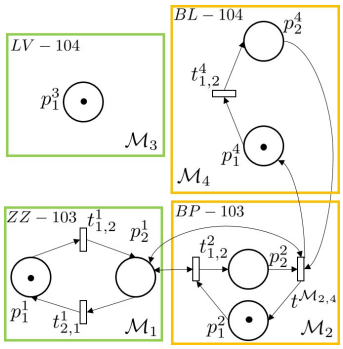

Fig. 4: Process IPN model of the bottle filling process.

According to [16], the operation states and conditions tables are used to derive the specification (the behavior to be imposed by the controller). The specification is represented by the IPN depicted in Fig. 5.

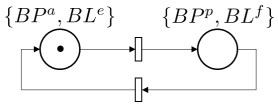

Fig. 5: Specification IPN model for the Example 1.

IV. Errors detection and recovery methodology

The methodology introduced in [16] translates the P&ID information into tables an then to an *IPN*. Since P&ID is an structured and unambiguous information, the translation process is free of errors. It is not the case for narratives, since they are described in natural language and may include involuntary errors that may be translated into tables. This section introduces a novel mechanism to determine if there exist errors in the narratives by analyzing the tables to detect omissions or inconsistencies (i.e. errors) in the information. If there exist errors, the engineer is asked for the concise information needed to remove the errors. Once the tables are updated without errors, the information is translated into an *IPN* as in [16].

Table VII describes the potential errors that can be detected, while Tables VIII and IX describe the corrective actions (functions) that must be performed to remove errors. In order to remove detected errors, the corresponding functions must be applied, i.e.:

"IF error $\mathcal{T}_i - \mathcal{E}o_x^k$ is detected THEN apply the error removing function $\mathcal{T}_i - \mathcal{F}o_x^k$".

Hence, the proposed methodology is composed of two stages, the first one focuses on detecting and isolating errors (the premise clause in the IF-THEN sentence), and the second one focuses on removing the errors (the conclusion clause in the IF-THEN sentence). This methodology is schematically represented in Fig. 6. It is important to remark that new types of errors can be considered just by adding their descriptions in Table VII and their corrective functions in Tables VIII and IX.

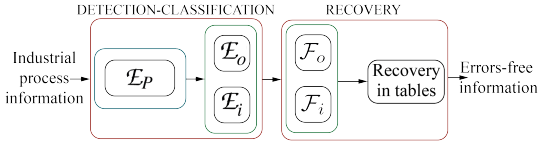

Fig. 6: Errors detection and recovery methodology.

A. Errors detection and classification

The set of potential errors $\mathcal{E}_P$ is partitioned in $\mathcal{E}_P = \mathcal{E}_O \dot{\cup} \mathcal{E}_I$, where $\mathcal{E}_O = \{\mathcal{E}o^m\}$ is the set of omission errors, and $\mathcal{E}_I = \{\mathcal{E}i^n\}$ is the set of inconsistency errors. These errors are described in Table VII. It is said that an error $\mathcal{E}_m^k$ has occurred if its logical predicate described in Table VII becomes true. For example, if there exists an element $\mathcal{E}_m$ in the P&ID but its range is not mentioned in the narratives, then there is a row k in the element table with its entry name equal to $\mathcal{E}_m$ and Range entry empty; thus, error $\mathcal{T}_1 - \mathcal{E}o_r^k$ becomes true since its logical predicate "*There is an empty cell $k,2$ in Range column*".

The following algorithm is used to detect errors in the system table structure.

Algorithm 1: Error detection algorithm

Result: Set of detected errors S_E
Select table $\mathcal{T}_i$
FOR k=1 to number of elements in $\mathcal{T}_i$ DO
IF $\mathcal{T}_i - \mathcal{E}o_x^k$ THEN type error x is detected in table $\mathcal{T}_i$ in the $k-th$ row. Add this error to S_E.
END FOR;

B. Errors recovery

In order to remove an error $\mathcal{E}_x^k$, the associated recovery function $\mathcal{F}_x^k$ must be executed. The associated function consists in asking the engineer for the required information

(in the case of omissions) or removing the inconsistency (in the case of inconsistencies).

For example, when the error $\mathcal{E}_O^1$ is detected, the function $\mathcal{F}_O^1$ must be executed, this function is applied by asking the engineer for the omitted variable range.

The following algorithm is used to remove detected errors. The mentioned S_E set is the one computed in the previous algorithm, it contains all detected errors.

Algorithm 2: Error recovery algorithm

Result: Set of tables free from errors
WHILE $S_E \neq \emptyset$ DO
IF type error x detected in table $\mathcal{T}_i$ in the $k-th$ row is in S_E THEN
Remove this error from S_E
Execute recovery function $\mathcal{T}_i - \mathcal{F}o_x^k$
END IF
END WHILE;

The recovery functions are described in Tables VIII and IX. The first one focuses in removing omission errors while the second one focuses in removing inconsistency errors. Notice that error $\mathcal{T}_i - \mathcal{E}o_x^k$ is removed by function $\mathcal{T}_i - \mathcal{F}o_x^k$.

V. Illustrative Example

The bottle filling system described in Example 1 is used to illustrate the proposed error detection and recovery methodology. The application of the methodology leads to the following steps:

1) An error $\mathcal{T}_1 - \mathcal{E}i_s^3$ is detected in the Element Description Table I. This error is detected because in the row 3 associated to the actuator LV-104, the Range states that the valve could be in two different states, but only the state closed is given; i.e. there is an inconsistency between the number of states and the listed states. Formally: $\mathcal{T}_1 - \mathcal{E}i_s^3$ is true, because $\mathcal{T}_1[3,2] \neq |\mathcal{T}_1[3,3]|$.
2) According to Table IX, this error is recovered by asking the engineer the actuator states LV-104 executing the function: $\mathcal{T}_1 - \mathcal{F}i_s^3$ is executed for the cell $\mathcal{T}_1[3,3]$.
In this case the engineer introduces the new state "opened" for the LV-104 valve and the inconsistency is removed.
3) Once the element description table (Table I) is edited, the correction of this error is propagated through the narratives, and to the process graph and the IPN. Hence the errors are removed and the information in the narratives and graphs becomes consistent with the system structure tables.
When the recovery functions are executed, the tables, narratives and graphs are changed. The updated tables are shown in Tables X-XII; the updated process graph is shown in Fig. 7, and the updated IPN model is shown in Fig. 8.

TABLE VII: Error from table structure

Error from the system table structure $\mathcal{T}$		
Error	**Description**	**Logical predicates**
Errors into the element description table (name to this table $\mathcal{T}_1$)		
Omission errors		
$\mathcal{T}_1 - \mathcal{E}o_r^k$	The element does not have a declared range. ***r*** is the element range.	There is an empty cell $[k,2]$ in the range column.
$\mathcal{T}_1 - \mathcal{E}o_s^k$	The element does not have its declared states. ***s*** is the element states.	There is an empty cell $[k,3]$ in the states column.
$\mathcal{T}_1 - \mathcal{E}o_{is}^k$	The element does not have its declared initial state. ***is*** represents the element initial state.	There is an empty cell $[k,4]$ in the initial state column.
Inconsistency errors		
$\mathcal{T}_1 - \mathcal{E}i_r^k$	The element range is 1. ***r*** is the element range.	There is a range < 2 in the cell $[k,2]$.
$\mathcal{T}_1 - \mathcal{E}i_s^k$	The element states number and its range are not equal. ***s*** is the element state.	There is an element range in cell $[k,2]$ not equal to the states number in cell $[k,3]$.
$\mathcal{T}_1 - \mathcal{E}i_{is}^k$	The element initial state does not belong to its declared states. ***is*** represents the element initial state.	There is an initial state in cell $[k,4]$ that does not belong to the states of cell $[k,3]$.
Errors into the element behavior Table (name to this table $\mathcal{T}_2$)		
Omission errors		
$\mathcal{T}_2 - \mathcal{E}o_{cm}^k$	The element does not have a check-mark that defines its behavior. ***cm*** is a check-mark.	There is an element without a check-mark that relates its states in cell $[k,:]$.
Inconsistency errors		
$\mathcal{T}_2 - \mathcal{E}i_{cm}^{kk}$	The element has a check-mark that relates two equal states. ***cm*** is a check-mark.	There is a check-mark that relates a same state in cell $[k,k]$.
$\mathcal{T}_2 - \mathcal{E}i_{cm}^k$	The element does not have check-marks that relates it with its previous and next states. ***cm*** is a check-mark.	There is an element without check-mark in cells $[k,k-1]$ and/or $[k,k+1]$.
Error detection in the permissive relation Table (name to this table $\mathcal{T}_3$)		
Omission errors		
$\mathcal{T}_3 - \mathcal{E}o_{as}^k$	The actuator does not have a declared state. ***as*** is the actuator state.	There is an empty cell $[k,2]$ in the actuator state column.
Inconsistency errors		
$\mathcal{T}_3 - \mathcal{E}i_{ec}^k$	The actuator initial state modifies the controlled variable states. ***ec*** are event columns of the controlled variable.	There is an initial condition without its empty cells in the event columns $[k,:]$.
Error detection in the operation states Table (name to this table $\mathcal{T}_5$)		
Omission errors		
$\mathcal{T}_5 - \mathcal{E}o_{os}^k$	The controlled variable does not have its declared operation states. ***os*** are the operation states of the controlled variable.	There is an empty cell $[k,2]$ in the operation states column.
Inconsistency errors		
$\mathcal{T}_5 - \mathcal{E}i_{os}^k$	An operation state does not belong to the declared states of the controlled variable. ***os*** are the operation states of the controlled variable.	There is an operation state in cell $[k,2]$ that does not belong to the states of cell $[k,3]$ of $\mathcal{T}_1$.
Error detection in the operation conditions table (name to this table $\mathcal{T}_6$)		
Omission errors		
$\mathcal{T}_6 - \mathcal{E}o_{ps}^k$	The sub-process does not have its previous operation conditions. ***ps*** is the pre-conditions states.	There are empty cells $[:,2]$ in the pre-conditions states column.
$\mathcal{T}_6 - \mathcal{E}o_{ts}^k$	The sub-process does not have its target state. ***ts*** is the target state.	There are empty cells $[:,4]$ in the target states column.

TABLE X: Recovered row 3 in the Element Description Table

Controlled variables & Actuators	**Range**	**State**	**Initial State**
LV-104 (Valve)	2	closed, opened	closed

TABLE XI: Recovered Element Behavior Table

LV-104	opened	closed
opened		✓
closed	✓	

TABLE VIII: Recovery omission error functions

$\mathcal{F}o$	**Correction action**
$\mathcal{T}_1 - \mathcal{F}o^k_r$	The engineer must provide the range in cell [k,2]
$\mathcal{T}_1 - \mathcal{F}o^k_s$	The engineer must provide the state in cell [k,3]
$\mathcal{T}_1 - \mathcal{F}o^k_{is}$	The engineer must provide the initial state in cell [k,4]
$\mathcal{T}_2 - \mathcal{F}o^k_{cm}$	The engineer must provide the element behavior in cell [k,:]
$\mathcal{T}_3 - \mathcal{F}o^k_{as}$	The engineer must provide the actuator states in cell [k,2]
$\mathcal{T}_5 - \mathcal{F}o^k_{os}$	The engineer must provide the operation states in cell [k,2]
$\mathcal{T}_6 - \mathcal{F}o^k_{ps}$	The engineer must provide the pre-condition state in cells [:,2]
$\mathcal{T}_6 - \mathcal{F}o^k_{ts}$	The engineer must provide the target state in cell [:,4]

TABLE IX: Recovery inconsistency error functions

$\mathcal{F}i$	**Correction action**
$\mathcal{T}_1 - \mathcal{F}i^k_r$	The engineer must provide the range in cell [k,2].
$\mathcal{T}_1 - \mathcal{F}i^k_s$	The engineer must provide the state in cell [k,3].
$\mathcal{T}_1 - \mathcal{F}i^k_{is}$	The engineer must provide the initial state in cell [k,4].
$\mathcal{T}_2 - \mathcal{F}i^{kk}_{cm}$	The engineer must provide the element behavior in cell [k,k].
$\mathcal{T}_2 - \mathcal{F}i^k_{cm}$	The engineer must provide the actuator state in cells [k,k-1] and/or [k,k+1].
$\mathcal{T}_3 - \mathcal{F}i^k_{ec}$	The engineer must provide the element states in cells [k;:].
$\mathcal{T}_5 - \mathcal{F}i^k_{os}$	The engineer must provide the operation states in cell [k,2].

TABLE XII: Recovered Permissive Relation Table

		Controlled variables	
		$BP-103$	$BL-104$
Actuator	**Actuator state**	**Events**	
LV-104	closed	-	-
	opened	-	BL^e to BL^f

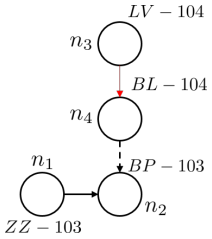

Fig. 7: Updated process graph of the current example.

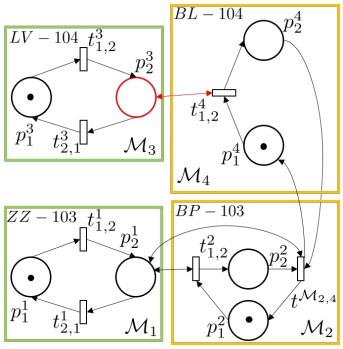

Fig. 8: Updated process IPN model of the current example.

VI. Conclusions

This work introduced a methodology to detect and remove omissions and inconsistencies in the process and operation narratives of an industrial process described by P&ID under norm ISA 5.1. The potential set of errors and recovery functions are given a priori. Thus when an error is detected, then the recovery function is executed in order to remove the detected error. This potential set of error and functions can be extended to include more errors if necessary.

Removing errors updates narratives, tables and graphs, hence the resulting information and models are consistent among them. In the near future, the computed error-free IPN models (system and specification) will be used to automatically design and implement a controller. For instance, to automatically generate the corresponding ladder diagram.

Acknowledgment

The research leading to these results has received support from the Conacyt Fondo Sectorial de Investigación para la Educación, project number 288470.

References

[1] D. Popescu, S. Rugaber, N. Medvidovic, and D. M. Berry, "Reducing ambiguities in requirements specifications via automatically created object-oriented models," in *Monterey Workshop*, ser. Lecture Notes in Computer Science, P. B. and M. C., Eds., vol. 5320. Springer, 2007, pp. 103–124.

[2] F. Pittke, H. Leopold, and J. Mendling, "Automatic detection and resolution of lexical ambiguity in process models," *IEEE Transactions on Software Engineering*, vol. 41, no. 6, pp. 526–544, 2015.

[3] I. Rivera-Rangel, A. Ramírez-Treviño, L. I. Aguirre-Salas, and J. Ruiz-Leon, "Geometrical characterization of observability in interpreted petri nets," *Kybernetika*, vol. 41, no. 5, pp. 553–574, 2005.

[4] A. Ramirez-Trevino, E. Ruiz-Beltran, I. Rivera-Rangel, and E. Lopez-Mellado, "Online fault diagnosis of discrete event systems. a petri net-based approach," *IEEE Transactions on Automation Science and Engineering*, vol. 4, no. 1, pp. 31–39, 2007.

[5] M. Skoldstam, K. Akesson, and M. Fabian, "Modeling of discrete event systems using finite automata with variables," in *Decision and Control, 2007 46th IEEE Conference on*. IEEE, 2007, pp. 3387–3392.

[6] N. Kim, D. Shin, R. Wysk, and L. Rothrock, "Using finite state automata (fsa) for formal modelling of affordances in human-machine cooperative manufacturing systems," *International Journal of Production Research*, vol. 48, no. 5, pp. 1303–1320, 2010.

[7] H. A. Awad, "Modeling of industrial productivity processes," *J. Eng. Sci. JES*, vol. 10, pp. 763–781, 2010.

[8] L.-P. Chung and C.-T. Chang, "Petri-net models for comprehensive hazard analysis of mocvd processes," *Computers & chemical engineering*, vol. 35, no. 2, pp. 356–371, 2011.

[9] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[10] S. G. Powell, K. R. Baker, and B. Lawson, "Errors in operational spreadsheets: A review of the state of the art," in *2009 42nd Hawaii International Conference on System Sciences*. IEEE, 2009, pp. 1–8.

[11] E. Kamsties and B. Peach, "Taming ambiguity in natural language requirements," in *Proceedings of the Thirteenth international conference on Software and Systems Engineering and Applications*, 2000.

[12] E. Kamsties, D. M. Berry, B. Paech, E. Kamsties, D. Berry, and B. Paech, "Detecting ambiguities in requirements documents using inspections," in *Proceedings of the first workshop on inspection in software engineering (WISE'01)*, 2001, pp. 68–80.

[13] E. Kamsties, "Understanding ambiguity in requirements engineering," in *Engineering and Managing Software Requirements*. Springer, 2005, pp. 245–266.

[14] F. Shull, G. H. Travassos, J. Carver, and V. R. Basili, "Evolving a set of techniques for oo inspections," Tech. Rep., 1999.

[15] B. Anda and D. I. Sjøberg, "Towards an inspection technique for use case models," in *Proceedings of the 14th international conference on Software engineering and knowledge engineering*, 2002, pp. 127–134.

[16] D. Rozo-Ibañez, J. Ruiz-León, D. Guevara-Lozano, and C. R. Vázquez, "Petri net modelling of industrial process from a p&id description," in *7th IEEE International Conference on Control, Decision and Information Technologies (CoDIT2020)*, 2020, p. Accepted.

[17] M. Toghraei, "Principles of p&id development: the tips provided here will streamline efforts to develop piping & instrumentation diagrams," *Chemical Engineering*, vol. 121, no. 4, pp. 62–72, 2014.