Chapter 9

AN ONTOLOGY FOR IDENTIFYING CYBER INTRUSION INDUCED FAULTS IN PROCESS CONTROL SYSTEMS

Jeffrey Hieb, James Graham and Jian Guan

Abstract This paper presents an ontological framework that permits formal representations of process control systems, including elements of the process being controlled and the control system itself. A fault diagnosis algorithm based on the ontological model is also presented. The algorithm can identify traditional process elements as well as control system elements (e.g., IP network and SCADA protocol) as fault sources. When these elements are identified as a likely fault source, the possibility exists that the process fault is induced by a cyber intrusion. A laboratory-scale distillation column is used to illustrate the model and the algorithm. Coupled with a well-defined statistical process model, this fault diagnosis approach provides cyber security enhanced fault diagnosis information to plant operators and can help identify that a cyber attack is underway before a major process failure is experienced.

Keywords: Process control systems, security, conceptual modeling, ontology

1. Introduction

Process control systems play a central role in the operation and management of many critical infrastructures, including the electric power grid, water treatment facilities, and chemical and industrial manufacturing plants. In the early days, process control systems were isolated and used proprietary, purpose-built software and hardware. Today, these systems are increasingly connected using Internet technologies, open or semi-open SCADA protocols, and commercial hardware and software. This environment, coupled with the changing landscape of telecommunications and computer networks, introduce the cyber dimension to the security (and safety) of process control systems [1, 10]. While major disasters have thus far been avoided, the recent penetration of a water treatment facility in Harrisburg, Pennsylvania [4] and the Idaho National Lab-

C. Palmer and S. Shenoi (Eds.): Critical Infrastructure Protection III, IFIP AICT 311, pp. 125–138, 2009.

oratory experiment involving the destruction of an electrical power generator [6] indicate that cyber intrusions are a very real threat.

The possibility of cyber intrusions raises several challenges related to control systems security – understanding and assessing risk, integrating cyber security in process control operations, and enforcing a security policy across multiple heterogeneous systems [1, 10]. Another major challenge is fault diagnosis. Fault diagnosis involves the determination of the cause of an identified process fault or abnormal process behavior. Traditionally, fault diagnosis has limited its scope to identifying faulty physical components (e.g., pumps and valves). However, increased network convergence in process control environments means that the detected faults (or abnormalities) in a process could be the result of cyber intrusions instead of component failure.

This paper proposes an ontological model for process control systems. The model supports the formal and explicit representation of a process control system and the process being controlled. A fault diagnosis algorithm is developed based on the ontological model and statistical process models. The algorithm reasons about possible and likely fault sources, including sources that indicate a cyber intrusion. Specifically, it provides cyber security enhanced fault diagnosis information to plant operators, alerting them to a possible cyber intrusion before a process failure is experienced.

The approach is related to the human-assisted intrusion detection technique for process control systems developed by Naedele and Biderbost [12] in that it involves plant operators in security activities. Naedele and Biderbost's approach provides process control operators with visual information generated from network-based intrusion detection metrics, enabling operators to translate their native experience with process monitoring to detect "unhealthy conditions" in the network. The ontological model and fault diagnosis algorithm presented in this paper differ from Naedele and Biderbost's approach by combining traditional process information and process control system information in a formal, precise and semantically-rich manner.

2. Ontological Modeling

An ontological model defines a set of constructs and rules for modeling a specific domain (e.g., the domain of process control systems) at a high level of abstraction. The model is "an explicit specification of a conceptualization" for the domain [8] and corresponds to a formal and precise definition (specification) of the conceptualization [8, 15]. An ontological model of a process control system allows the sharing of important properties of interconnected control system elements and their interoperability.

Recent research has identified the lack of a sound, comprehensive, theoretical basis for conceptual modeling [15]. Several ontological theories and, in particular, Bunge's ontology [2], have been proposed to provide a theoretical basis. Meanwhile, research on ontologies and conceptual modeling has focused on the evaluation of modeling languages and general modeling issues [3, 5, 14, 15]. Efforts have also been directed at domain ontology construction. It is often

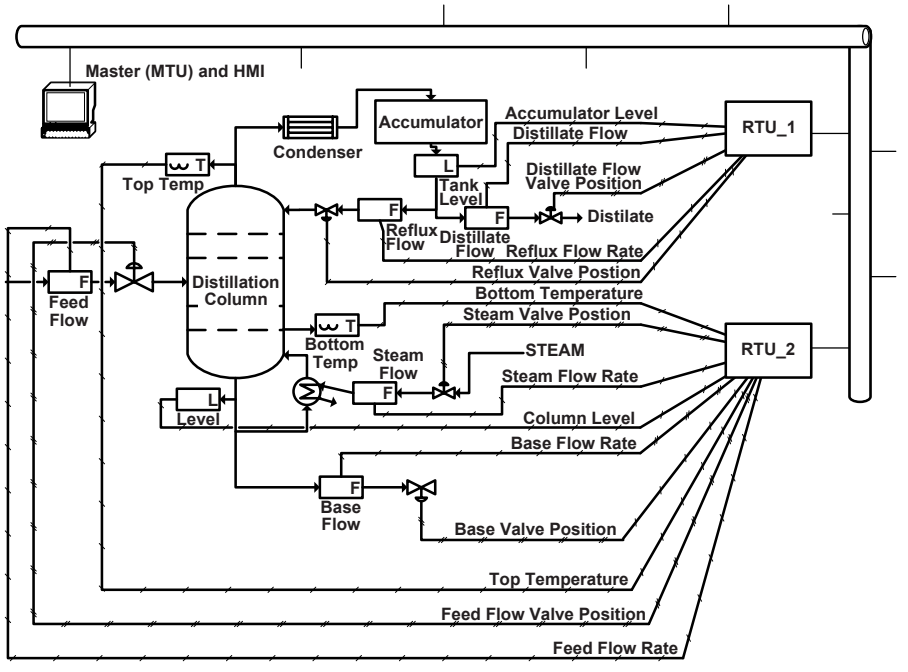

Figure 1. Distillation column system.

impossible to represent even a small part of the world in all its detail. It is more practical to use a limited, essential and relevant number of concepts to describe the static and dynamic aspects of a well-defined part of reality. This set of core concepts forms a conceptualization of that reality. However, such a conceptualization is often informal and ambiguous. A domain ontology is a formal and precise specification of a conceptualization, and is, therefore, concerned with the identification and definition of the essential (static and dynamic) phenomena of the particular domain [16]. A domain ontology for process control systems can help create a common conceptual model for what has become a very heterogeneous mix of technology and systems.

3. Ontological Model for Control Systems

This section presents the constructs used to develop the ontology for process control systems. Bunge's ontological principles [2] form the basis for the constructs. A process control system can be viewed as an information system, possibly embedded in a larger information system. Although Bunge's ontology was not created for information systems, its ontological principles provide a sound theoretical framework for modeling information systems [15].

Figure 1 presents a schematic diagram of a laboratory-scale distillation column and its control system. This distillation column system is used to illustrate the ontological model and fault diagnosis algorithm.

Table 1. Distillation column components and properties.

No.	Component	Properties	Values
3	Accumulator	Level	$\langle 0 \ldots 100 \rangle$
5	Reflux Flow	Flow Rate	$\langle 0 \ldots 100 \rangle$
		Valve Postiion	$\langle 0 \ldots 100 \rangle$
4	Distillate Flow	Flow Rate	$\langle 0 \ldots 100 \rangle$
		Valve Position	$\langle 0 \ldots 100 \rangle$
6	Steam Flow	Flow Rate	$\langle 0 \ldots 100 \rangle$
		Valve Position	$\langle 0 \ldots 100 \rangle$
7	Bottom Flow	Flow Rate	$\langle 0 \ldots 100 \rangle$
		Valve Position	$\langle 0 \ldots 100 \rangle$
1	Feed Flow	Flow Rate	$\langle 0 \ldots 100 \rangle$
		Valve Position	$\langle 0 \ldots 100 \rangle$
2	Distillation Column	Top Temp	$\langle 0 \ldots 100 \rangle$
		Bottom Temp	$\langle 0 \ldots 100 \rangle$
		Column Level	$\langle 0 \ldots 100 \rangle$
8	RTU_1	Analog Input 1	$\langle 0 \ldots 100 \rangle$
		Analog Input 2	$\langle 0 \ldots 100 \rangle$
		Analog Output 1	$\langle 0 \ldots 100 \rangle$
		Analog Output 2	$\langle 0 \ldots 100 \rangle$
		Digital Input 1	$\langle 0 \ldots 100 \rangle$
		Digital Output 1	$\langle 0 \ldots 100 \rangle$
		Digital Input 2	$\langle 0 \ldots 100 \rangle$
		Digital Output 2	$\langle 0 \ldots 100 \rangle$

According to Bunge, the world is made up of substantial individuals or things. Each substantial individual has properties and each property has at least one attribute. Therefore, a process control system can be modeled as a collection of components, each component defined as $X = (x, p(x))$ where x is the component and $p(x)$ denotes the properties of x. A component can be a control system element (e.g., master device, field device or RTU), a conceptual element (e.g., IP network or LAN) that connects control devices, or a SCADA protocol that facilitates communication between devices. Also, a component can be a concept related to the "system under control" (e.g., a flow, temperature or state of a circuit breaker).

Tables 1 and 2 list the components and properties of the distillation column system. The properties of some components (e.g., MTU/HMI) may depend more heavily on the specific system being modeled than others; in such cases, the appropriate properties should be readily identifiable for a given system.

Each component is characterized by a set of state functions, where each state function corresponds to a property of the component. Let X be a component

Table 2. Distillation column components and properties (cont'd.).

No.	Component	Properties	Values
9	RTU_2	Analog Input 1	$\langle 0 \ldots 100 \rangle$
		Analog Input 2	$\langle 0 \ldots 100 \rangle$
		Analog Output 1	$\langle 0 \ldots 100 \rangle$
		Analog Output 2	$\langle 0 \ldots 100 \rangle$
		Digital Input 1	$\langle 0 \ldots 100 \rangle$
		Digital Output 1	$\langle 0 \ldots 100 \rangle$
		Digital Input 2	$\langle 0 \ldots 100 \rangle$
		Digital Output 2	$\langle 0 \ldots 100 \rangle$
10	IP-Network	Source Address	$\{192.168.1.2, 3, 5, 7\}$
		Destination Address	$\{192.168.1.2, 3, 5, 7\}$
11	DNP3	Link Source	$\{0x02, 0x03, 0x05\}$
		Link Destination	$\{0x02, 0x03, 0x05\}$
		Link Direction	$\{0, 1\}$
		Link Function	$\{0x02, 0x03, 0x05\}$
		App Control	$\{0x02, 0x03, 0x05\}$
		App Function Code	$\{0x02, 0x03, 0x05\}$
12	MTU/HMI	Refresh Rate	$\langle 0 \ldots 100 \rangle$
		User Role	$\{Operator, Engineer, Tuner, Sys\_Admin\}$

of a control system. Then, X can be modeled by the functional schema $X_m = \langle M, \tilde{F} \rangle$, where each part of the function $\tilde{F} = \langle F_1, \ldots, F_n \rangle : M \longrightarrow V_1 \otimes \cdots \otimes V_n$ represents a property of X. In this case, M represents the domain of time instances. F_i $(1..n)$ denotes the i^{th} state function of X, $\tilde{F}$ is the total state function of X, and $S(X) = \langle p_i, \ldots, p_n \rangle \in V_1 \otimes \cdots \otimes V_n | p_i = F_i(M)$ is the possible state space of X. Note that the state space of the component Reflux Flow contains all possible combinations of the values of the properties Flow Rate and Valve Position.

Lawful states describe the normal operation of a component; this constrains the values that the properties of a component may take. This construct permits the model to capture and express valid combinations of property values. For example, the DNP3 link layer direction bit (indicating whether a master or outstation sent a particular frame) could be combined with known information about the DNP3 address of the master or outstation, and expressed using laws.

Let $S_L(X) \subseteq S(X)$ be the lawful state space of a component X and let $G_L(X)$ be the set of lawful transformations. Then, a lawful event in X is the ordered pair $\langle s, s \rangle$ where s, $s' \in S_L(X)$ and $s' = g(s)$, $g \in G_L(X)$. For example, for the Distillation Column component, a lawful event resulting from the Feed Flow valve being opened might be: $(\langle 90, 190, 70 \rangle, \langle 85, 150, 88 \rangle)$.

Obviously, an event in one component can affect or act on another component. This relationship is referred to as "coupling," which is defined in terms of the "histories" of the coupled components. The history of a component is simply the set of states that the component has held over time. Let X be a component modeled by a function schema $X_M = \langle M, \tilde{F} \rangle$ and let $t \in M, t > 0$ be a time instant. Then, a history of X is the set of ordered pairs: $h(x) = \langle t, \tilde{F}(t) \rangle$ that can be written as $h(X) = \langle t, (p_1(t), \ldots, p_n(t)) \rangle$. For example the history of the Distillation Column component for two time instances t_0 and t_1 can be expressed as: $h(\text{Distillation Column}) = \langle t_0, (90, 190, 100) \rangle, \langle t_1, (85, 150, 200) \rangle$.

The history notion is used to define when two or more elements interact. A component X acts on component Y if their histories are not independent; this is denoted by $X \triangleright Y$ if $h(Y|X) \neq h(Y)$. Two components X and Y are coupled (written as $B(X, Y)$) if $(X \triangleright Y) \vee (Y \triangleright X)$. For example, Feed Flow and Distillation Column are coupled because a change in Feed Flow affects the bottom temperature property (Bottom Temp) of Distillation Column.

4. Fault Diagnosis

Fault diagnosis seeks to identify the components that have led to a failure in a process control system or part of a process control system. Diagnosing a fault presents a challenge because it is often the case that when a system fails, the components where the failures are observed are not the failure sources, but the victims of faults that have propagated from other parts of the system [9, 13]. The problem is further complicated by the cyber component of process control systems. A cyber intrusion could cause a fault on its own, requiring plant operators to consider if the fault had a conventional source or was the result of a cyber event. Note that process faults are not necessarily catastrophic; more often than not they simply give rise to abnormal process behavior.

Identifying the source of an observed abnormality helps operators quickly isolate and fix the fault source. In the case of faults induced by a cyber intrusion, identifying that the faults have a common ancestor component that can be targeted by a cyber intrusion could warn operators that a cyber attack may be underway. Thus, operators have more time to react to the problem and, hopefully, direct their efforts appropriately.

Before specifying the fault diagnosis algorithm, we provide some terminology related to fault diagnosis and briefly describe the fault diagnosis process [7].

- **Process Fault:** This manifests itself as a difference between the observed behavior and the desired behavior of a process.
- **Fault Detection:** This involves the determination that abnormal behavior has occurred.
- **Fault Isolation/Diagnosis:** This involves the determination of the cause of a fault.
- **Fault Recovery:** This involves the restoration of the system to its proper operating state.

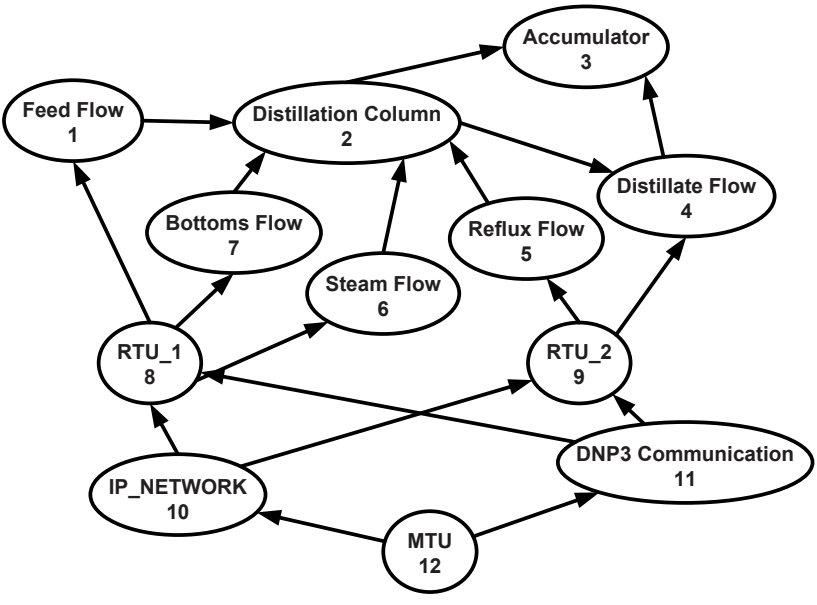

Figure 2. Digraph representation of the distillation column system.

- **Process Monitoring:** This involves the observation of process activity to detect faults and other abnormalities.

Faults must be detected before they can be diagnosed. Every process system has process variables. Manipulated variables are used to control the process; observed variables indicate the status of the process. A mathematical model of the process may be constructed using data gathered from the running system. After the mathematical model is constructed, the values of the current process variables can be compared with the model (process monitoring). Any deviation is considered to be a process fault (fault detection). Readers are referred to [7, 11] for a comprehensive discussion of process monitoring and fault detection.

The ontological constructs defined in the previous section allow the precise and formal representation of key aspects of a process control system. We now demonstrate the utility of the ontological model in fault diagnosis by adapting the Guan-Graham fault diagnosis algorithm [9] to the ontological model for diagnosing faults in a control system.

Let $C = \{X_1, \ldots, X_n\}$ be the set of components in a control system. Then, an impact relation R can be defined on C such that $X_i R X_j$ means that component X_i acts on component X_j, i.e., $X_i \triangleright X_j$. An impact digraph $G = (V, E)$ is then used to represent this relation where $V = \{X_i | X_i \in C\}$ is the vertex set and $E = \{(X_i, X_j) | X_i \triangleright X_j \text{ and } i \neq j\}$ is the edge set. Figure 2 shows the diagraph G for the distillation column example.

Let $A = (a_{ij})$ be the adjacency matrix representing the impact digraph such that $a_{ij} = 1$ if $(X_i, X_j) \in R$ and $a_{ij} = 0$ if $(X_i, X_j) \notin R$. If there is a path

in G from X_i to X_j, X_j is said to be reachable from X_i. For completeness, every vertex in G is defined to be reachable from itself by a path of length 0. As defined above, reachability is transitive. For convenience, the components in the distillation column example are numbered from 1 through 12 as shown in Tables 1 and 2.

The history $h(X)$ for each component can be obtained from data gathered when the system is operating and can be used to develop the impact relation R. For the laboratory-scale distillation column system, the impact relation R is based on input from operators familiar with similar distillation columns.

The adjacency matrix A for the laboratory-scale distillation column system is generated by assigning 1 to every a_{ij} if there is an arrow from element i to element j in G; all the other elements in A are assigned a value of 0. The reachability matrix P may be defined as $P = (A + I)^r = (A + I)^{(r-1)} \neq (A + I)^{(r-2)}$ where I is the identity matrix. P can be generated in $O(n^3)$ time using Warshall's algorithm. The adjacency matrix A and the reachability matrix P for the digraph in Figure 2 are given by:

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

$$P = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

The reachability matrix can be processed to extract important properties [9, 13]. Two partitions may be defined on the reachability matrix P, the level

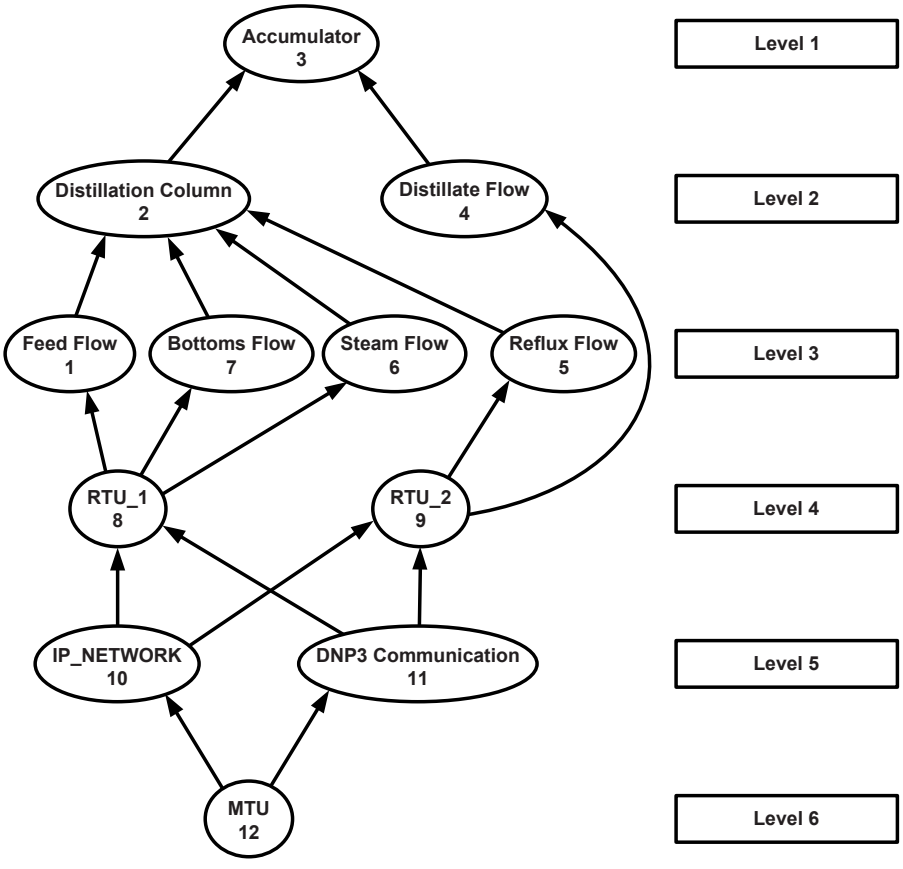

Figure 3. Level partitions for the distillation column.

partition and the separate parts partition. We define the reachability set $R(p_i)$ as the set of vertices reachable from p_i, and the antecedent set $A(p_i)$ as the set of vertices that reach p_i. Then, the level partition $L(P)$ is defined as $L(P) = [L_1, L_2, \ldots, L_l]$ where l is the number of levels. If the 0^{th} level is defined as the empty set, the level partitions of P, $L_0 = \oslash$, can be found iteratively as follows:

$$L_j = \{p_i \in P - L_0 - L_1 - \cdots - L_{j-1} \mid R_{j-1}(p_i) = R_{j-1}(p_i) \cap A_{j-1}(p_i)\}$$

where $j = 1..l$ and $i = 1..n$.

The levels have three properties: (i) $\cup L_i = V$ for $i = 1..l$, (ii) $L_i \cap L_j = \oslash$ for $i \neq j$, and (iii) edges leaving vertices in level L_i can only go to vertices in levels L_j such that $i \leq j$. Therefore, there are six levels for the distillation column example and the corresponding level partitions are: $L = (3), (2, 4), (1, 5, 6, 7), (8, 9), (10, 11), (12)$ (Figure 3).

It is possible that some of the components of P constitute a smaller digraph that is separate (i.e., disjoint) from the remainder of the digraph. The separate parts partition is used to identify any disjoint parts of the process control system. A set of bottom-level components is required to define the separate parts partition.

B is a set of bottom-level components if and only if for all $p_i \in B$, $A(p_i) = R(p_i) \cap A(p_i)$. Given the reachability matrix P for a process control system, a separate parts partition $S(P)$ is defined as $S(P) = [D_1; D_2 \ldots; D_m]$ where m is the number of disjoint digraphs in the digraph represented by A.

In order to find $S(P)$, the set of bottom-level components B is found as $B = p_i \in P | R(p_i) \cap A(p_i)$ and two elements $p_i, p_j \in B$ are placed in the same block if and only if $R(p_i) \cap R(p_j) \neq \oslash$. Once the components of B have been assigned to blocks, the remaining components of the reachability sets for each block are appended to the block.

The diagnostic process begins with the computation of the reachability matrix and the partitions described above. Let F be the set of components in which a fault has been identified. Adhering strictly to the ontological model, a component has a fault if it is in an unlawful state.

If the ontological model is used to model the system in real time, then an unlawful event indicates that there is at least one component in an unlawful state and these components would be added to F. Since some of the ontological elements have process variables as their properties, a statistical model of process variables can be used to partition $S(X)$ into lawful and unlawful state spaces $S_L(X)$ and $\overline{(S_L(X)}$, respectively.

Process variable values can be monitored and unlawful events can be easily detected, at least for components that have process variables as their properties. This is similar to the methods described in [7, 11]. The important difference is linking the information to the ontological model, which includes elements that are not in a purely statistical process model. Using this approach, it is not possible to detect all the faulty components in the ontological model because not all components have process variables as properties. Fortunately, this does not prevent the fault source search algorithm (Algorithm 1) from identifying these components as potential fault sources. Approaches for detecting faults in other components are discussed later in this paper.

After the faulty components have been identified, the fault source search algorithm can be used to identify candidate fault sources to be sequentially tested by process operators. The algorithm uses the directed graph G, level partition of G and separate parts partition of G to find the common ancestors of abnormal components and checks the ancestors that are farthest upstream first.

The algorithm starts with all potential error sources. These potential fault sources are the ancestors of the set of components F observed to have failed. According to Step 2, the algorithm terminates when the number of fault source candidates has been reduced to one. If there is more than one ancestor for all the observed faulty components, the testing begins with the ancestor(s)

Algorithm 1 : Fault Source Search Algorithm (adapted from [9]).

Compute the reachability matrix P of G.
Compute the level partition of G.
Compute the separate parts partition of G.

1. Find the set of potential error sources Q where $Q = \cap P(x_i)$ for $x_i \in F$.
2. If $|Q| = 1$, then return Q as the error source.
3. If $|Q| = 0$ or the number of disjoint graphs > 1, multiple sources of error exist. Apply the algorithm to each disjoint digraph that contains at least one observed error.
4. Let $\Sigma = \{v | v \in Q \wedge level(v) = min\{level(v_i)\ \forall v_i \in Q\}\}$.
5. If $|\Sigma| = 1$ and $A(v) = \oslash$ for $v \in \Sigma$, then $Q = \Sigma$ and return Q.
6. If $|\Sigma| = 1$, then $\Sigma = \{immediate\ ancestor(v) | v \in \Sigma\}$ and $Q = Q - \Sigma$; Go to Step 3.
7. Select a node $q \in \Sigma$ and test the component represented by q.
8. If q is normal, then $\Sigma = \Sigma - \{q\}$ and $F = F - \{q\}$.
9. $F = F \cup \{q\}$; Go to Step 1.

closest to the observed faulty components or ancestors with the lowest level in the level partition (Step 4). The algorithm also terminates if it reaches a component without any ancestor; in this case, the component is returned as the fault source (Step 5). Steps 6 through 9 attempt to reduce the candidate set Q. Step 7 could be improved by using a heuristic method to decide which component should be tested next.

Past experience often plays a role in determining the likely paths along which faults propagate. This could be included in the model by defining error propagation probabilities for each edge in G. For any component X, an error may have propagated to the component from any of the upstream components (ancestors of X). In some cases, there may be several immediate ancestors of X and the fault could have propagated from any of these ancestors or any of the error propagation paths headed by these ancestors. Obviously, for a given component, a fault is more likely to propagate from some of its ancestors instead of other components. This information can be captured using a propagation probability and added to the digraph G by associating with each edge the probability p_{ij} that a fault will propagate from X_i to X_j. The probability values may be obtained from experienced operators. If the information is not available, equal probability values may be assigned to each p_{ij} and the probability for p_{ij} can be gradually acquired through use of the system [9]. For each element X_j, p_{ij} is defined such that $\sum_{i=1}^{2} p_{ij} = 1$ where n is the in-degree of the vertex corresponding to the component X_j. For example, Figure 4 shows a simple node X_j that has three nodes that could propagate an error to it. The propagation probabilities are used in Step 7 to determine which component to test.

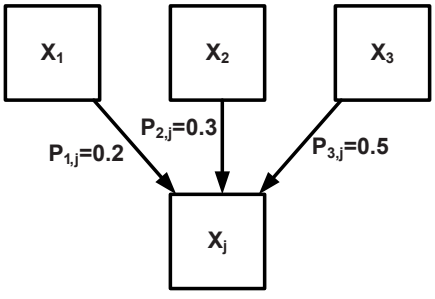

Figure 4. Fault propagation probability.

The benefits of combining the ontological model with the fault diagnosis algorithm can be clarified using a simple example. Consider a situation where the distillate flow valve is stuck. Process monitoring combined with the ontological model identifies two nodes (Accumulator and Distillate Flow) as entering unlawful states. The fault diagnosis algorithm would identify Distillate Flow as the first component to test, followed by Accumulator.

Now consider a more challenging situation. A hacker penetrates the corporate network and discovers a path through the control gateway that enables him to inject SCADA traffic into the control network. The hacker has no knowledge of the distillation column system, but is able to discover that DNP3 traffic is flowing in the network. To disrupt the process, the hacker injects DNP3 traffic into the control network using different link layer addresses, random sequence numbers and the Direct Operate function code to set Analog Outputs 1, 2 and 3 to their largest possible values (100%). As a result, the reflux flow valve periodically opens to 100% before returning to the target valve position setting, and the steam flow valve periodically opens to 100% and then returns to the target steam valve position setting. This causes the steam flow and reflux flow to become elevated and the bottom temperature to increase, all of which could cause the distillation column to flood. The process monitoring system indicates that Elements 2, 6 and 5 enter unlawful states. The fault diagnosis algorithm identifies the Elements 10 and 11 (IP_Network and DNP3) as the elements to test first. While these elements may not be testable in the same way as a flow valve, the algorithm alerts operators that the process faults may be (in this case are) induced by cyber intrusions, enabling them to take appropriate actions.

5. Conclusions

Plant operators need sophisticated models for understanding and reasoning about possible cyber intrusions in process control systems. The ontological model described in this paper permits formal and explicit representations of process control systems, including elements of the process being controlled and the process control system itself. The benefits of the ontological model are made apparent by the fault diagnosis algorithm developed using the model. In

particular, the fault diagnosis algorithm leverages the ontological model to fuse cyber security relevant components with traditional process control elements to provide plant operators with valuable synthesized information related to process operation and potential cyber intrusions.

Because faults are manifested by anomalous behavior, the fault diagnosis algorithm can be viewed as an anomaly-based intrusion detection system. However, instead of looking for traditional IT anomalies, this intrusion detection system identifies process anomalies and maps them to a traditional fault source or a control system element subject to cyber intrusions.

Several avenues exist for further research. Fault detection is currently limited to process variables. However, the algorithm can be extended because the ontological model is broad enough to include elements that do not have process variables as properties. Another research task is to explore the possibility of extending process monitoring to process control monitoring by adding inputs from traditional IT security appliances such as firewalls and network intrusion detection systems. The ontological model could then be used to aggregate and interpret the heterogeneous information, all of which is relevant to the security (and safety) of a process control system. Finally, it would be interesting to explore the use of forward propagation instead of backward propagation to identify the impact of a cyber intrusion on a specific process control element such as an RTU. This information would be invaluable to risk assessment and risk management efforts.

References

[1] M. Brandle and M. Naedele, Security for process control systems: An overview, *IEEE Security and Privacy*, vol. 6(6), pp. 24–29, 2008.

[2] M. Bunge, *Ontology I: The Furniture of the World; Treatise on Basic Philosophy (Volume 3)*, Reidel, Boston, Massachusetts, 1977.

[3] A. Burton-Jones and P. Meso, Conceptualizing systems for understanding: An empirical test of decomposition principles in object-oriented analysis, *Information Systems Research*, vol. 17(1), pp. 38–60, 2006.

[4] R. Esposito, Hackers penetrate water system computers, ABC News, New York (blogs.abcnews.com/theblotter/2006/10/hackers_penetra.html), October 30, 2006.

[5] J. Evermann and Y. Wand, Toward formalizing domain modeling semantics in language syntax, *IEEE Transactions on Software Engineering*, vol. 31(1), pp. 21–37, 2005.

[6] M. Fickes, Cyber terror, *Government Security*, July 1, 2008.

[7] J. Graham and P. Ralston, Intelligent computer-based monitoring and fault isolation for industrial processes, *International Journal of Computers and Their Applications*, vol. 9(3), pp. 147–157, 2002.

[8] T. Gruber, A translation approach to portable ontology specifications, *Knowledge Acquisition*, vol. 5(2), pp. 199–220, 1993.

[9] J. Guan and J. Graham, Diagnostic reasoning with fault propagation digraph and sequential testing, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 24(10), pp. 1552–1558, 1994.

[10] V. Igure, S. Laughter and R. Williams, Security issues in SCADA networks, *Computers and Security*, vol. 25(7), pp. 498–506, 2006.

[11] R. Isermann, Supervision, fault-detection and fault-diagnosis methods – An introduction, *Control Engineering Practice*, vol. 5(5), pp. 639–652, 1997.

[12] M. Naedele and O. Biderbost, Human-assisted intrusion detection for process control systems, *Proceedings of the Second International Conference on Applied Cryptography and Network Security*, pp. 216–225, 2004.

[13] N. Narayanan and N. Viswanadham, A methodology for knowledge acquisition and reasoning in failure analysis of systems, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 17(2), pp. 274–288, 1987.

[14] A. Opdahl and B. Henderson-Sellers, Ontological evaluation of the UML using the Bunge-Wand-Weber model, *Software and Systems Modeling*, vol. 1(1), pp. 43–67, 2002.

[15] Y. Wand and R. Weber, An ontological model of an information system, *IEEE Transactions on Software Engineering*, vol. 16(11), pp. 1282–1292, 1990.

[16] Y. Wand and R. Weber, Research commentary: Information systems and conceptual modeling – A research agenda, *Information Systems Research*, vol. 13(4), pp. 363–376, 2002.