MODELING CYBER EFFECTS IN CYBER-PHYSICAL SYSTEMS WITH DEVS

Suresh K. Damodaran
The MITRE Corporation
sdamodaran@mitre.org

Saurabh Mittal
The MITRE Corporation
smittal@mitre.org

ABSTRACT

Increased interconnectivity of Cyber-Physical Systems, by design or otherwise, increases the cyber attack surface and attack vectors. It is not always desirable to do cyber risk assessment of such interconnected systems by cyber attacks on the entire system. This paper presents a formal approach for simulating cyber effects for cyber risk assessment of subcomponents of such systems using a control component based on Discrete EEvent System (DEVS) models. Our approach first separates components of a system that may be directly attacked, from components of the system whose behavior need to be studied for the impact originating from those attacks. The cyber attack effects are simulated through interception of communications among the separated components. We define the types of cyber effects, and show an example of how an attack may be simulated with a control component through a case study.

Keywords: DEVS, CPS, cyber-physical systems, model based design, cyber attacks, cyber effects.

1 INTRODUCTION

Cyber-Physical Systems can range from industrial control system (ICS) to Internet of Things (IoT) systems, and encompass a wide variety of protocols, buses, and networks. While the definition of a Cyber-Physical System (CPS) is still evolving, we assume a CPS consists of interacting networks of physical devices and computational components that may be remotely controlled (Lee 2008). While an earlier CPS may have been designed as a stand-alone and isolated system, a modern CPS is designed with connectivity assumptions. Due to intentional or unintentional network connectivity through the Internet or other means, no CPS can be assumed to be isolated. The consequence of the increased interconnectivity among the systems is the addition of new cyber attack surfaces, and vulnerabilities exploitable with new or existing attack vectors. Risk posed by such vulnerabilities (Axelrod 2013) need to be assessed. Due to concerns for their safety, and fears of malware contamination, conducting unrestricted cyber risk assessment by directly attacking an entire CPS is not desirable, especially if it is hard to acquire, or is expensive. As demonstrated by the COATS project (Morse et al. 2014) for cyber operational training, the effects of attacks may be communicated directly among the physical or cyber components without actually attacking cyber components. This approach is helpful in addressing safety and contamination concerns. Another possible solution to address these concerns is to use a modeling and simulation-based approach. Such an approach isolates specific CPS components in a modeling environment. The selected components can be studied in two modes: either attacked or are affected by the attack (directly or indirectly).

Our main contribution in this paper is a formal foundation for applying cyber attack impact to models of CPS components. The attack is simulated using control components. The control component based approach allows for studying CPS behavior by starting with models, and progressively replacing the models with physical components after evaluating the model behaviors using simulation. This approach also allows
studying CPS behavior starting with acquiring a CPS, breaking it into physical components, replacing some components with Live, Virtual, and Constructive (LVC) models (Henninger et al. 2008) out of concern for safety or malware contamination, and then studying CPS properties through the insertion of a control component. We utilize the Discrete EEvent Systems (DEVS) theory (Zeigler et al. 2000) for its formal semantics. DEVS approach is shown to be capable of modeling emergent behavior in complex systems (Mittal 2013) and also applicable in the real-time design of CPS components through co-simulation (Mittal et al. 2015, Risco-Martín et al. 2016). Another motivation for considering DEVS formalism in a CPS is its applicability to both continuous and discrete event hybrid systems (Zeigler et al. 2000).

1.1 Related Works

Model-integrated (actor based) approaches such as (Karsai et al. 2014) and model-based approaches such as (Derler et al. 2012) and (Henriksson and Elmqvist 2011) are some of the approaches to study CPS properties. CPS are traditionally modeled for evaluating their functional features, and are widely known to be quite challenging to model (Derler et al. 2012, Karsai et al. 2014, Espinoza et al. 2009). Cyber LVC models are used in cyber testing or exercise events in cyber ranges (Damodaran and Couretas 2015). Cyber effects on missions are studied through simulations by works such as (Musman et al. 2011). All of these efforts assume that all components of a system may be attacked, and do not permit distinguishing between a CPS component that may be attacked directly and a CPS component that is so valuable that only impact of the attack on it should be studied. FDEVS is a DEVS-based formalism for modeling behavioral and structural faults (Kofman et al. 2000). However, the separation of attacked and affected is needed for a controlled study. The cyber effects we describe in this paper may be induced by faults maliciously injected. The focus of this paper differs from FDEVS in that we are using the fundamental DEVS formalism to incorporate a control component that can be used to inject effects to a subset of the components for controlled studies.

In the next section, we provide the basics of DEVS theory. Section 3 describes how to insert a control component into an existing model of a system. Section 4 describes key cyber impact modeling concepts of signal, effect, and control command. Section 5 uses an elevator system to illustrate the application of the techniques described in earlier sections, and Section 6 concludes this paper.

2 DEVS THEORY BASICS

DEVS formalism is made up of two orthogonal concepts: Hierarchy of system specifications that describe 5 levels of system behavior, and System specification formalisms that incorporate various modeling styles, such as continuous or discrete. For an elaborate discussion on these two concepts, see (Zeigler et al. 2000). Description of alternate system specification formalisms such as Differential Equation System Specification (DESS), Discrete Time System Specifications (DTSS), and Quantized systems are briefly described in (Mittal and Martin 2013) Chapter 3 and can be found in much greater detail in (Zeigler et al. 2000). While DESS and DTSS, as their names suggest, are self-explanatory, Quantized DEVS is specified through a quantization process. Quantized DEVS is used for representing and simulating continuous systems as an alternative to the more conventional time axis. CPS may comprise of various physical components (e.g. sensors and actuators) that have continuous nature that require quantization. While discretization leads to DTSS, quantization leads to discrete event systems. The universality of DEVS allows specification of hybrid systems.

DEVS models are of two types: atomic and coupled. When the system cannot be decomposed any further, it’s behavior is specified through an atomic DEVS model. A coupled model is a hierarchical model that comprises both of atomic and coupled models.

An Atomic DEVS (Zeigler et al. 2000) is specified as:
Definition 1. \( \text{DEV} \) S

Definition 2. \( \text{DEV} \) S

A distributed system may have concurrent events in multiple components at a given time instant which may result in collisions in simulations. In order to model distributed systems, the chosen formal representation must account for modeling these collisions and a way to resolve the order of these simultaneous events both for atomic and coupled models. The classical DEVS formalism (1976) was a sequential DEVS and did not account for the confluence of events. The parallel-DEVS (pDEVS) formalism has a confluent transition function \( \delta_{con} \) in atomic DEVS to account for such collisions. A coupled model is used to define a distributed system consisting of multiple atomic and coupled models that need to be wired together through their input and output ports. A DEVS coupled model is specified as in Definition 2:

Definition 2. \( \text{DEV} \) S coupled

Unless otherwise stated, whenever we use the term \( \text{DEV} \) S in this paper, we imply pDEVS described by Definitions 1 and 2. An important concept in \( \text{DEV} \) S systems is the closure under coupling principle. This principle is derived from the closure under composition principle in General Systems Theory. In DEVS theory, this principle states that any atomic model model can be decomposed into a coupled model and vice versa. This also implies that any DEVS representation of a discrete event system is a unique representation (Zeigler et al. 2000). While there can be many ways to engineer decomposition of a, the closure under coupling principle highlights the gaps in the manifested behavior that may result from incomplete specifications or restricts the behavior that may result from over-specification of the decomposed model. However, sometimes over-specification is needed to get to the required level of fidelity. Given that the coupled and atomic models are both DEVS models, the behavior manifestation is always unique and reproducible despite multiple instances of the decomposed model. The DEVS formalism allows us to decompose the system in various sub-systems and also allows us to reduce the multi-component sub-system into a single system while preserving the behavior. When a given coupled DEVS model (i.e. Definition 2) is represented as an atomic DEVS model (i.e. Definition 1), the resulting model is called a resultant model (Zeigler et al. 2000). We define an operator \( \mathbb{R} \) that generates the resultant model (Definition 3). To complement, we also define a decomposition operator, \( \mathbb{D} \), that will generate a decomposed DEVS coupled model, \( N' \), from an atomic model \( A \) (Definition 4). The next section provides a summary of our modeling approach.
Definition 3. $\mathbb{R}(N) \rightarrow A$ is a resultant operator, where $N$ is a $\text{DEVS}_{\text{coupled}}$ model (using Definition 2) and $A$ is the resultant $\text{DEVS}_{\text{atomic}}$ model, and

- $S = \times_{d \in D} Q_d$ is the state space,
- $ta(s) = \min\{\sigma_d | d \in D\}$, where $s \in S$ and $\sigma_d = ta(s_d) - e_d$.

For detailed description of $\delta_{\text{int}}$, $\delta_{\text{ext}}$, $\delta_{\text{con}}$ and $\lambda(s)$, refer (Zeigler et al. 2000).

Definition 4. $\mathbb{D}(A) \rightarrow N$ is a decomposition operator, where $N$ is a $\text{DEVS}_{\text{coupled}}$ model with exactly the same input $X$ and output $Y$ as, $A$, a $\text{DEVS}_{\text{atomic}}$ model, thus preserving the I/O function.

2.1 Modeling Approach

An atomic DEVS can specify the behavior of a CPS "as a whole" (Definition 1). Alternatively, a coupled CPS can be reduced to an atomic DEVS model using the resultant operator (Definition 3). Our overall approach for modeling cyber impact is altering the behavior of the CPS through insertion of a control component that can interact with various sub-components of the CPS. These sub-components can comprise of both atomic and resultant systems, in case of complex hierarchical system-of-systems (SoS) whose detailed models are not available. This inserted control component can alter information flowing through it, or output new information that mimics cyber attacks to the components of a CPS. The overall process is summarized in Figure 1. Let $A^o$ be the original CPS. We decompose $A^o$ and specify it as a coupled model $N'$, per Definition 2, through the decomposition operator (Definition 4). This decomposition is informed by the current architecture of CPS and the information available through various subject matter experts (SMEs). A simulation of $N'$ must yield the same behavior as $A^o$ per closure under coupling principle. This aspect will become more significant when some of the model sub-components are replaced by actual hardware for advanced testing and evaluation. A control component, $C$, is inserted into $N'$ so that some of the communication among components in $N'$ are routed through $C$, to yield $N$. Theoretically, there exists an $A$, shown in Figure 1, that is a resultant of $N$. After addition of $C$ to $N'$, the behavior of $N'$ can be further enhanced through the addition of signals, effects, and control commands, yielding $N_E$. 
3 INJECTING CONTROL COMPONENT INTO AN EXISTING CPS MODEL

In this section, first, we explain the concept of decomposition with an example. Then, we introduce the concept of a control component, and describe how to insert a control component into a coupled DEVS model. Let $A'$ be an atomic model of the original CPS that is going to be modeled and simulated (Figure 2). By convention, the inputs are shown on the left and outputs are shown on the right in Figure 2. Note that if the original system can be only represented by a coupled DEVS model instead of an atomic model, we can apply the resultant operator (Definition 3) to it to create an equivalent atomic model. On the other hand, if the original system is already a coupled DEVS model one may proceed to insert a control component in that model as described in Section 3.1. We now specify $A'$ as a $DEVS_{atomic}$ (Definition 1) in equation below.

$$A' = \langle X', Y', S', \lambda', \delta_{int}', \delta_{ext}', \delta_{con}', ta' \rangle$$

Now, we obtain a $DEVS_{coupled}$ (Definition 2) for $A'$ by applying the decomposition operator (Definition 4). The decomposed model contains components referred to as functional components.

$$\mathbb{D}(A') \rightarrow N' = \langle X', Y', F, \{M_f\}, \{I_f\}, \{Z_{i,f}\} \rangle$$

In Equation 2, for each $f \in F$, $M_f$ is a DEVS model and $I_f$ and $Z_{i,f}$ (Definition 2) hold their respective meanings.

Now, let us specify $A'$, the equivalent atomic model to $N'$ using the resultant operator (Definition 3), as below.

$$\mathbb{R}(N') \rightarrow A' = \langle X', Y', S', \lambda', \delta_{int}', \delta_{ext}', \delta_{con}', ta' \rangle$$

Theorem 1 below establishes how the inputs and outputs of $A'$ are related to $A'$.

Theorem 1. Every $A' \in \{\mathbb{R}(\mathbb{D}(A'))\}$ will have the same inputs, and outputs as $A'$.

Proof. By Definition 4, $\mathbb{D}(A')$ yields a coupled DEVS model, $N'$, (Equation 2) which has the same inputs and outputs as $A'$. Zeigler et al. have shown that $N'$ exhibits closure under coupling by constructing and showing the resultant of a coupled DEVS model is also a well defined DEVS model, with the same inputs and outputs (Zeigler et al. 2000). Therefore, $\mathbb{R}(N')$, which yields $A'$, will have the same inputs and outputs as $N'$, implying, all models in $\{\mathbb{R}(\mathbb{D}(A'))\}$ will have the same inputs and outputs.

It is also possible that the decomposition operation, yielding $N'$ will result in addition of more states (that may be only transitory and not impact the overall behavior $A'$) and there may be many ways a decomposition can be done. However, at the external interface level, the behavior is same.

Example 1. The atomic model of a sample CPS is shown in Figure 2. Figure 3 shows a decomposed version $A'$ obtained in Equation 2. $N'$ contains five components: $F_1, F_2, F_3, F_4$ and $F_5$. These components represent different constituents of a CPS. $F_1$ is the cyber component of interest that is subject to cyber attacks, $F_2$ is the physical component whose behavior changes due to the attacks on $F_1$, while $F_3, F_4$, and $F_5$ are other system components that get influenced by $F_1$ or $F_2$. Thus, the coupled model $N'$ (Definition 2) is represented with: $X' = \{< x_1, p_1 >, < x_2, p_2 >\}$, $Y' = \{< y_1, p_3 >, < y_2, p_4 >\}$ with five components, $F_1$, $F_2$, $F_3$, $F_4$ and $F_5$, and each component has a corresponding atomic model $M_1$, $M_2$, $M_3$, $M_4$ and $M_5$ respectively. $I_1 = \{N', F_2\}$, $I_2 = \{N', F_1, F_4, F_5\}$, $I_3 = \{F_1\}$, $I_4 = \{F_2\}$ and $I_5 = \{F_3\}$. Example elements in $Z_{i,f}$ categorized as $EIC$, $EOC$, and $IC$, are:

$EIC$: $\{(< x_1, p_1 >, < x_1, p_5 >), (< x_2, p_2 >, < x_2, p_{15} >)\}$
3.1 Control Component Insertion

Some CPS consist of expensive physical components that may get damaged under uncontrolled cyber attacks. To protect the system under test (SUT) that includes the expensive components from the direct impact of cyber attacks, we need to exert control on what impacts (of cyber attacks) get transmitted to the SUT. To implement this control, we need to separate the components that are subjected to cyber attacks from the components whose behaviors we want to study as a result of the attacks.

**Definition 5.** For a coupled DEVS model with a component set \(F\), for the purpose of cyber risk assessment, we identify \(F^\delta \subseteq F\), and \(F^i \subseteq F\), such that \(F^\delta \neq \emptyset\), \(F^i \neq \emptyset\), \(F^\delta \cap F^i = \emptyset\). We also define a control component, \(C\), as a DEVS\textsubscript{atomic} model.

\(F^\delta\) consists of components that may be attacked, and \(F^i\) consists of components whose behavior need to be studied as a result of the attacks to components in \(F^\delta\). The definitions of \(F^\delta\) and \(F^i\) are dependent on test scenarios in the sense that while in one scenario, some attacks may be conducted in \(F^i\) and their effects are studied on \(F^\delta\), in another scenario, the roles of components in \(F^\delta\) and \(F^i\) may be reversed. We augment \(N'\) with \(C\) to interrupt all interactions between components in \(F^\delta\), and components in \(F^i\) so that \(C\) can control the behavior of components in \(N'\). The control component may have interfaces with other components in \(N'\), and can also have additional inputs and outputs than in \(A'\). \(C\) may also be a resultant (Definition 3) of a coupled DEVS model.

**Example 2.** We define \(F^\delta\) and \(F^i\) in the DEVS model shown in Figure 3. \(F^\delta = \{F1\}, F^i = \{F2\}\). Now, \(F-F^\delta-F^i = \{F3,F4,F5\}, F'_2 = \{N',F1,F4,F5\}\), and \(T'_2 = \{N',F1,F3,F4,F5\}\).

Let \(N\) be composed with \(C\) and \(N'\), formally specified as below:

\[
\text{N = \langle X,Y,D,\{M_d\},\{I_d\},\{Z_{i,d}\}\rangle}
\] (4)

In Equation 4, \(X = X^o \cup X_c\), where \(X_c\) is a set of input events to \(C\), \(Y = Y^o \cup Y_c\), where \(Y_c\) is a set of output events from \(C\), \(D = C \cup F\), and \(F\) is the set of component models in \(N'\). For each \(d \in D\), \(M_d\) is a DEVS model. For each \(d \in F \cup C \cup \{N\}\), \(I_d\) is the influencer set of \(d\), \(d \notin I_d\). \(T'_d\) is the transitive closure of influencers of a component \(d\) in \(N'\). For each \(i \in I_d\), \(Z_{i,d}\) is the i-to-d coupling with:

(a) \(Z_{i,d}\) is the same as in \(N'\) (Equation 2), except if:

\[
((i \in F^\delta) \land (d \in F^i)) \lor (\exists j \in F^d \mid i \in F^\delta \land d \in (T'_d - F^\delta - F^i))
\]

or

\[
((i \in F^i) \land (d \in F^d)) \lor (\exists j \in F^d \mid i \in F^i \land d \in (T'_d - F^\delta - F^i))
\]

(b) \(Z_{i,d}: Y_i \rightarrow X_d\), if \(i \in (F^\delta \cup F^i), d = C\)

(c) \(Z_{i,d}: Y_i \rightarrow X_d\), if \(i = C, d \in (F^\delta \cup F^i)\)

(d) \(Z_{i,d}: Y_i \rightarrow X_d\), if \(\exists j \in (F^i \cup F^d)\mid i \in (T'_d - F^\delta - F^i), d = C\)

(e) \(Z_{i,d}: Y_i \rightarrow X_d\), if \(i = C\), and \(\exists j \in (F^i \cup F^d)\mid i \in (T'_d - F^\delta - F^i)\)

(f) \(Z_{i,d}: Y_i \rightarrow X_d\), if \(i = N\) and \(d = C\)

(g) \(Z_{i,d}: Y_i \rightarrow Y_d\), if \(i = C\) and \(d = N\).

**Example 3.** In Figure 4, a new control component, \(C\) is inserted in CPS model \(N'\), yielding \(N\). Equation 4, Case (a) includes all couplings from \(N'\) into \(N\), including external input and output, except those that result
in any component in \( F^s \) influencing any component in \( F^t \) or vice versa, directly or transitively. The direct influencing couplings are disconnected in Figure 3 by Cut 1 of green and red connections, and corresponds to the \((i \in F^s) \land (d \in F^t)\) and \((i \in F^t) \land (d \in F^s)\) clauses in Case (a). Cut 2 of orange connection, corresponds to \((\exists j \in F^t | i \in F^s \land d \in (T^N \setminus F^s + F^t))\) in Case (a). In Figure 4, the red and green couplings among \( F_1 \) and \( F_2 \) through \( C \) correspond to Cases (b) and (c), respectively, while Case (d) corresponds to orange coupling from \( F_1 \) to \( C \), and Case (e) corresponds to orange coupling from \( C \) to \( F_3 \). Note that in Cases (d) and (e) couplings to and from \( C \) are created to intercept all communications from any component in \( F^s \) to any component in \( F \setminus F^s + F^t \) that could transitively influence \( F^s \), and all communications from any component in \( F^t \) to any component in \( F \setminus F^s + F^t \) that could transitively influence \( F^s \).

**Theorem 2.** \( N \) is a DEVS model.

**Proof.** \( N' \) is a DEVS model by Definition 4 and Equation 2. \( C \) is a DEVS model by Definition 5. The composition of \( C \) and \( N' \) is specified conforming to a coupled DEVS model in Equation 4. Therefore, \( N \) is a DEVS model based on the closure under coupling principle of DEVS.

It is easy to observe that the excluded couplings in Case (a) in Equation 4 are replaced by new pairs of couplings, one from a component to \( C \), and another from \( C \) to a component in Cases (b), (c), (d), and (e). Therefore, \( N \) also preserves the coupling relationships in \( N' \) via \( C \). Finally, because \( N \) is a DEVS model by Theorem 2, it can be represented as an atomic model \( A \) (Figure 1) by the application of the resultant operator to \( N \) as below.

\[
\mathbb{R}(N) \rightarrow A = \langle X, Y, S, \lambda, \delta_{int}, \delta_{ext}, \delta_{con}, ta \rangle \tag{5}
\]

We may control the outputs from components in \( F^s \) that go to \( F^t \) (and vice versa) while \( F^s \) is under attack so that the components in \( F^t \) can be studied for the effects of attacks on \( F^s \). \( C \) can also have additional input and output than in \( A' \). \( F^s \) and \( F^t \) may contain only cyber components or combinations of cyber and physical components.

### 4 CONTROLLING IMPACT OF CYBER ATTACK

In the previous section, we introduced the idea of a control component, and showed how a control component can be inserted into a coupled DEVS model in order to ensure all communications among components in \( F^s \) and components in \( F^t \) go through \( C \). Observe that we may use \( C \) also to simulate the impact of the attacks in \( F^s \) simply by injecting inputs into \( F^t \) and modifying the states and state transitions of components in \( F^t \). Such an approach is useful in studying the potential impact of cyber attacks using models of \( F^t \) without subjecting \( F^s \) to cyber attacks. The selection of the constituents of \( F^s \) and \( F^t \) is dependent on test scenarios and may not be automated. In this section we define the concept of *effects*, and show how to design *effects* and use them for simulating cyber attack impact.

#### 4.1 Effect

The state of a system is expressed as a multivariable wherein each constituent variable has a set of potential values. The cartesian product of all of these possible values of these state variables define the potential states of a system. In any implementation of a system, the system state variables may not assume all these possible values. Indeed, only a small subset of this potential state space would have been considered by the designers of a system. A cyber attack may put a system into any of these potential states, some of which may not be present in a given implementation of a system under normal operations or a model of the system as shown in Figure 5. For analysis purposes, we choose such a subset of the potential state space, and call that the set.
of effects. The effects that must occur in normal operation for the correct operation of a system or model are named normal effects, and those that must not occur in correct operation of a system are named abnormal effects. Let us define these concepts formally below.

**Definition 6.** A normal effect, $e_{nm} \in E_{nm}$, is a state such that $e_{nm} \in S^0 \cup S'$, where $S^0$ is the state space of $A^0$ and $S'$ is the state space of $A'$ such that $R(D(A^0)) \rightarrow A'$.

**Definition 7.** A abnormal effect, $e_{ab} \in E_{ab}$, is a state such that $e_{ab} \notin S^0 \cup S'$ under normal operations, where $S^0$ is the state space of $A^0$ and $S'$ is the state space of $A'$ such that $R(D(A^0)) \rightarrow A'$.

A subset of normal effects may be observed for analysis or validation under both normal and correct operations. Such effects, when expressed through the constituent state variables, define the properties of a system that can be observed through instrumentation.

### 4.2 Modeling Effects of Cyber Attacks

As a result of a cyber attack a system may continue to exhibit a subset of normal effects, exhibit a new set of abnormal effects, or exhibit normal effects albeit in a different context. In this section, we define signals, effects injection operators, and control commands to model such cyber attack impacts. When a control component is inserted into a coupled model, $N'$ (Equation 2), new internal couplings and corresponding transformation functions are defined among the control component, $C$, and the functional components, $F$, corresponding to the internal couplings and transformations that existed prior to the insertion of the control component. For every coupling severed between any two $f \in F$ in $N'$, two couplings are added via the control component $C$ in $N$. These new couplings may be used by $C$ to manipulate the contents of the communication with the functional components to mimic the effects of cyber attacks. Two types of information are transmitted through these couplings: signals and control commands.

**Definition 8.** A signal is any designated input from a control component $C$ in $N$ to a functional component, $f \in F$, that is intended to trigger a normal or abnormal effect in $N$.

As an example, if $C$ does not alter any content of communication between $F_1$ and $F_2$ (Figure 4), $C$ will behave as a faithful relay. Any input from $C$ to $F_2$ or $F_3$ that is intended to trigger a behavior change in $F_2$ is named a signal.

Effects of cyber attacks may be modeled in four ways: changing the range of input values to a model component, putting the system into another state that triggers the associated transitions from that state, adding one or more new states that do not exist in normal operations with the associated new transitions and
Table 1: Effects Injection operator cases.

<table>
<thead>
<tr>
<th>Case</th>
<th>Input</th>
<th>State</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>I₁</td>
<td>x → x* ∈ X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I₂</td>
<td>x ∈ X</td>
<td>-</td>
<td>Add</td>
</tr>
<tr>
<td>I₃</td>
<td>x ∈ X</td>
<td>Add</td>
<td>Add</td>
</tr>
<tr>
<td>I₄</td>
<td>u /∈ X</td>
<td>Add</td>
<td>Add</td>
</tr>
</tbody>
</table>

Definition 9. An effects injection operator, I, may be defined to alter the behavior of a functional component, f ∈ N', that receives input from a control component, C in N, through the following means. Below, x ∈ X is the input to f.

I₁) Transform x to another input, x* ∈ X.
I₂) Add new internal or external state transitions in f, and new transitions upon receipt of a signal that is an existing input in X.
I₃) Add new states in f, and new transitions involving these states upon receipt of an existing input in X.
I₄) Add new states in f, and new transitions involving these states upon receipt of a new signal, u.

Table 1 captures these categories of injection operators succinctly. By application of the injection operators belonging to different categories, a model may change through the addition of new states and state transitions in order to effectively mimic cyber attack effects. The addition of new behavior as a result of addition of new states and transitions may make the model N illegitimate, for example, it may introduce the Halting problem. To avoid this condition, we have to ensure that N is a legitimate DEVS model. From Section 6.2.3 (Zeigler et al. 2000), Theorem 2, we have that application of the injection operators and subsequent triggering of those operators in N results in a legitimate DEVS model under the following mandatory conditions: (a) Every cycle in the state diagram of δ_m for f ∈ F and C in N contains a nontransitory state i.e. a state with ta(s) > 0, (b) There is a positive lower bound on the time advances for any new states that are added in f ∈ F, and (c) C is a legitimate DEVS model. The injection operators defined in Definition 9, are activated or deactivated dynamically with the use of a control command, defined below.

Definition 10. A control command specifies when one or more effects injection operator, I, must be active or inactive.

As an example, a control command may specify "activate effect injection operator 5 immediately." Note that a control command is not required to have any immediate effect in the recipient until a signal triggers the state transitions specified in the injection operator. In Section 5, we illustrate the application of signals and control commands for modeling the effects of a cyber attack.

5 CASE STUDY

We illustrate the application of concepts in previous sections through an elevator example inspired by (Hong et al. 1997). Consider the elevator system as a CPS as it comprises of physical components such as doors, motor, and elevator car interacting with processors. Our DEVS_coupled model of the elevator consists of

![Figure 5: State space before and after attack.](image-url)
elevator controller (F1), floor request processor (F2), door status processor (F3), elevator car (F4) and six floors. The model does not include the physical models of the motors or pulleys, since our focus is limited to cyber effects on the elevator system. The associated functionalities are listed in a hierarchical model (Table 2). Figure 6 shows only 2 floors for brevity. For the purpose of this case study, we assume that the attacker has access to CarCtrl (F4.2) from inside the elevator car in the elevator system.

Table 2: Elevator component-functionality map.

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>ElevatorCtrl</td>
<td>Implements the control logic for elevator movement</td>
</tr>
<tr>
<td>F2</td>
<td>RequestProc</td>
<td>Collects and forwards inputs such as destination floor requests from different floors and inside the car to F1</td>
</tr>
<tr>
<td>F3</td>
<td>DoorStatusProc</td>
<td>Provides the door status of both the floor and the car door to F1</td>
</tr>
<tr>
<td>F4</td>
<td>Car</td>
<td>This is the elevator car that moves between the floors</td>
</tr>
<tr>
<td></td>
<td>F4.1 Motor</td>
<td>A motor that activates the pulley</td>
</tr>
<tr>
<td></td>
<td>F4.2 CarCtrl</td>
<td>Controller that operates motor and car door under commands from ElevatorCtrl, and provides car position to F1</td>
</tr>
<tr>
<td></td>
<td>F4.3 CarBtn</td>
<td>Button inside the car that generates a floor request to F2</td>
</tr>
<tr>
<td></td>
<td>F4.4 CarDoor</td>
<td>Car door position sensor, sends position to F3</td>
</tr>
<tr>
<td>F5</td>
<td>Floor</td>
<td>Floors 1-6</td>
</tr>
<tr>
<td></td>
<td>F5.1 FloorBtn</td>
<td>Button on a floor for floor requests sent to F2</td>
</tr>
<tr>
<td></td>
<td>F5.2 FloorDoor</td>
<td>Floor door</td>
</tr>
</tbody>
</table>

The requests from FloorBtn (F5.1) and CarBtn (F4.3) are routed through RequestProc (F2) to ElevatorCtrl (F1) that processes the floor requests in a sequential manner. The couplings are shown in Figure 6. F1 receives the car position from CarCtrl (F4.2) as it begins to move by triggering Motor (4.1). F1 sends a stop command to F4.2 on reaching the requested floor. After F1 determines the car is stopped, it transitions to stop state, and sends open command to both CarDoor (F4.4) and FloorDoor (F5.2), and transitions to open state when the doors are open. The status of both doors are are processed by DoorStatusProc (F3) and relayed to F1, after which a new request is processed.

We show presently how a cyber attack can be simulated using a control component by using the $T_1$ operator. Here, $E_s = \{F4.2\}$, $F_i = \{F1\}$. This control component simulates the attack by sending the wrong car position from CarCtrl (F4.2) to ElevatorCtrl (F1), originally flowing through the orange coupling in Figure 6. For example, let us assume the elevator is at floor 3 when this attack was activated by a control command, and the car position was manipulated to...
stick to floor 4 by an inserted control component. Pressing car door button to floor 6 would move the car upward all the way to the topmost floor. Instead, if car button 2 was pressed, the car would go to the lowest floor. In both cases, the doors won’t open, because \( F_1 \) falsely believes the elevator car is not at the destination floor but is at floor 4. Therefore, the passengers are trapped in the elevator. As shown in Figure 6 using the same color coded couplings as in Figure 3, the cuts can be identified very clearly. Identification of cuts imply that components can be isolated and can be stimulated without any contamination of external input. Consequently, similar to Figure 4, a new control component can be injected that can intercept the information flow between \( F_s \) and \( F_t \). Therefore, communications from and to \( F_s \) can be intercepted and routed through a control component without contaminating the \( F_s \) for testing and evaluation purposes in a complex CPS. We hope to present examples corresponding to injection effects, \( I_2, I_3, \) and \( I_4 \) (Definition 9) in an extended version of this paper in the near future.

6 CONCLUSIONS

This paper presents a formal foundation for defining and applying cyber effects to models of CPS components using a control component. To our knowledge, this is the first work that formally specifies injection of cyber effects on system models by mimicking cyber attacks through a control component based approach. We first formally decompose an atomic DEVS model of a CPS. This decomposition allows us to focus on sub-components of interest, which are then modeled at both the structural and behavioral levels using the DEVS formalism. A control component is then added to the resulting DEVS model to separate the attacked and affected components within the decomposed CPS. The control component is then used to inject cyber effects, which could result from cyber attacks. We also formalized the concept of cyber effect. While it is trivial to reset the CPS model to an initial state, the recovery of a CPS system containing hardware-in-a-loop is a future area of research. We are also working on an automated algorithm for the insertion of a control component into a coupled model. Additional future work include automatic enumeration and injection of cyber effects using a control component possibly guided by FDEVS (Kofman et al. 2000) under multiple test scenarios, as well as investigation of sufficiency of the effects injection operators.

ACKNOWLEDGEMENTS

This research was funded by the MITRE Corporation’s independent research and development program. We also thank Emily Heath and anonymous reviewers for considerably improving the quality of this paper with their comments.

REFERENCES


AUTHOR BIOGRAPHIES

SURESH K. DAMODARAN is Principal Cybersecurity Architect at the MITRE Corporation, Bedford, MA, USA. He has actively contributed to security applications, standards, and research for over 15 years. He authored or co-authored 9 granted patents. He currently contributes to the Security Framework of Industrial Internet Consortium (IIC), and is a Lifetime Member of Association for Computing Machinery (ACM).

SAURABH MITTAL is Lead Systems Engineer/Scientist at the MITRE Corporation, McLean, VA USA. He is also affiliated with Society of Computer Simulation (SCS) International; and Enterprise Architecture Body of Knowledge (EABOK) Consortium. He can be reached at smittal@mitre.org.