ISSN No. 0976-5697

Volume 4, No. 8, May-June 2013

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

A Formalism for Checking Consistency of Component-Based Real-Time Systems

Dr. Jaber Karimpour
Department of Computer Science
University of Tabriz
Tabriz. Iran
Karimpour.jaber@gmail.com

Sina Zangbari Kouhi
Department of Computer Science
University of Tabriz
Tabriz. Iran
Zangbari89@ms.tabrizu.ac.ir

Faranak Nejati
Department of Computer Science
University of Tabriz
Tabriz. Iran
Najati89@ms.tabrizu.ac.ir

Abstract: In real-time systems, specification, analysis and verification are very important research topics and practical implementation of real time systems need great accuracy. Component-based design is an approach to design and manufacture real-timed systems. This method is a reuse-based technique which improves some of requirements such as increasing their usability, flexibility, adaptation, reducing the cost of software products and Etc. In this Paper, we present a theory for modelling behaviour, interactions, and processes of component based real-time systems. For this purpose, we used concepts of time automata, time interface automata and discrete event components. Each timed component is made based on its corresponding timed interface. Timed Interfaces identify all input-output expects and time of occurrence of any processes. One of challenges in production of systems based on components is that the component produced conforms to its equivalent interface. We developed a theory to check the consistency between timed component and a given timed interface. Theory presented in this paper is considered as a framework for formal specification and verification.

Keywords: real-time systems, component-base systems, time automata, time interface, consistency.

I. INTRODUCTION

Verification of systems is the main and principal part in development of systems. This process is very important in systems such as real-time systems which have critical processes. In real-time systems [20] not only logical accuracy of operations and processes, but also execution time should be considered. These systems include time limitation and if the operation is not executed on demanded time, the system is failed and causes heavy damages such as loss of resources or even endangering the human lives. These systems need precision and speed. For example, air bag system in a car is a simple sample of real-time systems. This system should operate in a short time and if is not completed in time limitation; it means system is failed.

Some examples of these systems include digital control system, signal process, telecommunication system, and industrial systems. Component-based design [19] is a useful way todesign and develops real time systems. Since use of components in industry is increased during recent years, there is more attempts for designing efficient components based on principles and structure of systems designed. The method used for modeling of component based real time systems should have appropriate syntax and semantic able to precise specification of timed components, and provide faultless way to their composition and break or reduce the state space explosion (One of the problems in checking consistency).

To solve the state space explosion problem a variety of approaches have been proposed. These methods are classified into multiple categories such as abstraction [21, 22], on the fly model checking [15] and compositional verification [16, 17 and 18]. In this paper for attacking the state space explosion we used compositional verification which is based on “divided and conquer”. This method break up a complex system into subtasks which involves the checking of its components and each subtask verified independently on the equivalent component.

In this paper, we want to extend a formal definition of timed discrete event component (TDEC) and verify consistency with using the promoted theories of Timed Automata [1], Timed Interface Automata [2] and Discrete Event Components [3].

a. Timed automata [1, 10],is a theory for description, analyzes and verify timed systems. In this paper we use the work of Alur and Dill [1] which extended finite automata with a set of clock-variables. Each transition associated with a set of clock-variables and clock-guards; if current clock-variables satisfy the clock guard then a transition can be take place. The clock-values increase with the same rate and before starting of any transition set to zero.

b. A timed interface automaton [2], a precise definition of components interaction with each other and with the environment, is independent components specified by their interfaces. In [2] Alfaro and Henzinger established a framework successfully for timed Interface automata which modeled as a two player game: the input player is the inputs that each component accepts from the environment, and the player output is the achievable outputs of component. The component is usable in any design if its corresponding TI will be a well-formed, this means that, there is some environment which satisfies component expects.

c. In [3], Jin promoted a formal syntax for discrete event component and used interface automata (IAs) to describe component assumptions. Interface automata is a formalism that proposed by Alfaro and Henzinger [6, 7]. In Jin's theory are general definitions of Reactive Transition Systems. The conformances of a component to an IA determined by calculating local state space of component with regard to IA and search for error states in this local space. Local state space is Synchronized product of two composed components.

Since specification and implementation in real timed systems are very important, the proposed framework in this paper should support certain characteristics: *simple and unambiguous syntax and semantic*, which allows to specify the set of requirements and assumption of components. The first challenge of making component-based real-timed systems is well-defined timed components. *Structural composition and communication*, which combine two or more separate components, that each of them has their own properties, assumptions and specifications. In the composite component, interacts can determine by synchronization vectors [8] of input and output events. *Consistency*, components are implementation of its corresponding interface, hence, we need a reliable method to checking the conformance of TDEC with TIAs.

A. Paper Outline:

Next section includes some definitions of labeled transition systems [11, 12], discrete event components [3] and timed interface automata [2]. In section 3, we first introduce TDEC and the composition of two TDEC by developing the work of Jin, for this purpose, we use the regular framework of Arnold and Henzinger [2]. Then we check the conformance of TDEC to a given TIA with searching in local state space of composite TDEC and TIA (synchronized product) for the lack of unexpected states. Finally, section 4, constitutes a summary of result of this paper and future work.

II. PREVIOUS WORKS

Due to the need for well-defined specification of components and ensure their correctness, many proposal have been established. For example, time Petri net, timed automata, finite-state machine, labeled transition system, timed interface automata, labeled transition systems (LTSs) and discrete event component, which are theories in computer sciences and used for modeling systems and components. In the following, some of these theories that are related work will be summarized.

A. Labeled transition system:

A labeled transition system is a graph consisting set of nodes and set of edges. The nodes present states and transitions indicated by the edges which labeled whit actions. To test systems by LTSs, we should simulate behavior of systems via LTS semantic. For this purpose, we should consider quantitative aspects. A LTS is defined as follow:

Definition 1: A labeled transition system is 4-tuple $< S, s_0, \Sigma, \Delta >$, where:

a. S is a non-empty finite set of states;
b. s_0 is the initial state;
c. Σ is a set of labels; consists of two countable disjoint set of input labels I, and set of output labels O;
d. $\Delta \subseteq S \times 2^{(I \cup Out)} \times S$ is a set of the transition relation;

B. Discrete event component:

In [3], Jin introduced general definition of reactive transition systems (RTS) which is similar to labeled transition systems (LTS). Difference of RTS and LTS is in control of actions (called internal event in RTSs and labeled in LTSs). RTSs may control input and output actions, but input actions are out of control. Input actions are under control of environment.

Synchronization vector is used for composition of RTS. This concept presented in [8] by Arnold and Nivate which is presented as a general mathematical model for any empty and non-empty set of synchronize processes, components may accept any number of input at any time, but outputs are limited, therefore, author in [3] changed concept of synchronization vector in a way that any vector exactly has one output and the number of input events:

Definition 2: Consider $\Gamma_0, \Gamma_1, \dots, \Gamma_n$ are the sets of events, R is a relation such that $R \subseteq \prod_{i=1}^{m} \Gamma_i$, a set $\Gamma \subseteq \bigcup_{i=1}^{m} \Gamma_i$ and $(\Gamma \cap \Gamma_i \cap \Gamma_j)$ is empty for all i, j which: $i, j = 1,2 \dots m \wedge i \neq j$. Let projections $\pi_i : R \to \Gamma_i$ for $0 \leq i \leq n$ and sets of keys $\kappa_r = \{\pi_i(r) \in \Gamma | 0 \leq i \leq n\}$ for $r \in R$, then r indexed by Γ if there exists:

a. $|\kappa_r| = 1 for\ all\ r \in R;$
b. $\forall e \in \Gamma, \exists r \in R, e \in \kappa_r \wedge \forall \acute{r} \in R \backslash \{r\}, e \notin \kappa_r.$

Since, RTS has limitations and they are not suitable for practical use. Therefore, the author in [3] has developed these systems as follows:

a. Any event includes two parts of kind and value. Kind is used for classification of events and value for exchanged data.
b. Several output-input ports are added to RSTS. Any port indicates special type of events. The component shall use

these ports for relationship with other components and environment.

Definition 3: A discrete event component (DEC) is a tuple$D = (S, s_0, \rho, \theta, \Gamma, \Delta)$, where:

a. S is a set of states;
b. s_0 is the initial state;
c. ρ is a set of ports, which consisting of two disjoint ports: input ports ρ^i and output ports$\rho^0, \rho^i \cap \rho^0 = \emptyset$.
d. $\theta: \rho \to 2^v$, is a total function which mapping each port to a subset of values form the universe;
e. $\Gamma = \Gamma^I \cup \Gamma^o \cup \Gamma^H$ is a set of events, where $\Gamma^I = \{(e, v) | e \in \rho^I, v \in \theta(e)\}$ is a set of input events, $\Gamma^O \subseteq \{(e, v) | e \in \rho^O, v \in \theta(e)\}$ is a set of output events, and set of internal event has not any relation with set of port ρ.
f. Δ is a set of transition.

C. Time interface:

Alfaro and Henzinger [2] presented a theory which is able to modeling the timing of the behavior and interaction a component. We can design component-based real-timed systems in the two ways: *specification of component interface*, if an interface is well-formed, then the component is usable in any design. *Checking for interface compatibility*, if composition of two interfaces is well-formed, then we can say they are compatible. A timed interface automaton defined as follow:

***Definition 4*: *(Timed Interface Automata)*:** A TI is a tuple $A = (S, s_0, \Sigma^I, \Sigma^O, \rho^I, \rho^O)$, where:

a. S is a set of all state;
b. s_0 is the initial state;
c. Σ^I is the set of all input action, $\Sigma_A^I = acts_A^I \cup T$ which $acts_A^I$ is immediate input. T is a set of timed action;
d. Σ^O is the set of all output action, $\Sigma_A^O = acts_A^O \cup T$ which $acts_A^O$ is immediate output;
e. $\rho_A^I \subseteq S_A \times \Sigma_A^I \times S_A$ is the input transition relation;
f. $\rho_A^O \subseteq S_A \times \Sigma_I^O \times S_A$ is the output transition relation.

III. PROPOSED FORMALISM

Timed discrete event component, has three kinds of events: input events, output events, internal events. Events are transitions that occur between components. These transitions can be classified in two ways: The *Instant transitions*, which must be done rapidly upon entry to a state. The *Timed Transition*, which describes the time interval for each of transition.

We used definition of Timed Automata [1], Timed Interface Automata [2] and Discrete Event Component [3] to describe the Timed Discrete Event Components (TDEC) as follow:

Definition 5: (*Timed Discrete-Event Component*)**:** A TDEC is a tuple $C = (S, s_0, \chi, \Sigma, inv, \Delta, \alpha, \theta)$, where:

a. S is a finite set of state,
b. s_0 is the initial state,
c. χ is a finite set of clocks;
d. $\rho$is a set of ports, which consisting of input ports $\rho^I$and output ports ρ^O, $\rho^I \cap \rho^O = \emptyset$;
e. $\theta: \alpha \to 2^v$, which mapping each port to a subset of real value;
f. $\sum$ is a finite set of all timed events, which consisting of three mutually disjoint set of input timed events $\sum^{\mathrm{I}}$, output timed events $\sum^{\mathrm{O}}$, and internal timed events $\sum^{\mathrm{H}}$, where $\sum^{\mathrm{H}}$ is a pair (t, e) where e is an action taken by an automata C after t ∈ R_+ and set of internal event $\sum^{\mathrm{H}}$ has not any relation with set of port ρ, $\Sigma^I = \{\langle t.x.v\rangle | t \in R_+, x \in \rho^I, v \in \theta(x)\}, \Sigma^O \subseteq \{\langle t.x.v\rangle | t \in R_+, x \in \rho^O, v \in \theta(x)\}$;
g. $inv: I \to \chi$ is a function that maps each state of TDEC to its invariant, TDEC has three kind of invariants: input invariants which specify upper bounds (U) and lower bounds (L) for the time of input events, output invariants, which specify upper bounds and lower bounds for the time of output events and internal invariants which specify upper bounds and lower bounds for the time of internal events.
h. $\Delta \subseteq S \times \Sigma^{I/O} \times 2^{\Sigma^H} \times \chi \times 2^X \times \acute{S}$ is a finite set of transitions, and $(S, g, e, r, \acute{S}) \in \Delta$ is a step and we often write $S \xrightarrow{g,e,r} \acute{S}$, state S is the source and $\acute{S}$ is destination of the transition, g is a guard on the clock value that specifies when the transition can be taken, $e \in \Sigma$ is an timed event, and in each transitions r resets clock values to 0.

Let V is a function and we use $V \vDash \varphi$ to mean that the clock values satisfy the guard g. If the guard φ is false under the valuation V, we write$V \nvDash \varphi$.

Definition 6: A *timed trace* of TDEC is a sequence of timed events $\xi = (t_1, e_1)(t_2, e_2), \ldots, (t_i, e_i), \ldots$where $t_i < t_{i+1}$ for all $i \geq 1, e_1, e_2, \ldots, e_i \in \Sigma$. The trace projection $\pi(\xi)$ of ξ on C is a timed event sequence consisting of the action that C takes which defined by:

$$\pi_C(\xi) := \begin{cases} 1.\lambda & if\ \xi = \lambda \\ 2.(t,e).\pi(\acute{\xi}) & if (t,e) \epsilon \Sigma^H \Sigma^O \\ 3.(t,e^i).\pi(\acute{\xi}) & if (t,e^i) \in \{env, other\ TDEC\} \\ 4.\pi(\acute{\xi}) & otherwise. \end{cases}$$

a. If TDEC C has no timed events (t, e), then the trace projection is empty.
b. If TDEC C has an internal or output timed event (t, e), then (t, e) added to the trace projection.
c. If (t, e) is an input event from environment or other TDEC which corresponds to a synchronization vector r

with $\pi_c(r)$ (an input event of c), then $\pi_c(r)$ is added to the trace projection.

d. In otherwise trace projection remains unchanged.

The set of timed enabled events at s are timed event that their guards always satisfied, $V \vDash \varphi$, and there is at least one $\acute{s}$ such that $s \xrightarrow{g,e,r} \acute{s}$, for denote the timed enabled events often we write *en*. *en* have two kind of disjoint set of input en (en^I) and output en (en^O). A TDEC C is input-universal if$\forall s \in S_C, en^I(s) = \Sigma^I$. The TDEC C is called mirror if $\Sigma_m^I = \Sigma_C^O$ and $\Sigma_m^O = \Sigma_C^I$.

The TDEC may give a rise to two kinds of error: *error state*, where $s \notin S_c$, and *timed error*, where $V \nvDash g$ and $(U, L) \notin I$. The set of trap step in TDEC is $\Delta_{trap} = \{(s, g, e, r, \perp) | s \in S_C, V \vDash g, e \in \Sigma_C^I \backslash en_C^I(s)\}$,Trap step while occur that TDEC has an timed event that is not in set of timed enable input.If $\Delta_{trap} \neq \emptyset$ then the input-universal version of TDEC defined as follow:

(a). $(s_L^0, S \cup \{\perp\}, \Sigma_C, \Delta_L, \Delta_{trap}, \Delta_{idle})$

A. Composition of two TDEC:

Two TDEC p, q are composable if they have no shared output actions, $\Sigma_p^O \cap \Sigma_q^O = \emptyset$. We can compose two components as follow:

Definition 7: For two composable TDEC C_1 and C_2, the product $C_1 \otimes C_2$ is the TDEC N, where:

a. Γ is a set of timed events consisting of three disjoint sets of input timed event $\Gamma^I = \Sigma_{C_1}^I \cup \Sigma_{C_2}^I$ \ shared input event (C_1, C_2) and output timed events $\Gamma^O = \Sigma_{C_1}^O \cup \Sigma_{C_2}^O$ and $\Sigma_{C_1}^O \cap \Sigma_{C_2}^O = \emptyset$. We let internal timed events $\Gamma^\# = \left(\Sigma_{C_1}^I \cap \Sigma_{C_2}^O\right) \cup \left(\Sigma_{C_1}^O \cap \Sigma_{C_2}^I\right) \cup \Sigma_{C_1}^\# \cup \Sigma_{C_2}^\# \cup \{\varepsilon\}$.

b. $W = C_1 \cup C_2$.

c. $R \subseteq \Gamma^\# \times \Pi_{C \in W} \Gamma_C^\#$ is relation index by $\Gamma^I \cup \bigcup_{c \in W} \Gamma_C^O, \Gamma^I = \Gamma_{env}^O$.

A composition of TDEC is called closed of $\Gamma = 0$, or open otherwise. We often write N = (W, R) for a close composition of TDEC N.

Definition 8: Consider a composite TDEC C = (Γ, W, R) such that all l ∈W are input universal. The synchronized product of C is a TDEC $L = \{S, s_0, \chi, \Sigma, inv, \Delta, \rho, \theta\}$, where:

a. $S_{C_1 \times C_2} = S_{C_1} \times S_{C_2}$, and $S_{C_1 \times C_2}^0 = (S_{C_1}^0, S_{C_2}^0)$;

b. Inv (The time bounds (L, U)) in C:

a) If $(t, e) \in (\Sigma_i - \Sigma_j) \cup (\Sigma_j - \Sigma_i)$ *then* (L_e, U_e) *remain unchanged;*

b) If $(t, e) \in \Sigma_i \cap \Sigma_j$ *then its time bounds in C are defined as follow:*

$(L_e, U_e) = (max(L_{i,e}, L_{j,e}), min(U_{i,e}, U_{j,e}))$.

c. g is a guard on the clock value that specifies when the transition can be taken, the guard expression of the transition in C:

a) If $(t, e) \in \Gamma^\# \backslash (\Sigma_{C_1}^\# \cup \Sigma_{C_2}^\# \cup \{\varepsilon\})$ *then g is the conjunction of those of* g_{C1} *and* g_{C2}.

b) If $(t, e) \in \Sigma_{C_1}$ *and* $(t, e) \notin \Sigma_{C_2}$ *then g is the same as with* g_{C_1}.

c) If $(t, e) \notin \Sigma_{C_1}$ *and* $(t, e) \in \Sigma_{C_2}$ *then g is the same as with* g_{C_2}.

d. Δ is a set of transitions $((q_i, q_t), (g_i, g_i), e, r, (\acute{q}_i, \acute{q}_i))$ such as, for i = 1,2:

a) If $(t, e) \in \Gamma^I$ *then:*

{($s \xrightarrow{g,e,r} s'$*) | V⊨ g,* $e \in \sum_C^I$*,* $\exists f \in R$ *(R is synchronization vector),* $producer_f = env \wedge e = \pi_{env}(f)$ *or (s' = s)}∪{(*$s \xrightarrow{g,e,r} s'$*) | V⊨ g,* $e \in \sum_C^I$*,* $\exists f \in R$ *(R is synchronization vector),* $\exists k \notin W$, $producer_f = k$, $e = \pi_k(f)$ *or (s' = s)}.*

The timed input event, which triggered the transition from s to s' has been produced in two different ways: *the environment* and *other components*.

b) If (t, e)∈Γ^O *then:*

{($s \xrightarrow{g,e,r} s'$*) | V⊨ g, f ∈ R, ∃L ∈ W, L =* $producer_f \wedge e \in \pi_{env}(f) \wedge (s_L, g_L, \pi_L(f), r, s'_L) \in \Delta_L$ *or* $s'_i = s_i$*} ∪ {(*$s \xrightarrow{g,e,r} s'$*) | V⊨ g, f ∈R, ∃L ∈W, ∃k ∉W, k = consumer, L =* $producer_f \wedge e \in \pi_k(f) \wedge (s_L, g_L, \pi_L(f), r, s'_L) \in \Delta_L$ *or* $s'_i = s_i$*}*

Let (t, e) is a timed output event, thus, this timed event produced by one of the components which its consumer is the environment or other components.

c) If (t, e)∈$\Gamma^\#$ *then:*

{($s_i \xrightarrow{g,e,r} s'_i$*) | V⊨ g, f ∈R, ∃L ∈W, e ∈*$\sum_L^H \wedge (s_L, g_L, e_L, s'_L) \in \Delta_L \wedge (\forall i \in W \setminus \{L\}, s'_i = s_i)\}$ *∪ {(*$s \xrightarrow{g,e,r} s'$*) | V⊨ g, f ∈ R, ∃L ∈ W, e ∈*$\sum_L^O$*, L =* $producer_f$*, e ∈*$\pi_L(f) \wedge \pi_{env}(f) = \epsilon \wedge (s_L, g_L, e, s'_L) \in \Delta_L$ *or s' = s} ∪ {(*$s \xrightarrow{g,e,r} s'$*) | V⊨ g, ∃L ∈ W, e ∈*$\sum_L \wedge \nexists f \in R, (s_L, g_L, \pi_L(f), r, s'_L) \in \Delta_L \wedge (\forall i \in W \backslash \{L\}$, *s' = s)}*

The Timed internal event can occur in several ways: The timed internal event taken by any components in the composite TDEC which is a timed internal step of the TDEC. The timed output event of any components which has no contiguity on environment or other components are not in composite TDEC. The timed event can be taken by any component but no corresponding synchronization vector exists.

In designing of timed components, the run time of transitions should be considered with corresponding interface. In so doing, those transitions that are instant transitions should

be designed in a way that it could perform immediately, but in timed transitions two factors should be considered:

a. Timed guard (g) that exists in the interface.
b. Observing upper bound and lower bound.

Guards represents as (<, >, =, ≤, ≥), which in checking of consistency should be considered to use exactly those which are used interface.

Also the upper bound and lower bound of component should not exceed of those that are used in interface. The following examples are some cases in which we can design some components according to their interfaces.

In the following, we show Examples of guard expression on the timed interface automata and timed discrete event components:

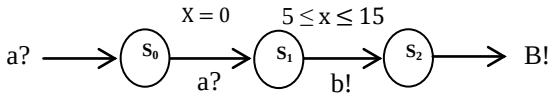

Figure 1: Timed Interface.

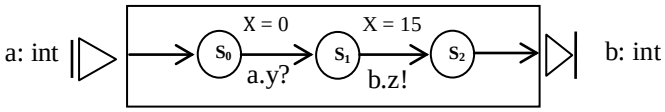

Figure 2: Timed Discrete Event Component.

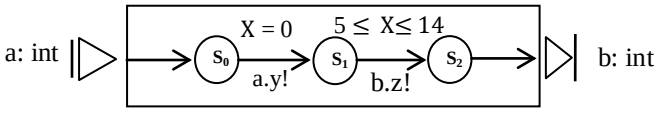

Figure 3: Timed DEC.

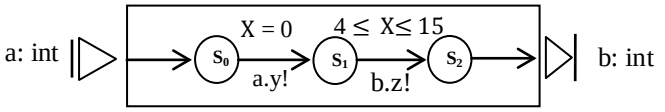

Figure 4: Timed DEC.

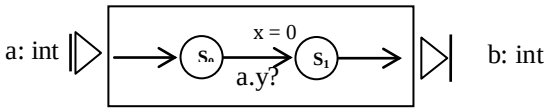

Figure 5: Timed DEC.

Another point is that, a component can accept one or some input from the environment or other components. If the number of output is smaller than or equal with the number of interface outputs, then likely to observe the system safety. In other word the output components considered subsets of their corresponding output interface. But having any number of inputs for a component is allowed because the environment will not offer them. The following examples are samples of input and output of the components which we can design from interface.

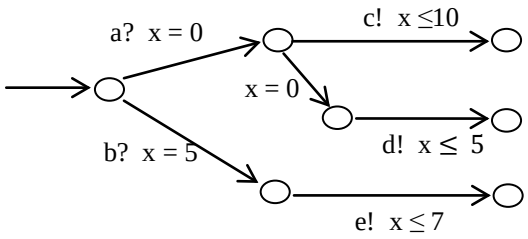

Figure 6: Timed Interface.

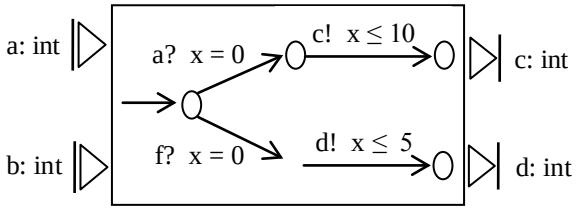

Figure 7: Timed DEC.

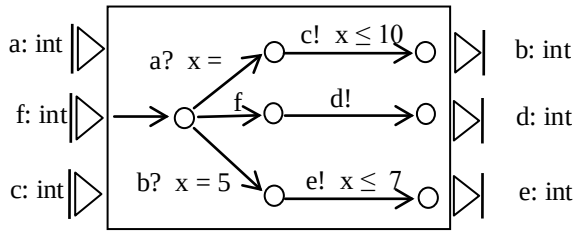

Figure 8: Timed DEC.

Definition 8 (Normalization): Consider TDEC C and TIA A. The Normalization operation on C is defined as follow:

$$Norm_L(c) = \begin{cases} L_{(\pi_C(s))} \coloneqq L_{(\pi_A(s))} & if \exists s \in S_C \text{ and } S_A, L_{(\pi_C(s))} < L_{(\pi_A(s))} \\ unchanged & others; \end{cases}$$

$$Norm_U(C) = \begin{cases} U_{(\pi_C(s))} \coloneqq U_{(\pi_A(s))} & if\ \exists s \in S_C \text{ and } S_A, U_{(\pi_C(s))} > U_{(\pi_A(s))} \\ unchanged & others; \end{cases}$$

Let n is a valuation function and we use $\eta_A \vDash \pi_C(q)$ to mean that upper bound and lower bound $\pi_C(q)$ are consistent with corresponding TA A.

To facilitate checking conformance, we need to change the interface which can act as a component. The TIA events considered as ports and each port has value, the TIA is able to produce all permissive data values and we call this the most abstract implementations. If MAI of TIA taken an unspecified timed input event then it goes to the error state. MAI is an input-universal version of TDEC. The most abstract

implementation (MAI) of TA A is a TDEC $J = (S, s_0, \chi, \Sigma_J, inv, \Delta_J, \rho_J, \theta)$, where:

b. $\rho_J^I = \Sigma_A^I$
c. $\rho_J^O = \Sigma_A^O$
d. $\Sigma_J^I = \{\langle t.x.v\rangle | t.x \in \Sigma_A^I, v \in \theta(x)\}$ and $\Sigma_J^O \subseteq \{\langle t.x.v\rangle | t.x \in \Sigma_A^O, v \in \theta(x)\}$ and $\Sigma_J^H = \emptyset$

B. Consistency of Timed Discrete Event Component:

To define consistency, Jin et al. [3] used alternating simulation [6] that always assumed the environment to satisfy the assumption of the specification. In this way, the TDEC can accept more inputs and provide less output from its TIA and lower bound and upper bound of each state of TDEC must not exceed of lower bound and upper bound from its corresponding TIA states. In the other words, time-conditions of component should be subset of TIA time-conditions.

Definition 9: Let a TDEC C and corresponding TA A such that $\Sigma_A^I \subseteq \alpha_C^I$ and $\alpha_C^O \subseteq \Sigma_A^O$. If there exists an alternating simulation relation $\prec \subseteq S_C$ g S_A; we can say C conforms to A and written C $\prec$ A such that $s^0_C \prec s^0_A$, for q $\prec$ s and $g_{q'} \prec g_s$ three following conditions should be hold:

a. Since TIA has not any internal timed event, when TDEC take an internal step from q the aftereffect of TDEC must imitate the previous state s, guard g and invariant of TIA:
e $\in \sum^H_C \wedge \vee \models$ g, $\exists$(q, g, e, r, q') $\in \Delta_C$ implies q' $\prec$ s, $g_{q'} \prec g_s$ and ($L_{q'}$ and $U_{q'}$) $\prec$ (L_s and U_s).

b. Since the output component isa subset of its corresponding interface outputs, the TDEC must not produce an output timed event that the TIA cannot produce.
t.x.v $\in \sum^O_C \wedge \vee \models$ g, $\exists$(q, g, *t.x.v*, r, q') $\in \Delta_C$ implies that $\exists$(s, g, *t.x*, r, s') $\in \Delta_A$ such that q' $\prec$ s', $g_{q'} \prec g_{s'}$, ($L_{q'}$ and $U_{q'}$) $\prec$ ($L_{s'}$ and $U_{s'}$).

c. Since the input interface should be a subset of component inputs, when the TDEC takes a timed event *t.x.v* from *q* the resultant state of TDEC must be simulate the resultant state of TIA:
t.x $\in \sum^I_A \wedge \vee \models$ g, $\exists$(s, g, *t.x*, r, s') $\in \Delta_A$ implies that $\forall$v $\in \theta_C$(x), (q, g, *t.x.v*, q') $\in \Delta_C$ such that q' $\prec$ s', $g_{q'} \prec g_{s'}$, ($L_{q'}$ and $U_{q'}$) $\prec$ (L_s and U_s).

Further, The TDEC must simulate guards of TIA and takes its timed event when timed event of TIA occurs (condition 2, 3).

In the following, theory 1 present a formalism to checking consistency of TDECs with using local state space, alternating simulation, equal or less output ports of TIA and timed conditions that respects time-condition of TIA.

Theorem 1: Let a TDEC C and corresponding TA A that $\sum^I_A \subseteq \alpha^I_C$ and $\alpha^O_C \subseteq \sum^O_A$, and J be MAI mirror of A. Let N = {W, G, R} be a closed composite of TDEC and w = {C, J}. Then the local state space of C with respect to A is the synchronized product of N and we let $L_\otimes$ = { $s^0_\otimes$, $S_\otimes$, $g_\otimes$, $\sum_\otimes$, $\Delta_\otimes$} as the local state space. Then we can say C conforms to A if and only if $\forall$s, q $\in S_\otimes$, $\pi_J(s) \neq \perp$, $\eta_J \models \pi_C(q)$.

Proof of sufficiency: Consider Θ = {(q, g, s) $\in L_\otimes$ | q $\in S_C$, s $\in S_A$, g = $g_\otimes$}, then using the induction and prove Θ is an alternative simulation relation between C and A. First, (q^0_C, g, s^0_A) $\in \Theta$ because $s^0_A = s^0_J$. Next, suppose (q, g, s) $\in \Theta$, then:

a) For e $\in \sum^H_C \wedge \vee \models$ g, if $\exists q \xrightarrow{g,e,r}_C q'$, then (q', s) $\in S_\otimes$ and since time-conditions of TDEC respects TIA time-condition then ($L_{q'}$ and $U_{q'}$) $\prec$ ($L_{s'}$ and $U_{s'}$) implies $g_{q'} \prec g_s$. Hence (q', g, s) $\in \Theta$;

b) For *t.x.v* $\in \sum^O_C$, *t.x* $\in \sum^O_A$ and thus *t.x.v* $\in \sum^I_J$. Since, J is input-universal, If $\exists q \xrightarrow{g,t.x.v,r}_J q'$, then $\exists$s' $\in S_J \wedge \vee \models$ g, (q, s) $\xrightarrow{g,t.x.v,r}_\otimes$ (q', s'). Since (q', s') $\in S_\otimes$, from the condition of the theorem, we have $\not\in \perp$ and time-conditions of TDEC respects TIA time-condition then $L_{(\pi_C(q'))} \prec L_{(\pi_A(s'))}$, $U_{(\pi_C(q'))} \not\succ U_{(\pi_A(s'))}$ and $g_{q'} \prec g_{s'}$. Hence s' $\in S_A$, $\eta_A \models \pi_C(q)$ and (q', g, s') $\in \Theta$;

c) For $t.x \in \sum^I_A$, $t.x \in \alpha^I_C$ and $\forall v \in \theta_C(x)$, $t.x.v \in \sum^O_J$. If $\exists s \xrightarrow{g,t.x,r}_A s'$, then V$\models$ g $\wedge$ $s \xrightarrow{g,t.x.v,r}_J s'$ and $\exists q' \in S_C$, V$\models$ g, (q, s) $\xrightarrow{g,t.x.v,r}$(q', s') (because C is input-universal). Hence, $\eta_A \models \pi_C(q)$ and (q', g, s') $\in \Theta$.

Proof of necessity: Consider an alternating simulation relation $\prec$ between TDEC C and TI A, and (q, g, s) $\in S_\otimes$ be a state reachable via trace σ (σ be a trace of $L_\otimes$ from $s^0_\otimes$). Then we prove s $\neq\perp$, $\eta_J \models \pi_C(q)$, q $\prec$ s and $g_q \prec g_s$ by induction on the length of σ. First, when $\sigma = \lambda$, we know (L, U) $_{(\pi c\,(s0))}$ = (L, U) $_{(\pi j(s0))}$ = 0, (q, s) = (s^0_C, s^0_A) = $s^0_\otimes$. Hence s $\neq\perp$, $\eta_J \models \pi_C(q)$ (it means $L_{(\pi c\,(s))} \not\prec L_{(\pi j(s))}$, $U_{(\pi c\,(s))} \not\succ U_{(\pi J(s))}$), q $\prec$ s and $g_q \prec g_s$. Next, suppose s $\neq\perp \wedge \eta_J \models \pi_C(q) \wedge$ q $\prec$ s $\wedge$ $g_q \prec g_s$ hold for any σ and we considered that time-conditions of TDEC respects TIA time-condition. Since $S_J = S_A \cup \{\perp\}$, we know s $\in S_A$.

a) For e $\in \sum^H_C \wedge \vee \models$ g, if $\exists$(q, s) $\xrightarrow{g,e,r}_\otimes$ (q', s'), then s' = s (thus s' $\neq\perp$) and $q \xrightarrow{e}_C q'$. Since q $\prec$ s $\wedge$ $g_q \prec g_s \wedge$ (L_q and U_q) $\prec$ (L_s and U_s) , we can get q' $\prec$ s $\wedge$ $g_{q'} \prec g_s$. Thus s' $\neq\perp \wedge \eta_A \models \pi_C(q)$.

b) For *t.x.v* $\in \sum^O_C \wedge \vee \models$ g, if $\exists$(q, s) $\xrightarrow{g,t.x.v,r}_\otimes$ (q', s'), then q $\xrightarrow{g,t.x.v,r}_C$ q'. Since q $\prec$ s, $g_q \prec g_s$ and A is deterministic, s' $\in S_A$, $s \xrightarrow{g,t.x,r}_A$ s' and q' $\prec$ s'. Since we considered that time-conditions of TDEC respects TIA time-condition, then $g_{q'} \prec g_{s'}$ and ($L_{q'}$ and $U_{q'}$) $\prec$ ($L_{s'}$ and $U_{s'}$). Thus s' $\neq\perp \wedge \eta_A \models \pi_C(q)$.

c) For $t.x.v \in \sum^{O}_{J} \wedge \forall \models g$, if $\exists(q, s) \xrightarrow{g,t.x.v,r} \otimes (q', s')$, Then $s \xrightarrow{g,f.v,r}_{J} s'$, $g_{q'} \prec g_{s'}$ and $(L_{q'}$ and $U_{q'}) \prec (L_s$ and $U_s)$. Since $t.x \in \sum^{I}_{A}$, $s \xrightarrow{g,t.x,r}_{A} s'$ and $s' \neq \perp \wedge \cap_A \models \pi_C(q)$.

IV. CONCLUSION

In this paper, we have extended a theory for specificationand verification the consistency of component-based real-timed systems based on timed automata, timed interface automata and discrete event components. Since real time systems have critical processes, the framework which proposed should have certain characteristics that defined in section 1:

a. The work that presented in this paper has *simple and unambiguous definition* for specifying timed components that uses definition of timed automata [1], time interface automata [2] and discrete event component [3]. The components have a set of variables that simulates clock and when the transition started the clock variables increase whit the same speed and clock constraint are used to restrict the transition.

b. We developed a definition for *composition of two component and communication* between them, described by synchronization vector. Two components allowed combining if they have not any shared output.

c. For checking the consistency between TDEC and TIA, we extended the theory which presented in [3] which detected the local state space for the lack of unexpected states, also upper bound and lower bound of each timed transition in component should not exceed of corresponding transition in its timed interface.

Currently, we work on implementing tool based on our approach that can automatically verify consistency. To future work, we want to extend networks of both DECs and IAs defined in [2] and hierarchical components defined in [13] for component-based real-time systems.

V. REFERENCES

[1]. R. Alur and D. L. Dill,“A theory of timed automata,”Theor. Comput.Sci,1994, pp. 183-235.

[2]. L. de Alfaro, T. A. Henzinger, and M. I. A. Stoelinga,“Timed interfaces,” In A. L. Sangiovanni-Vincentelli and J. Sifakis, editors, EMSOFT, Springer , 2002, volume 2491 of LNCS, pp. 108-122.

[3]. jin Y, Lakos C andEsser R,“modular consistency analysis of component-based designs,” J Res PractInfTechnol, 2004, 36(3): 186-208.

[4]. jin Y, Lakos C and Esser R,“component-based design and analysis: a case study,” In: Software engineering and formal methods (SEFM), IEEE computer society press, Los Alamitos, 2003, pp. 126-135.

[5]. Rajeev Alur and David L. Dill, “Automata for modeling real-time systems,” In Proceedings, Seventeenth International Colloquium on Automata, Languages and Programming, Springer-Verlag, volume 443 of Lecture Notes in Computer Science, 1990, pp. 322–335.

[6]. de Alfaro L and Henzinger T, “Interface Automata,” In: proceedings of the joint 8th European software engineering (ESEC/FSE-10), 5 of software engineering notes, ACM press, vol 26, new York,septeber 10-14 2001, pp. 109- 120.

[7]. de Alfaro L and Henzinger T,“Interface-Based Design’” In: marktoberdorf summer school, Kluwer, aprril 12-15 2004, pp. 50-56.

[8]. A. Arnold,“Finite transition systems - Semantics of communicating processes,” Prentice Hall,1994.

[9]. R. Alur and T. A. Henzinger,“Reactive modules,” Formal Methods in System Design: AnInternational Journal, July 1999, pp. 7–48.

[10]. Johan Bengtsson and Wang Yi,“Timed Automata: Semantics, Algorithms and tools,” Uppsala University.

[11]. Jan Tretmans,“Model Based Testing With Labeled Transition Systems,”Embedded Systems Institute, EindhovenAnd Radboud University, Nijmegen,The Netherlands, Jan.Tretmans@Esi.Nl.

[12]. A. Arnold Labri,“Automatic Verification Of Properties In Transition Systems,” Universit´E Bordeaux I, 351, Cours De La Lib´Eration, F-33405 TalenceCedex, FranceANDS. BRLEKLacim, Universit´E Du Qu´Ebec `A Montr´Eal, P.O. Box 8888, Succursale “Centre Ville”, Montr´Eal, (QC) Canada.

[13]. Ayaz. Isazadeh, Jaber. Karimpour, “A new formalism for mathematical description and verification of component-based systems,” Springer, 2 October 2008, pp. 334-353, DOI 10.1007/s11227-008-0240-y.

[14]. Szyperski C, “Component Software: beyond object-oriented programming,” 2nd end. Addition Wesley, reading, 2000.

[15]. R. Gerth, D. Peled, M. Vardi, and P, Wolper,“Simple on-the-fly automatic verification of linear temporal logic,” In Protocol Specification Testing and Verification, Warsaw, Poland, 1995,pp. 3–18.

[16]. K. G. Larsen and L. Xinxin,“Compositionality through an operational semantics of contexts,” In M. Paterson, editor, Proceedings of the 17th International Colloquium on Automata, Languages, and Programming (ICALP’90), Springer-Verlag, LNCS 443, 1990, pp. 526–539.

[17]. Kupferman, O, Vardi and M.Y,“Modular Model Checking,” In: Compositionality: The SignificantDifference. International Symposium, COMPOS’97, Bad Malente, Germany, Springer-Verlag, Volume 1536 of LNCS, 1998, pp. 381–401.

[18]. G. Winskel,"on the compositional checking of validity," In J. C. M. Baeten and J. W. Klop,editors, Theories of Concurrency: Unification and Extension (CONCUR'90), LNCS 458, Springer-Verlag, 1990.

[19]. Andreas Fredriksson,"Component-Based Systems," Development, basic concept, Växjö University, Department of mathematics, statistics, and computer science Informatics, Spring 99.

[20]. Philip A. Laplante and Seppo J. Ovaska,"Real time systems design and analysis: Tools for the practitioner," the institute of electrical and electronics engineers,

[21]. S. Graf and H. Saïdi,"Construction of abstract state graphs with PVS," Springer Verlag, 1997.

[22]. H. Saïdi, "Model checking guided abstraction and analysis," In Proceedings of the 7th International Static Analysis Symposium (SAS'00), LNCS 1824, 2000.