Safety Analysis of Computer-Controlled Real-Time Systems with Message Loss Using Communicating DEVS Models

Hae Sang Song[1] and Tag Gon Kim[2]

[1] Computer Engineering Dept., Seowon University,
337-3 Musimseoro, Heungduk-gu,
Cheongju 361-742, Korea
hssong@seowon.ac.kr
[2] Electrical Engineering Dept., Korea Advanced Institute of Science and Technology,
1291 Daehak-ro(373-1 Guseong-dong), Yuseong-gu,
Daejeon 305-701, Korea
tgkim@ee.kaist.ac.kr

Abstract. The Communication DEVS formalism is an analysis means for discrete event systems modeled by DEVS formalism which has been widely used as a system theoretical specification. This paper proposes a new method for analyzing safety of real-time discrete event systems using communicating DEVS formalism. It is a part of efforts toward a unified method for modeling, simulation, and logical analysis based on the DEVS formalism and associate theory. For safety analysis of such real-time discrete event systems we first define communicating DEVS and then propose a timed reachability analysis algorithm for the models. The algorithm visits all possible timed states of the model, which is not always possible by using a simulation based state traversal. The proposed method can be well used especially for DEVS-specified systems to check various logical properties such safety, liveness and so on. A case study of a safety analysis for a rail road crossing system illustrates the usefulness of the proposed method.

Keywords: Safety Analysis, DEVS Formalism, Communicating DEVS, Real-time Systems, Reachability Analysis.

1 Introduction

Computer-based controllers have been widely used as active or passive components of real-time systems such as traffic control, aircraft, nuclear power plants, hospital patient care systems and others [1]. In those systems a problem of safety would be regarded as one of the most important factors, for the consequences of failures of such systems cause serious and vital damages to human life and property. Thus safety analysis needs to be done from the early design stage in developing such computer-controlled real-time systems. In fact, a developer of such systems would implement the design after verification of safety is completed.

T. Xiao, L. Zhang, and M. Fei (Eds.): AsiaSim 2012, Part I, CCIS 323, pp. 480–489, 2012.

Safety of such a real-time system is defined that the whole behavior of the system does not go into any bad state at which the system would cause serious consequences. Thus, safety analysis should visit all states of a system to be analyzed. Safety analysis would not be completely done by simulation of a system model. This is because simulation would not visit all possible states of the model. States to be visited in simulation are controlled by an experimental design of input stimuli which is applied to a system model.

Safety analysis is usually conducted by reachability analysis in which all direct and/or indirect paths from an initial state are generated[1][2][3][4]. In this case the state explosion problem, however, is inevitable as the number of states of components becomes large and the time is an independent variable.

Generally, a generation of all possible states in reachability analysis of a real-time system with multiple components employs state transition rules of each component. Since the real-time system has time constrained such rules should include information of sojourn time between states. Moreover, since each component communicates with each other components the generation needs state transition rules for communicating between components. Furthermore, the generation should consider a message loss in communication between components. Thus, the generation would require a formal model with sound semantics to specify both timed state transition of each component and communication between components with message losses.

There have been several approaches for safety analysis based on different modeling formalisms. Petri-net has been used for modeling and safety analysis of real-time discrete event systems for various applications in recent decades[1][4]. However, it would be difficult to implement a Petri-net model in software in terms of a set of variables with associated operations, for a Petri-net model does not specify a system in terms of variables (or states). RT-DEVS formalism is an alternative for safety analysis of such systems[2]. But RT-DEVS formalism specifies the sojourn times between states in interval. Moreover the analysis did not consider a message loss in communication between components. The timed automaton has been used in this area but it uses integer time rather than continuous time in analysis, which makes the state explosion problem worse[3].

This paper employs the Communicating Discrete Event Specification (C-DEVS) formalism[5] for safety analysis, which has a sound semantics for both timed state transition and communication between components with a message loss. Having DEVS models as components C-DEVS formalism is an analysis means by generation of all possible timed state in a global state space.

The proposed method has been applied to safety analysis for the well-known railroad gate problem. The proposed safety analysis method, though it is a yet another safety analysis method, is the first one in application of DEVS theory in safety analysis. The proposed method is different from a simulation-based approach using DEVS model in that simulation would not visit all possible states of a system model but visit only partial states set.

This paper is organized as follows. In the next section, we review the DEVS formalism first to define the Communicating Discrete Event Systems Specification (C-DEVS) formalism. Then we propose a C-DEVS model construction procedure from a set of atomic DEVS models. In section 3, we apply the C-DEVS formalism and the construction method to analyze the safeness of a railroad crossing controller problem. We conclude in section 4.

2 Background and Communicating DEVS

2.1 DEVS Formalism

As well known in the literatures [6], the DEVS formalism specifies a discrete event system in two aspects: one for the behavior of a basic component, and the other for the overall structure of a system. The former is modeled with atomic DEVS formalism, which describes the behavior of a unit component not further decomposable, which consists of three sets and four functions.

$$\text{AM} = < X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta >$$

, where

X: input event set,
Y: output event set,
S: sequential state set,
with total state set $Q = \{(s,e) \mid s \in S,\ 0 \le e \le ta(s)\ \}$,
$\delta_{ext}: Q \to Q$: external transition function, for $\delta_{ext}(s, e, x) = (s', e')$, $e < ta(s), e' = 0$.
$\delta_{int}: Q \times X \to Q$: internal transition function, for $\delta_{int}(s, e) = (s', e')$, $e = ta(s), e' = 0$.
$\lambda: Q \to Y$: output function, for $(s, e) \in Q, e = ta(s)$
$ta: S \to R_0^+$: time advance function,
R_0^+ is the non-negative real number set.

There are two types of transitions of an atomic model: 1) external transitions entailed by external events; and 2) internal transitions in the case of no event occurrence until current state sojourn time has elapsed. In the latter case, just before the internal transition, an output event is produced at the state. In an analogy to the continuous systems, external transitions would correspond to the input driven state transition and internal ones the input-free state transition.

Every state is involved in a time advance, up to which the system can remain at the state. A total state $q = (s, e) \in Q$, often we call it a 'timed state', is a pair of a state and an elapsed time at the state. As an inversion, we would define a residual total state $[\![\tilde{q} = (s, ta(s) - e) \in \tilde{Q}]\!]$ of total state q, or shortly a 'residual state', to be a pair of a state and a residual time that is left to the time advance of the total state.

The coupled DEVS formalism, not recited here, specifies the structure of discrete event systems composed of components communicating with each other through event couplings. In this paper, to focus attention on C-DEVS, we assume that the structure of coupled models of a system is collapsed into a set of flattened atomic models and that any input and/or output events with the same name are all coupled together.

2.2 C-DEVS Formalism

Analysis of a discrete event system modeled by a set of DEVS models could be executed by traversal of the state space without simulation, which generates all

possible sequences of events and/or states that can reach from a given initial state. Communicating Discrete Event Systems Specification, or C-DEVS formalism, formally defines the timed interaction mechanism between N atomic DEVS models. Unlike RT-DEVS [2] that has two types of interaction – asynchronous and synchronous, C-DEVS has three types of interaction: asynchronous, synchronous and event loss.

Consider a system model of two atomic DEVS models, $M = \{AM_i, AM_j\}$, $AM_k = < X_k, Y_k, S_k, \delta_{ext,k}, \delta_{int,k}, \lambda_k, ta_k >$ for $k = i, j$. Let the atomic models are currently at their total states $q_i = (s_i, e_i)$ and $q_j = (s_j, e_j)$ respectively. In other words we could say they are at compatible residual states $\tilde{q}_i = (s_i, \sigma_i)$ and $\tilde{q}_j = (s_j, \sigma_j)$ respectively, where $\sigma_i = ta(s_i) - e_i$ and $\sigma_j = ta(s_j) - e_j$ are the residual times. Let's denote by $(s_i, s_i') \in \delta_{int,i}$ that $\delta_{int,i}(s_i, e_i) = (s_i', 0), e_i = ta_i(s_i)$ is defined. Then C-DEVS model of the system M can be defined as follows:

$$CDEVS(M) \equiv AM_i \,||AM_j =< E, S, T, ta, M >$$

where

E: events set,

S : composed discrete states set,

T : transition relation of composed states,

ta : time advance function,

$M = \{AM_i, AM_j\}$: set of atomic DEVS models

with the following constraints:

$E = (X_i \cup Y_i \cup \{\varepsilon\}) \times (X_j \cup Y_j \cup \{\varepsilon\})$, where ε is a null event,

$S = S_i \times S_j$,

where $Q = (S_i \times R) \times (S_j \times R) = S \times R^n$ composed total states set,

$T \subseteq Q \times E \times Q$,

$ta: Q \rightarrow R$.

The transition relation T and time advance ta are subject to the following rules:

Rule 1 (asynchronous transition): Transit AM_i only for $\sigma_i = ta_i(s_i) - e_i \leq \sigma_j = ta_j(s_j) - e_j$:

(a) Transition relation

If $(s_i, s_i') \in \delta_{int,i}$ and $(s_i, !a) \in \lambda_i$ and $(s_j, ?a, s_j') \notin \delta_{ext,j}$, then it implies

$((q_i, q_j,), (!a, \varepsilon), (q_i', \dot{q}_j)) \in T$, for all $j \neq i$, where $q_i' = (s_i', 0), \dot{q}_j = (s_j, ta_j(s_j) - (\sigma_j - \sigma_i))$

(b) Time advance

$$ta((q_i', \dot{q}_j)) = min\{ta_i(s_i'), ta_j(s_j) - (\sigma_j - \sigma_i)\}.$$

Rule 2 (synchronous transition): Transit both AM_i and AM_j with event synchronization for $\sigma_i = ta_i(s_i) - e_i \leq \sigma_j = ta_j(s_j) - e_j$:

(a) Transition relation

If $(s_i, s_i') \in \delta_{int,i}$ and $(s_i, !\,a) \in \lambda_i$ and $(s_j, ?\,a, s_j') \in \delta_{ext,j}$, then it implies

$((q_i, q_j,), (!a, ?a), (q_i', q_j')) \in T$, for all $j \neq i$, where $q_i' = (s_i', 0), q_j' = (s_j', 0)$

(b) Time advance

$$\mathrm{ta}((q_i', q_j')) = \min\{\mathrm{ta_i}(\mathrm{s_i'}), \mathrm{ta_j}(\mathrm{s_j'})\}$$

Rule 3 (lossy transition): Transit only AM_i with **event lost** for $\sigma_i = ta_i(s_i) - e_i \leq \sigma_j = ta_j(s_j) - e_j$:

(a) Transition relation

If $(s_i, s_i') \in \delta_{int,i}$ and $(s_i, !\,a) \in \lambda_i$ and $(s_j, ?\,a, s_j') \in \delta_{ext,j}$ with event a lost, then it implies

$((q_i, q_j,), (!a, \varepsilon), (q_i', \dot{q}_j)) \in T$, for all $j \neq i$, where $q_i' = (s_i', 0), \dot{q}_j = (s_j, ta(s_j) - (\sigma_j - \sigma_i))$

(b) Time advance

$ta((q_i', \dot{q}_j)) = min\{ta_i(s_i'), ta_j(s_j) - (\sigma_j - \sigma_i)\}$.

Note that Rule 1 is asynchronous type of interaction when one model that has the smallest residual time is ready to send an output event while the other is at a state not ready to accept the event; therefore only the sender model will transit. The second rule explains a case where both the sender model and the receiver model with respect to a same event are ready to interact at right time at right state; then both of the model transit to start their new states. The third is intrinsic to C-DEVS, where both the sender model and the receiver one are ready to interact against an event, but the event is lost due to a communication mishap. Then only the sender model transits asynchronously in the same way with the first one.

We remark that the transition rules above of two atomic DEVS models can be easily extended to those of multiple atomic models by assuming model AM_i has the smallest resual time among models and applying the three rules for all the remaining models.

2.3 C-DEVS Model Construction Algorithm

A C-DEVS model for a set of atomic DEVS models can be constructed analytically by the C-DEVS formalism. We start with a given initial total state then we choose among the DEVS models a model with the least residual time. Holding on the model chosen, we pick one of the remaining models and apply the transition rules until all the models are examined. For each destination states determined by the transition rules we repeat the process described above. It ends when there is no destination state left to examine. Formally the procedure to construct a C-DEVS model from a set of models M is as follows.

1. Let $q_0 \in Q$ be the initial state, $Q_{done} \subseteq Q$ the finished states set and $Q_{exam} \subseteq Q$ the set of states to be visited. Initialize that $Q_{done} = T = \emptyset$, $Q_{exam} = \{q_0\}$. Then define $ta(q_0) = \min_{k=1..n}\{\sigma_k\}$, $\sigma_k = ta_k(s_k) - e_k$.
2. Pick a composed total state $q = (q_1, q_2, \ldots, q_n) \in Q_{exam}$.
3. Choose an element total state $q_i = (s_i, e_i)$ of the composed total state q, where $\sigma_i = \min_{k=1..n}\{\sigma_k\}$, $\sigma_k = ta_k(s_k) - e_k$.
4. Apply every transition rule for q_i and each element state q_j of q, $j \neq i$, which add up valid transitions and time advances into T and ta. If a destination state $q' = (q_1', q_2', \ldots, q_n')$ is obtained, then insert q' into Q_{exam} only if $q' \notin Q_{done}$.
5. If we have finished step 4, then move the composed state q from Q_{exam} to Q_{done}.
6. Repeat step 2-5 until $Q_{exam} = \emptyset$ or a predestined condition meets.
7. Then, we finally get CDEVS $= < E, S, T, ta, M >$.

The procedure describe above does not guarantee finiteness of transition relation even though the sets S and E are finite, for a composed total state is a pair of a composed discrete state and an elapsed time that might be infinite. However this state explosion problem is not of concern of the paper and in most cases the timed behavior converges if designed correctly in authors' experience.

3 Case Study: Safety Analysis of a Railroad Crossing System

3.1 The Problem Statement

Consider a railroad crossing system (RRC) in which trains and cars share the same crossing area. To prevent a disastrous accident we need safety measures such as a gate that blocks cars passing while a train approaches, and sensors that detect trains approaching and exiting the area. Thus the computer-based controller has two design objectives:

(1) (*safeness*) Prevent any accident by sensing the train's position and manipulating the gate in a right order as well as in a right time.
(2) (*liveness*) Cars eventually go through the gate in a finite time.

We also assume that messages between components of the railroad crossing system might be lost during the bad communication environment due to noise, software bugs or any human errors. To acquire a robust control system that satisfies the objectives, first we need to model the railroad crossing system and design a controller that *might* meet the objectives. Then we need to assure the safeness of the controller before implementation. DEVS formalism is an appropriate specification method for this type of problem since DEVS is very suitable specification formalism for modeling, simulation and analysis of real-time discrete event systems.

3.2 RRC DEVS Models

The railroad crossing system can be specified with three atomic models: TRAIN, GATE and CONTROLLER as in Fig. 1. Each circle denotes a total state (timed state) and a dotted arc an internal transition with an output event, and a solid arc an external transition with an input event. An output event is prefixed by '!' and an input event '?'. As assumed before, events with the same name are all coupled together and the system model is flattened into a set of atomic models.

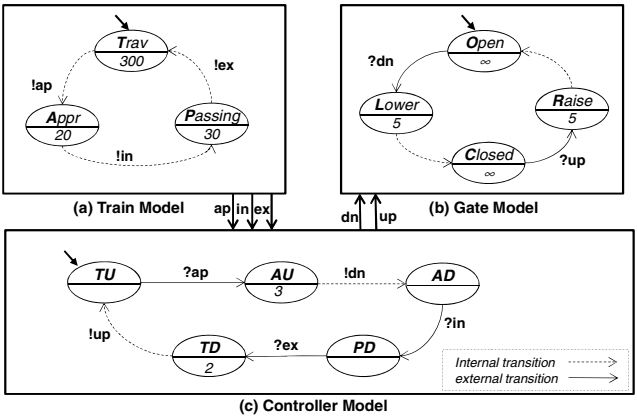

Fig. 1. A Railroad Crossing Model: Train, Gate and Controller

These atomic models in the figure can be mathematically specified in DEVS as follows.

TRAIN $=< X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta >$
$X = \{\}, Y = \{!\,ap, !\,ex, !\,in\}$,
S $= \{Trav, Appr, Passing\}$,
δ_{int}(Trav)=Appr, δ_{int}(Appr)=Passing, δ_{int}(Passing)=Trav,
$\lambda(Trav) = !\,ap$, $\lambda(Appr) = !\,in$, $\lambda(Passing) = !\,ex$,
$ta(Trav) = 300$, ta(Appr) $= 20$, $ta(Passing) = 30$.

GATE $=< X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta >$
$X = \{?\,dn, ?\,up\}, Y = \{\}$,
S $= \{Open, Lower, Closed, Raised\}$,
δ_{int}(Lower)=Closed, δ_{int}(Raise)=Open,
δ_{ext}(Closed, ?up)=Raise,
$\lambda(Lower) = \varepsilon$, $\lambda(Raise) = \varepsilon$,
$ta(Lower) = 5$, ta(Raise) $= 5$.

CONTROLLER$=< X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta >$
$X = \{?\,ap, ?\,in, ?\,ex\}, Y = \{!\,dn, !\,up\}$,
S $= \{TU, AU, AD, PD, TD\}$

δ_{int}(AU)=AD, δ_{int}(TD)=TU,
δ_{ext}(TU, ?ap)=AU, δ_{ext}(AD, ?in)=PD, δ_{ext}(PD, ?ex)=TD,
$\lambda(AU) = !\,dn$, $\lambda(TD) = !\,up$,
$ta(AU) = 3, ta(\text{TD}) = 2.$

As we can see, the controller model receives the position events of the train and sends commands to the gate model. For reasons it is assumed that these events might be lost.

3.3 RRC C-DEVS Model Constructed without Event Loss

The safety analysis can be conducted either on the fly during C-DEVS model construction phase or be checked after we once get the C-DEVS model. For safety analysis we first need to define a set of bad states, $B \subseteq S$, any element of which should not be reached from the initial state. In the case study we define the bad set as $B = \{(P, O, -)\}$, that is, the gate is open while the train is passing the crossing area, which is very dangerous.

To be specific, Fig. 2 shown below depicts the transition relation and time advance function of a C-DEVS model for the RRC model, which is obtained by applying only Rule 1 and Rule 2, but not Rule 3. This analysis is conducted to see the composed behavior if no event is lost. The initial composed total state is ((T,O,TU),[0,0,0]) or equivalently ((T,O,TU),[300, ∞, ∞]) in residual form.

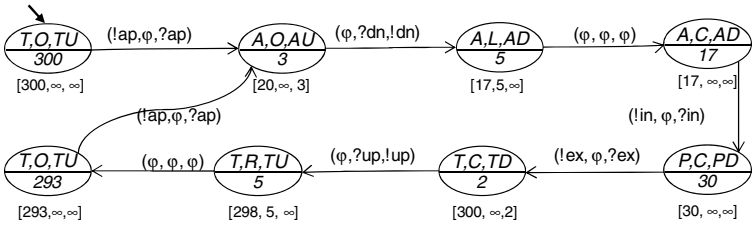

Fig. 2. Transition diagram of RRC model without message loss

We can easily see that each circle represents a composed state (above) with the time advance (below) of the state. A vector below a circle denotes the ordered residual times of each model at the states. For example, the circle upper-left represents that the initial state (T,O,TU) has its residual time vector $[300, \infty, \infty]$, which means model TRAIN is at state Trav at which it may stay for 300; GATE is at state Open and CONTROLLER TU, and both wait forever. Consequently at the composed state the system will stay for 300. An arc represents a transition with an associated event when the time advance is fired.

We conclude in this analysis that there is no bad state (P,O,-) reached from the initial state (T,O,TU) and thus it is said to be safe if we have no message (event) lost. Note that ((T,O,TU),[300, ∞, ∞]) and ((T,O,TU),[293, ∞, ∞]) are not the same state

as the residual times are different. Furthermore we find that the system satisfies the liveness since the state sequence of the gate cycles through (-,O,-) → … → (-,C,-) → … →(-,O,-) in 355. The gate is available for 296 and unavailable for 59 in one cycle, which means cars eventually pass the crossing area.

3.4 RRC C-DEVS Model Constructed with Event Loss

The purpose of this analysis is to know how the system behaves if events or messages are lost. Fig. 3 is the resultant C-DEVS model for the railroad crossing system model in Fig. 1 using the analysis algorithm. It is assumed that all events might be lost and thus we apply all the three transition rules. We can see that in normal case without event lost the behavior is the same as in Fig. 2. With any event lost, however, we can also see the system digresses to unexpected sequences of states.

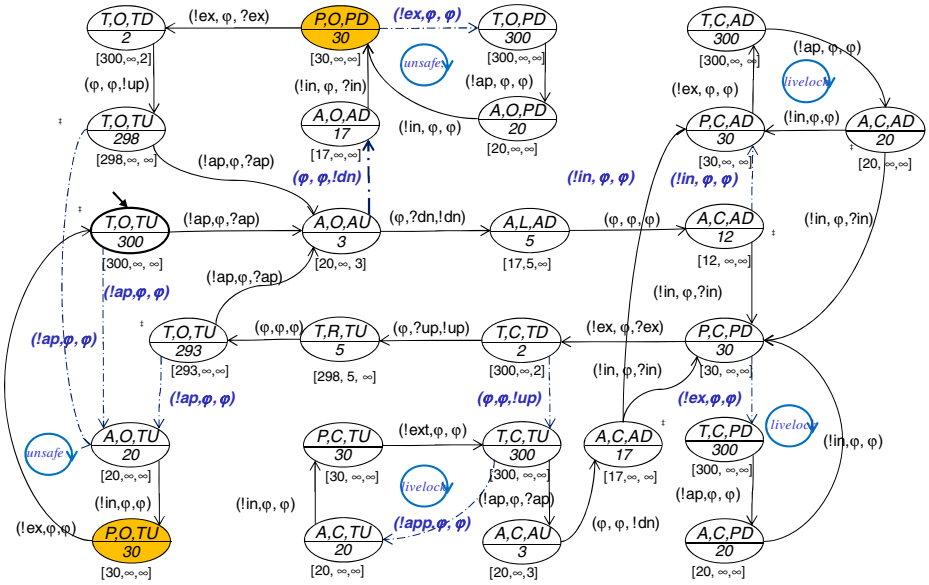

Fig. 3. C-DEVS model with message loss : lost message = {ap, in, ex, dn, up}

The RRC model starts with an initial composed timed state ((T,O,TU),[0,0,0]), which means model TRAIN is at timed state (T,0), and GATE (O,0) and CONTROLER (TU,0). Taking into account the time advances of the atomic models, the initial composed residual state is ((T,O,TU),[300,∞, ∞]) and the time advance of the composed state is 300. After the time elapsed, the train fires and generates an output event !ap, which can be accepted by the controller; then the two models transit synchronously by (!ap,ε,?ap) to a residual state ((A,O,AU),[20,∞,3]). However, it is possible the output event could be lost and the controller cannot receive the event. In this case the system will transit to a residual state ((A,O,TU),[20,∞,∞]) where the train is approaching while the gate is open and the controller never perceive the approching due to the event loss; thus after the time 20 has elapsed, the train is start to

enter the crossing area while the gate is still open, i.e., ((P,O,TU),[30,∞,∞]) that is a bad state. This means if event !ap is lost at (T,O,TU) then the whole system become unsafe. Moreover, we have another bad state ((P,O,PD),[30,∞,∞]) at top of Fig. 3, which comes from ((A,O,AU),[20, ∞, 3]) by the gate's losing event !dn sent from the contoller.

Thus we can conclude by this analysis that the system is unsafe in case of event losses, especially with event !ap and !dn lost. Fortunately any of the other event losses does not affect the safeness of the whole system. However, we can find that those violate the liveness as annotated by 'livelock' in the figure.

4 Conclusion

This paper proposed a safety analysis method for real-time discrete event systems based on the C-DEVS formalism that is mathematically renewed here. C-DEVS itself only defines the interaction mechanism between atomic DEVS models with a representation of the whole behavior as the result of the interaction. It is beneficial for analysis purposes to get the all possible behavior of DEVS models without simulation. We proposed a procedure to construct the C-DEVS model of a set of atomic DEVS models and illustrated the usefulness by a case study, the railroad crossing control system. For this, we first modeled the system in the DEVS formalism and then obtained the C-DEVS model using the construction procedure proposed here. From the resultant model we formulated and analyzed the safety of the system. The state explosion problem due to time, however, is inevitable and is not concern of the paper, though. As a future work, we will extend this research to the controller design problem, where the time advance is not fixed. In addition to this, we have a plan to develop a tool to automatically construct a C-DEVS model from an implemented DEVS simulator.

References

1. Leveson, N.G., Stolzy, J.L.: Safety Analysis Using Petri Nets. IEEE Trans. Software Engineering SE-13(3), 386–397 (1987)
2. Song, H.S., Kim, T.G.: Application of Real-Time DEVS to Analysis of Safety-Critical Embedded Control Systems: Railroad Crossing Control Example. Simulation 81, 119–136 (2005)
3. Bengtsson, J.E., Yi, W.: Timed Automata: Semantics, Algorithms and Tools. In: Desel, J., Reisig, W., Rozenberg, G. (eds.) ACPN 2003. LNCS, vol. 3098, pp. 87–124. Springer, Heidelberg (2004)
4. Boucheneb, H., Barkaoui, K.: Relevant Timed Schedules/Clock Vectors for Constructing Time Petri Net Reachability Graphs. Discrete Event Dynamic Systems 21(2), 171–204 (2011)
5. Kim, T.G.: DEVS Formalism for Modeling of Discrete-Event Systems. In: Handbook of Dynamic System Modeling, ch. 6. Chapman & Hall/CRC (2007)
6. Zeigler, B.P., Kim, T.G., Praehofer, H.: Theory of Modeling and Simulation. Academic (2000)