2011 American Control Conference
on O'Farrell Street, San Francisco, CA, USA
June 29 - July 01, 2011

# Dynamic Discrete-Event Systems with Instances for the Modelling of Emergency Response Protocols

Lenko Grigorov and Karen Rudie

*Abstract*— **A type of model, dynamic discrete-event systems with instances, is proposed for use with time-varying systems where a number of system components may share common behavior. The motivation for this model comes from the area of emergency response protocols where systems may change unexpectedly, however, there are predefined types of actors. The model is based on previous work on dynamic discrete-event systems and on template design. Synchronization patterns are introduced which allow the generic definition of component interactions regardless of a particular system composition. The synchronous product operation is updated in order to employ these patterns. Advantages of the proposed model include compactness of representation, preservation of the identity (history) of individual components and, with the use of online control, amenability to dynamic changes of control specifications.**

## I. Introduction

The application of discrete-event system (DES) control theory is faced with many difficulties, including the demands placed on users in terms of familiarity with the theory. In our previous research, we proposed a methodology where certain aspects of DES modelling can be simplified with the use of templates [1]. Recently, in cooperation with the Kingston, Frontenac and Lennox & Addington Public Health Unit, we have been interested in the application of DES supervisory control to the management of emergency situations, e.g., disease outbreaks [2], [3].

Emergency situations are usually volatile and unpredictable. In a recent forest-fire training exercise at the Frontenac County, Ontario, for example, trainees had to respond to an unexpected fire expansion into the area of a densely populated township. A disease outbreak scenario could serve as another illustration: new patients may appear at any time, the time of recovery cannot be determined precisely for each individual, and health-care workers may succumb to the disease and turn into patients themselves.

It is clear that in such situations models of static systems, composed of pre-determined components, cannot be used. A common approach then is to use what we will call *aggregating* models—such as Petri nets, [4], and vector DES, [5]. Aggregating models ignore the individuality of components of the system and simply count how many components are in each specific state, allowing for a dynamic variation of the number of components with time, for example. Such models allow for a great simplification of the state space and the control solutions. We will argue, however, that such an approach is not always suitable, especially in modelling emergency situations. The main drawback of aggregating models is the fact that information about the past behavior of individual components is discarded—and dynamically updated control specifications may not access it. Let us consider a disease outbreak scenario and let us assume that, upon diagnosis, patients get treated with some medication. As it might be, the lack of recovery in some patients prompts an internal investigation of the treatment procedure. Two possible outcomes (and for which numerous examples exist in real life) are that a batch of medication used for treatment was ineffective (e.g., due to faulty manufacture) or that one of the nurses made a systematic error by administering a different medication (e.g., due to a confusion about the name of the medication). Note that an investigation such as the one described already requires detailed information about the treatment of patients: when, how, by whom. More importantly, however, the results of such an investigation would necessitate a change in the control specifications, i.e., patients treated at a given time or by a specific nurse will need to be treated again. If the history of individual components is discarded, as in aggregating models, such specification updates are not possible.

In this work we propose a new type of model, *dynamic discrete-event system with instances* (DDES-I), which is based on the ideas from our investigation of modelling emergency response protocols. We take advantage of the dynamic discrete-event system (DDES) framework from [6] and the idea of instantiating template models from [1]. A set of *roles* are defined for the actors in an emergency scenario. Then, the model of each actor is an *instance* of one of the role models. The DDES consists of a set of actors which may vary as time advances. *Synchronization patterns* are also introduced in order to define how actors interact. The DDES-I model

- is a compact description similar to aggregating models,
- supports the modelling of dynamic systems and
- preserves the identity and history of modules.

## II. Preliminaries

Before introducing the DDES-I model, let us first review some concepts and notation which will be used at different points in the rest of the paper. The review is very brief due to the limited space. More details are available in the references.

A DES is a system where events, from a finite alphabet $\Sigma$, occur instantaneously and asynchronously. The collection of all strings of events that can potentially occur in the system forms the language generated by it. If the language is regular, it can be modelled as a finite-state automaton, $G =$

L. Grigorov and K. Rudie are with the Department of Electrical and Computer Engineering, Queen's University, Kingston, ON, Canada lenko.grigorov@banica.org

978-1-4577-0079-8/11/$26.00 ©2011 AACC

$(\Sigma, Q, \delta, q_0, Q_m)$. Then, the total behavior (the generated language) is denoted $L(G)$ and the set of completed tasks (the accepted language) is denoted $L_m(G)$. The overall system may be defined as the parallel composition of a number of modules, $G = \|G_i$ for $i$ in some finite index set, where $\|$ denotes the synchronous product operation [7]. For a language $L$, a string $s$ and $n \in \mathbb{N}$, let us define the notation $L/s = \{t \,|\, s.t \in L\}$ and $L|_n = \{t \,|\, t \in L,\, |t| = n\}$.

Some of the modules in a composed system may share a common pattern of behavior, e.g., all nurses in a unit may have similar duties. Thus, it is possible to describe modules as particular *instances*, $G_i^p$, of some *template* module, $G^p$, [1]. Instantiation simply creates a copy of the template model and indexes events to differentiate the instance from other instances. If $G^p = (\Sigma, Q, \delta, q_0, Q_m)$ is a template, then the instance with index $i$ is $\mathbf{Ins}(G^p, i) = (\Sigma_i, Q, \delta_i, q_0, Q_m)$, where

$$\Sigma_i = \{\sigma_i | \sigma \in \Sigma\},$$
$$\delta_i(q, \sigma_i) = \delta(q, \sigma).$$

Specifications for the control of a DES are given as $K \subseteq L_m(G)$, called the *legal* language. In supervisory control, [7], the controller can be described as a function $\gamma : \Sigma^* \to 2^\Sigma$ such that for every string in $L(G)$, it defines which events are allowed to occur next. If proper limits on the occurrences of events are chosen, then the controller would be able to ensure that the controlled behavior is contained in $K$.

A dynamic DES (DDES) is a system defined as the composition of modules, where the set of modules may vary with time, [6]. Thus, if $t$ stands for time and $M(t) = \{M_{1t}, M_{2t}, \ldots, M_{nt}\}$ is the set of modules at time $t$,

$$G(t) = \|M(t).$$

The control of a DDES requires the use of *online* strategies as the system varies with time, i.e., instead of precomputing $\gamma$ for all possible $s$, the control function is computed as the system evolves. In [8], the authors propose the use of a limited lookahead projection of the system behavior to compute $\gamma$. For a selected window size of $N$, a tree is constructed such that it contains all possible future $N$-step evolutions of the system. The control decisions are then chosen so that evolutions not contained in the legal language are avoided. The tree construction and decision-making are repeated after each event occurrence.

## III. Dynamic DES with instances

When modelling disease outbreak control protocols, and emergency response protocols in general, it is not practical to create a separate model for every potential actor. Indeed, all patients (or, respectively, nurses, visitors, etc.) share the same underlying model behavior. We will call this behavior a *role*. If the identities of actors are not important, it is sufficient to keep track of how many actors are in each possible state of the role, e.g., through aggregating models such as vector DES or Petri nets. This approach has been assumed, for example, in other work at our laboratory, [2], [3], or in the area of patient flows and health-care resource allocation [9]. However, as already discussed in Section I, the identity of actors cannot always be ignored.

In order to support the modelling of emergency response protocols where actor identities need to be preserved, we propose the use of an extension of the DDES model where actor models are obtained through the instantiation of role models. Indeed, roles can be viewed as templates in the context of template design, [1], and the same instantiation procedure can be used. The approach also naturally takes advantage of the ability to describe time-varying systems as DDESs; emergency situations are usually dynamic and it is not possible to design the system *a priori*.

In this paper we will have a running example to illustrate the introduced concepts. We will consider a disease outbreak scenario which is inspired by a real protocol for the control of respiratory disease outbreaks in a long-term care home [10]. However, for the purposes of clarity and brevity, the situation has been greatly simplified and only a small part is considered.

### A. Roles and instances

Let $\mathcal{R} = \{R^1, \ldots, R^l\}$, $l \in \mathbb{N}$ be a collection of roles, where each role is expressed as some finite-state automaton $R^j = (\Sigma^j, Q^j, \delta^j, q_0^j, Q_m^j)$. Each role describes the prototypical behavior of a type of actors. In a disease outbreak response scenario these could be "long-term care home", "doctor", "nurse", "patient", "Ministry of Health", etc. It is understood that for institutions, such as "Ministry of Health", there will be only one actor with the given role. The models of actors are obtained by instantiating the roles, in the same way that templates are instantiated in [1]. The instance with index $k$ of the role $R^j$ is defined as

$$R_k^j = \mathbf{Ins}(R^j, k) = (\Sigma_k^j, Q^j, \delta_k^j, q_0^j, Q_m^j).$$

In Section I, we mentioned that the proposed model is compact. Indeed, it is not necessary to actually create separate models for every instance. The state space of an instance is, by definition, identical with the state space of its role. Thus, it is sufficient to keep track of the indices of instances and of which instances are in which states.

Let $\mathcal{R}$ be a set of roles. The set of instances is defined as

$$\mathcal{A} = \{(i_1, j_1), \ldots, (i_n, j_n)\},$$

where $n \in \mathbb{N}$, $\forall a \in \mathcal{A} : a \in \mathbb{N} \times \{j \,|\, R^j \in \mathcal{R}\}$ and if $(i, j)$, $(i, j') \in \mathcal{A}$, then $j = j'$. In other words, each instance is a pair containing the (unique) instance index and the index of the associated role. If $(i, j) \in \mathcal{A}$, then the model for the instance would be $R_i^j = \mathbf{Ins}(R^j, i)$. We can also define three functions for convenience:

$$\iota\left((i, j)\right) = i$$

is the index retrieval function,

$$\rho\left((i, j)\right) = j$$

is the role retrieval function, and

$$\mathrm{Idx}(\mathcal{A}) = \{i \,|\, (i, j) \in \mathcal{A}\}$$

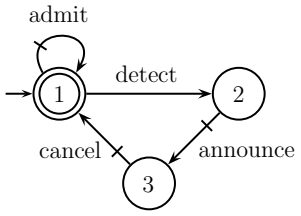


Fig. 1. The model for actors with the role of a long-term care home. As this is an institution, there will be only one actor with this role.

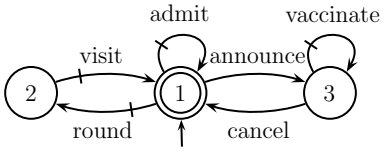


Fig. 2. The model for actors with the role of a nurse.

is a function which returns the indices of all instances in $\mathcal{A}$.

There are three roles which we will consider in our example: the long-term care home, the health-care workers there (whom we will refer to as "nurses"), and the patients. Let us denote the models for each role $R^H$, $R^N$ and $R^P$, respectively. Thus, we have $\mathcal{R} = \{R^H, R^N, R^P\}$. In reality, of course, one would need to consider many more roles. The models for the three roles are shown in Figs. 1 to 3. The events used in the models are as follows:

admit Admit a patient to the facility.
announce Announce a disease outbreak state.
cancel Cancel the state of disease outbreak.
detect Detect a disease outbreak.
round Start a patient visitation round.
vaccinate Vaccinate a patient.
visit Visit a patient.

Naturally, there is only one actor with the role $R^H$; let the instance index be 1 (and we have the instance model $R^H_1$). Let us say that there are two nurses with instance indices 2 and 3 (models $R^N_2$ and $R^N_3$), and five patients with the instance indices 4 to 8 (models $R^P_4$ to $R^P_8$). An illustration of an instance model (for a patient) is shown in Fig. 4.

With the above selection, $\mathcal{A} = \{(1, H),\ (2, N),\ (3, N),\ (4, P),\ (5, P),\ (6, P),\ (7, P),\ (8, P)\}$.

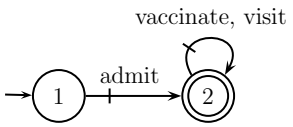


Fig. 3. The model for actors with the role of a patient.

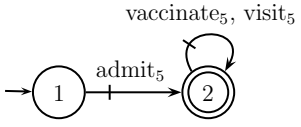


Fig. 4. The model for the actor with the role of a patient and index 5.

### B. Synchronization patterns

The consideration of instances introduces a problem: the actor models can no longer synchronize via common events as usually done in modular systems (i.e., using synchronous product). In our example, both the $R^N$ and $R^P$ models contain the event "vaccinate", meaning that a nurse vaccinates a patient. However, the models of the instances with indices 2 and 4, $R^N_2$ and $R^P_4$, will have events with different names: "vaccinate$_2$" and "vaccinate$_4$", respectively. The naming issue could be resolved, in this particular situation, with the use of appropriate event maps. However, when one considers the rest of the nurses and patients, and the fact that the number of instances may vary, the situation becomes much more complex. Naturally, then, simple event maps will not be able to describe the desired synchronization. Here, we propose four synchronization patterns which are suitable for the definition of interactions between instance models: "all", "many", "any" and "one". These patterns will be explained next.

Let $\Sigma^{\mathcal{R}} = \bigcup_{R^j \in \mathcal{R}} \Sigma^j$ signify the set of all events in the models of roles. Consequently, for every $\sigma \in \Sigma^{\mathcal{R}}$ there is the non-empty set $\mathcal{R}(\sigma) = \{R^j \in \mathcal{R} | \sigma \in \Sigma^j\}$ of roles that contain the event. Then, for each event $\sigma \in \Sigma^{\mathcal{R}}$ we define the synchronization pattern

$$\pi(\sigma) = \{(j, \Diamond) | R^j \in \mathcal{R}(\sigma)\} \text{ or } \emptyset$$

where $\Diamond$ stands for one of the following:

all All instances of the given role must participate in the synchronization.
many All available instances, but at least one, of the given role must participate in the synchronization. Here "available instances" means instances which are in a state where the event can occur.
any All available instances, if any, of the given role must participate in the synchronization.
one Exactly one instance of the given role must participate in the synchronization.

The empty pattern, $\pi(\sigma) = \emptyset$, means that no synchronization is associated with the event. In particular, events local to the instances will have the empty pattern.

Thus, if we go back to our example with the nurse and patient roles and the "vaccinate" event, one could specify the following pattern:

$$\pi(\text{"vaccinate"}) = \{(N, \text{"one"}), (P, \text{"many"})\},$$

where, as before, $N$ is the subscript for the nurse role and $P$ is the subscript for the patient role. This pattern signifies that, when actors synchronize on the event "vaccinate", exactly one nurse instance will synchronize with at least one but potentially many patient instances—thus a single nurse will vaccinate all the patients which can be vaccinated. An alternative pattern could be

$$\pi(\text{"vaccinate"}) = \{(N, \text{"one"}), (P, \text{"one"})\},$$

which means that a single nurse will vaccinate a single patient (and, if more patients can be vaccinated, the event will have to occur multiple times).

In our example, the following synchronization patterns will be used:

$$\begin{aligned}
\pi(\text{“admit”}) &= \{(H,\text{“one”}),(N,\text{“one”}),(P,\text{“many”})\}\\
\pi(\text{“announce”}) &= \{(H,\text{“one”}),(N,\text{“all”})\}\\
\pi(\text{“cancel”}) &= \{(H,\text{“one”}),(N,\text{“all”})\}\\
\pi(\text{“detect”}) &= \emptyset\\
\pi(\text{“round”}) &= \{(N,\text{“one”})\}\\
\pi(\text{“vaccinate”}) &= \{(N,\text{“one”}),(P,\text{“one”})\}\\
\pi(\text{“visit”}) &= \{(N,\text{“one”}),(P,\text{“any”})\}
\end{aligned}$$

Note that classical synchronization on $\sigma$, if we ignore event indices, can be defined as

$$\pi(\sigma) = \{(j,\text{“all”}) | R^j \in \mathcal{R}(\sigma)\}$$

and, if event indices are not ignored (i.e., $\sigma$ is local for each instance), as

$$\pi(\sigma) = \emptyset.$$

Furthermore, the prioritized synchronization from [11] can also be expressed. Let $A$ and $B$ be, correspondingly, the priority sets of $R^{j1}$ and $R^{j2}$. If $\sigma \in A \cap B$, then $\sigma$ should be classically synchronized (or *strictly* synchronized in the authors' terminology). If $\sigma \in A \setminus B$ then we can use the pattern

$$\pi(\sigma) = \{(j1,\text{“all”}),(j2,\text{“any”})\}$$

(and a similar pattern for $\sigma \in B \setminus A$). As our focus is on the synchronization of instances, we did not explore this topic further, but it is conceivable that some of the other synchronization patterns introduced in [12] can also be expressed using our approach.

Multi-agent (MA) product, as proposed in [13], also discusses systems composed of individual modules. It generalizes the notion of state transition to allow for the simultaneous occurrence of a number of events. Thus, the operations of multiple actors on different tasks in parallel can be modelled—something not possible with our approach. However, MA product does not seem to be suitable for dynamic systems. The vector space is of fixed dimension (fixed number of modules) and it is not possible to express module synchronization other than the classical one. Furthermore, the model representation is not compact as it does not make use of templates (or prototypes) for the system components.

### C. Synchronous product of instances

If a system is given as a collection of roles, instances, and synchronization patterns, it is still necessary to be able to determine the overall behavior of the global system. The synchronous product operation is used for this purpose when a system consists of usual modules. However, a new composition operation is needed to take advantage of the proposed information structures. Let us call this operation the *synchronous product of instances*, denoted $\|_{spi}$. The definition is given next.

Let $\mathcal{R}$ be a set of role models, $\pi$ be the associated synchronization patterns, and $\mathcal{A} = \{a1,\ldots,an\}$ be a set of $n$ instances.

Then

$$\|_{spi}(\mathcal{R},\pi,\mathcal{A}) = (\Sigma^{spi}, Q^{spi}, \delta^{spi}, q_0^{spi}, Q_m^{spi}),$$

where the elements $Q^{spi}$, $q_0^{spi}$ and $Q_m^{spi}$ are defined as expected, i.e., the product of the states of the components:

$$\begin{aligned}
Q^{spi} &= Q^{\rho(a1)} \times \ldots \times Q^{\rho(an)}\\
q_0^{spi} &= (q_0^{\rho(a1)},\ldots,q_0^{\rho(an)})\\
Q_m^{spi} &= Q_m^{\rho(a1)} \times \ldots \times Q_m^{\rho(an)}.
\end{aligned}$$

Let $\sigma \in \Sigma^{\mathcal{R}}$, $q = (q_1,\ldots,q_n) \in Q^{spi}$. The transition function is defined as follows:

$$\delta^{spi}(q,\sigma_{\{i1,\ldots,il\}}) = \begin{cases}
(q_1,\ldots,\delta_{i1}^{j1}(q_{i1},\sigma_{i1}),\ldots,q_n)\\
\qquad \text{if } \pi(\sigma)=\emptyset, l=1 \text{ and}\\
\qquad \delta_{i1}^{j1}(q_{i1},\sigma_{i1}) \text{ is defined}\\
(q_1,\ldots,\\
\delta_{i1}^{j1}(q_{i1},\sigma_{i1}),\ldots,q_\upsilon,\ldots,\\
\delta_{il}^{jl}(q_{il},\sigma_{il}),\ldots,q_n)\\
\qquad \text{if } \pi(\sigma)\neq\emptyset \text{ and } \textbf{Condition}\\
\text{undefined otherwise,}
\end{cases}$$

where $\upsilon \in [1,n]$, $i1$ to $il$ denote the indices and $j1$ to $jl$ denote the roles of some set of instances $\{ak1,\ldots,akl\} \subseteq \mathcal{A}$ (i.e., $\forall x \in \{1,\ldots,l\} : ix = \iota(akx)$ and $jx = \rho(akx)$), and the condition **Condition** is defined as follows:

1) $\forall ix \in \{i1,\ldots,il\} : \delta_{ix}^{jx}(q_{ix},\sigma_{ix})$ is defined;
2) $\forall (i,j) \in \mathcal{A}, (j,\text{“all”}) \in \pi(\sigma) : i \in \{i1,\ldots,il\}$;
3) if $\exists j, (j,\text{“many”}) \in \pi(\sigma)$ or $(j,\text{“one”}) \in \pi(\sigma)$, then $\exists ix \in \{i1,\ldots,il\} : jx = j$;
4) $\forall (i,j) \in \mathcal{A}, (j,\text{“many”}) \in \pi(\sigma)$ or $(j,\text{“any”}) \in \pi(\sigma)$ where $\delta_i^j(q_i,\sigma_i)$ is defined: $i \in \{i1,\ldots,il\}$;
5) if $\exists ix, iy \in \{i1,\ldots,il\}, jx = jy$ and $(jx,\text{“one”}) \in \pi(\sigma)$, then $ix = iy$.

The interpretation of the parts of the condition is as follows:

1) This part simply states that all transitions that take place are defined in the corresponding instances.
2) If there is an instance for which on the event $\sigma$ the synchronization pattern is “all”, the instance must be making a transition in the composed system.
3) If there is a “many” or “one” synchronization pattern for the event $\sigma$, there must be at least one instance (with the corresponding role) making a transition on the event.
4) All instances for which on the event $\sigma$ the synchronization pattern is “many” or “any” and which can make a transition on $\sigma$ must participate in the transition.
5) This part states that there cannot be two different instances with the same role which participate in the transition when the synchronization pattern is “one”.

The event set of the composition is then defined in terms of the transitions in the model, i.e.,

$$\Sigma^{spi} = \{\sigma_I | \exists q \in Q^{spi} : \delta^{spi}(q,\sigma_I) \text{ is defined}\},$$

where $I \subseteq \text{Idx}(\mathcal{A})$ denotes a set of instance indices.

In our example, the complete composition is too large to show in this paper, however, it is possible to illustrate the construction of the transition function $\delta^{spi}$ from a single state. Let us assume that the actors are in the following states: the long-term care home is in state 1, nurse 2 is in state 1, nurse 3 is in state 2, patients 4 to 6 are in state 2 and patients 7 and 8 are in state 1. Then, the composed state of interest is $q = (1, 1, 2, 2, 2, 2, 1, 1)$ and the transitions from $q$ are as follows:

$$\begin{aligned}
\delta^{spi}(q, \text{“admit}_{\{1,2,7,8\}}\text{”}) &= (1, 1, 2, 2, 2, 2, 2, 2) \\
\delta^{spi}(q, \text{“detect}_{\{1\}}\text{”}) &= (2, 1, 2, 2, 2, 2, 1, 1) \\
\delta^{spi}(q, \text{“round}_{\{2\}}\text{”}) &= (1, 2, 2, 2, 2, 2, 1, 1) \\
\delta^{spi}(q, \text{“visit}_{\{3,4,5,6\}}\text{”}) &= (1, 1, 1, 2, 2, 2, 1, 1)
\end{aligned}$$

Transitions on “announce$_I$”, “cancel$_I$” and “vaccinate$_I$” are not defined for any $I \subseteq \text{Idx}(\mathcal{A})$.

Note that the use of synchronization patterns is not necessary if the set of actors is static and known *a priori*. Then, with suitable event masks, the desired synchronization can be accomplished. However, as already discussed, we envision the use of this modelling methodology in dynamic systems, where the number of instances of roles may change unexpectedly as time advances. Then, static event masks cannot be used. Synchronization patterns, in essence, form dynamic event masks which can adapt to system changes.

The reader may also observe that the computation of potential transitions is non-trivial. Especially when the pattern “one” is used and there are a number of instances which could synchronize, there could be a high-order polynomial number of transitions. This is, indeed, the nature of the systems considered in this paper (or, one could argue, the nature of DES systems in general). Under a limited-lookahead control strategy, however, the system size would pose less of a problem as only a portion of the full system would be considered at a time.

With all definitions in place, we can finally proceed with the formal definition of *dynamic discrete-event systems with instances*, DDES-I. Let $t$ stand for time, let $\mathcal{R}$ be a set of roles, $\pi$ be the associated synchronization patterns, and let the set of role instances be time-varying, $\mathcal{A}(t)$. Then DDES-I is defined as

$$\begin{aligned}
G(t) &= \|_{spi}(\mathcal{R}, \pi, \mathcal{A}(t)) \\
&= (\Sigma_t^{spi}, Q_t^{spi}, \delta_t^{spi}, q_{0t}^{spi}, Q_{mt}^{spi}).
\end{aligned}$$

Without loss of generality, it can be assumed that in the life span of the system instance indices cannot be reused: for all time instances $t' < t'' < t'''$ such that $i \in \text{Idx}(\mathcal{A}(t'))$ and $i \notin \text{Idx}(\mathcal{A}(t''))$, then $i \notin \text{Idx}(\mathcal{A}(t'''))$.

Furthermore, for the purpose of consistency, the current states of instances are preserved as time advances. For all consecutive time instances $t' = t + 1$ where time advances on the occurrence of the event $\sigma_I$: if $ak = (i, j) \in \mathcal{A}(t)$ and $ak' = (i, j) \in \mathcal{A}(t')$, at time $t$ the current state of the system was $q = (q_1, \ldots, q_k, \ldots, q_n)$, at time $t'$ the current state of the system was $q' = (q'_1, \ldots, q'_{k'}, \ldots, q'_{n'})$, and $\delta_t^{spi}(q, \sigma_I) = (q''_1, \ldots, q''_k, \ldots, q''_n)$, then $q'_{k'} = q''_k$.

In our example, as time advances, there could be variations in the number of patients and nurses.

Most of the ideas introduced with DDES-I, such as roles, instance identities and interaction histories, can be expressed also in colored Petri net models, [14]. Colored Petri nets form an extension to Petri nets where the basic net structure of places and transitions is complemented with the expressiveness of a programming language. The notion of tokens is replaced by the concept of data types (of possibly non-finite range) and values, where each token can carry an identifier and, potentially, its transition history. However, with our simplified approach, we maintain focus on the essential properties we would like to model: dynamic behavior, interaction and history, and avoid the unnecessary use of highly-expressive models. As discussed next, DDES-I models can take advantage of the existing techniques developed for online control.

## IV. Control of DDES-I

In this section, due to space limitations, we will provide only a brief overview of the topic of DDES-I control.

Dynamic DESs with instances are time-varying models and the classical approach of offline synthesis of supervisory controllers is not applicable. In [6], the limited-lookahead online control proposed by Chung *et al.* in [8] was applied to DDES. Thus, the same method was used for DDES-I.

The limited-lookahead controller relies on the availability of a partial view of the system. More specifically, a description of the potential behavior of the system over a limited period in the future is considered. In [8], the limited view of the system is provided by the function $f^N_{L(G)}$ defined over the strings $s$ in a given language $L(G)$ as follows: $f^N_{L(G)}(s) = (L(G)/s|_N, L_m(G)/s|_N)$, where $N$ is the depth of the look-ahead window. In this paper, we will use a slightly modified version of this function, where the history of event occurrences is preserved; this allows us to consider specifications which are not state-based. Given a DDES-I $G$, we can define $f^N_{L(G)}$ as follows:

$$f^N_{L(G)}(s) = (\,\{s\}.\,(L(G(t_s))|_N)\,,\ \{s\}.\,(L_m(G(t_s))|_N)\,)\,,$$

where $t_s$ is the time right after the string $s$ has occurred.

### A. Control specifications

Online control provides greater flexibility in defining control specifications. For example, they can be a function of time, of past performance and/or of current system composition. For example, in [15], the authors propose a framework where the supervisor may choose to switch between a number of control policies during runtime.

In limited-lookahead control, [8], the authors propose the use of a function, $f^N_K$, to specify which subset of the strings in the limited-lookahead window are desirable and which ones are not. In other words, given the legal language $K$, and a string $s$, $f^N_K \circ f^N_{L(G)}(s) = (\overline{K}/s|_N,\ K/s|_N)$.

Here we use a modified version of this function in order to enable control strategies for DDES, e.g., as proposed in [6]. The differences are the following:

1) the history of event occurrences, the string $s$, is considered and
2) for each node (state) in the lookahead tree, the function determines a desirability value, rather than a simple dual evaluation ("legal" or "illegal").

Thus, we have

$$f_K^N \circ f_{L(G)}^N(s) = (f_{L(G)}^N(s),\, v)$$

where $v : \Sigma^* \to \mathbb{R}$ is an evaluation of every prefix of the strings in $f_{L(G)}^N(s)$. As noted in [6], the classical separation of strings as "legal" or "illegal" can be accomplished simply by returning one of two values as the evaluation of strings, e.g., 0 and $-\infty$.

With DDES-I, a controller may take advantage of additional information which is not normally available with other models, i.e.:

- the identity of modules executing events,[1]
- the role (or type, class) of individual modules and
- the interactions of modules (in the composite indices of events).

Examples of the kinds of specifications which can be expressed in terms of the above properties include

- The occupation of nurses must be fair, i.e., within a time interval, there should not be a discrepancy larger than one in the number of tasks a nurse has completed.
- Nurses should attend to the same patients, i.e., nurses are assigned patients.
- All actors with the role "visitor" are informed about the regulations on patient visits.
- All actors who have interacted with the patient with instance index 7 must undergo a screening procedure.

It is important to note that, in contrast to classical supervisory control and due to the nature of the dynamic systems considered, a controller for DDES-I might not be able to guarantee the achievement of the control goals under all circumstances. For example, if all nurses walk out of a hospital (i.e., all instances of the role "nurse" disappear from the system), it might not be possible to avoid a deadlock. However, static analysis and simulation runs of a given emergency response protocol might give the insight needed to modify the protocol so that undesired conditions could be avoided.

## V. Conclusions

In this work we described a type of compact DES model, dynamic DES with instances, designed for use with time-varying systems consisting of individual components with shared behavior. The motivation for DDES-I comes from our ongoing research in the area of emergency response protocols, such as protocols for the control of infectious disease outbreaks. An emergency situation may evolve unexpectedly with time, however, the actors in such a scenario usually have common, predefined roles. The DDES-I models support the definition of synchronization patterns for a time-varying number of modules and, together with online control, the use of dynamic specifications. Modelling can be done naturally and in a compact form.

Future work can focus on a number of aspects which are not addressed fully in the proposed model. These include: a formal framework for the definition of control specifications which takes advantage of the information available with DDES-I, the development of synchronization patterns which depend on past instances of synchronization, and investigating if it is possible to incorporate some of the ideas of multi-agent product, [13], within DDES-I.

[1]This information is not available in aggregating models such as Petri nets and vector DES.

## VI. Acknowledgments

We would like to thank Emergency Management Ontario, Frontenac County (Ontario), City of Kingston (Ontario) and the Kingston, Frontenac and Lennox & Addington Public Health Unit for all the help during our investigation of emergency management. Input from our colleagues at our research laboratory also helped in shaping this paper. This work was supported by a grant from NSERC, Canada.

## References

[1] L. Grigorov, B. E. Butler, J. E. R. Cury, and K. Rudie, "Conceptual design of discrete-event systems using templates," To appear in *Journal of Discrete-Event Dynamic Systems*, 2011.

[2] T. Brunsch and K. Rudie, "Discrete-event systems model of an outbreak response," in *Proceedings of the 2008 American Control Conference*, Seattle, WA, USA, June 2008, pp. 1709–1714.

[3] S.-J. Whittaker, K. Rudie, J. McLellan, and S. Haar, "Choice-point nets: a discrete-event modelling technique for analyzing health care protocols," in *Proceedings of the 47th Annual Allerton Conference on Communication, Control and Computing*, Monticello, IL, USA, 2009, pp. 652–659.

[4] J. O. Moody and P. J. Antsaklis, *Supervisory Control of Discrete Event Systems Using Petri Nets*. Kluwer Academic Publishers, 1998.

[5] Y. Li and W. M. Wonham, "Control of vector discrete-event systems *I*—the base model," *IEEE Transactions on Automatic Control*, vol. 38, no. 8, pp. 1214–1227, 1993.

[6] L. Grigorov and K. Rudie, "Near-optimal online control of dynamic discrete-event systems," *Journal of Discrete-Event Dynamic Systems*, vol. 16, no. 4, pp. 419–449, 2006.

[7] W. M. Wonham, "Supervisory control of discrete-event systems," Available at http://www.control.toronto.edu/DES/, June 2008.

[8] S.-L. Chung, S. Lafortune, and F. Lin, "Limited lookahead policies in supervisory control of discrete event systems," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1921–1935, 1992.

[9] E. D. Gunes and H. Yaman, "Modeling change in a health system: Implications on patient flows and resource allocations," *Clinical and Investigative Medicine*, vol. 28, no. 6, pp. 331–333, 2005.

[10] *A Guide to the Control of Respiratory Infection Outbreaks in Long-Term Care Homes*, Public Health Division and Long-Term Care Homes Branch, Ministry of Health and Long-Term Care, Ontario, October 2004.

[11] R. Kumar and M. Heymann, "Masked prioritized synchronization for interaction and control of discrete event systems," *IEEE Transactions on Automatic Control*, vol. 45, no. 11, pp. 1970–1982, 2000.

[12] V. Chandra, Z. Hunag, W. Qui, and R. Kumar, "Prioritized composition with exclusion and generation for the interaction and control of discrete event systems," *Mathematical and Computer Modeling of Dynamical Systems*, vol. 9, no. 3, pp. 255–280, 2003.

[13] I. Romanovski and P. E. Caines, "On the supervisory control of multi-agent product systems," in *Proceedings of the 41st IEEE Conference On Decision and Control*, Las Vegas, NV, USA, December 2002, pp. 1181–1186.

[14] K. Jensen and L. M. Kristensen, *Coloured Petri Nets*. Springer, 2009.

[15] C. Winacott and K. Rudie, "Limited lookahead supervisory control of probabilistic discrete-event systems," in *Proceedings of the 47th Annual Allerton Conference on Communication, Control and Computing*, Monticello, IL, USA, 2009, pp. 660–667.