INTERNATIONAL JOURNAL OF INTELLIGENT CONTROL AND SYSTEMS
VOL. 12, NO. 1, MARCH 2007, 15-23

# Resource-Constrained and Decision Support Workflow Modeling

William TEPFENHART, Jiacun WANG, Daniela ROSCA and Anni TSAI

**Abstract-In our previous work, we introduced WIFA approach for incident command systems workflow modeling and analysis. In this paper, we extend WIFA to take resources into account when modeling and enacting workflows. Resources can become important decision factors when combined with control flow information. In many situations, business processes are constrained by scarce resources. The lack of resources can cause contention, the need for some tasks to wait for others to complete, and the slowing down of the accomplishment of larger goals. This is particularly true in an emergency response system where large quantity of resources, including emergency responders, ambulances, fire trucks, medications, food, clothing, etc., are required. Often potential delays can be avoided or reduced by using resource analysis to identify ways in which tasks can be executed in parallel, in the most efficient way. Currently, during a time of crisis, a decision maker often concentrates on a single criterion in order to simplify, speed up or control the decision process itself. A resource-constrained workflow model can support the decision process by analyzing multiple criteria. It can keep track of resources availability, disable the path that is not executable, and present all executable paths, allowing the emergency responders to make decisions and implement them more confidently.**

***Index Terms*—Workflows, resource constraints, decision support, incident command systems.**

## 1. INTRODUCTION

Incident command systems (ICS) resulted from the need for a new approach to the problem of managing rapidly moving wildfires in the early 1970s [14]. At that time, emergency managers faced a number of problems, such as too many people reporting to one supervisor, lack of reliable incident information, and inadequate and incompatible communications. ICS is now widely used throughout the United States by fire agencies, and is increasingly used by law enforcement and many other public safety organizations for emergency and incident management.

We can distinguish several cooperating workflows in an ICS. They correspond to the roles played by the various officers and supporting staff of the incident command centers. Workflow management is an important part of an ICS. It is characterized by several distinguishing requirements. The first requirement is a considerable flexibility. An ICS workflow can grow or shrink to cope with frequent changes of the course of actions dictated by incoming events. This is totally different from normal manufacturing system workflows where, once the workflow is established, the users just execute it repeatedly without the necessity of frequent modification. The need of making many ad-hoc changes to ICS workflows calls for an on-the-fly verification of the correctness of the modified workflows.

The second requirement is related to the predominantly volunteer-based workforce of ICS. In the event of a typical incident, many volunteers are called upon to respond to the incident. These volunteers may not have a good understanding of the ICS workflow, not to mention the underlying principles that are used to build the workflow. Therefore, it is a challenge to design and present an ICS workflow in such an *intuitive* way that inexperienced responders are able to follow and use it.

The third requirement comes from the high stakes of ICS workflows. When an incident occurs, we depend on the ICS workflow to reduce property damage and save lives. Therefore, no defects can be tolerated in the workflow. As we mentioned before, an ICS workflow may undergo frequent changes during the course of execution. These changes may introduce inadequate sequences of activities that can jeopardize the correctness of the overall workflow. Therefore, on-the-fly correctness verification is highly recommended. This cannot be achieved without an underlying formal approach of the workflow, which does not leave any scope for ambiguity and sets the ground for analysis.

Therefore, to meet the workflow management needs of an incident command system, a workflow tool must be flexible, intuitive and capable of verifying the correctness of the modeled workflows. An informal workflow tool, such as ANSI Flowchart, UML Activity Diagram [8], or Event-driven Process Chain (EPC) [6], is not suitable for ICS workflow management because it does not allow the verification of the workflows. A number of formal modeling techniques have been proposed in the past decades [1, 2, 3, 4, 7, 9, 10]. Unfortunately, they are not intuitive.

To address these requirements, we introduced a new Workflows Intuitive Formal Approach (WIFA) for the modeling and analysis of workflows [13, 15]. In addition to the abilities of supporting automatic workflow validation and enactment, WIFA possesses the distinguishing feature of allowing users who are not proficient in formal methods to build up and dynamically modify the workflows that address their needs.

In this paper, we further extend WIFA to take resources into account when modeling and enacting workflows. Resources can become important decision factors when combined with control flow information. In many situations, business processes are constrained by scarce resources. The lack of resources can cause contention, the need for some tasks to wait for others to complete, and the slowing down of the accomplishment of larger goals. This is particularly true in an emergency response system where large quantity of resources, including emergency responders, ambulances, fire trucks, medications, food, clothing, etc., are required. Often potential delays can be avoided or reduced by using resource analysis to identify ways in which tasks can be executed in parallel, in the most efficient way. Currently, during a time of crisis, a decision maker often concentrates on a single criterion in

order to simplify, speed up or control the decision process itself. A resource-constrained workflow model can support the decision process by analyzing multiple criteria. It can keep track of resources availability, disable the path that is not executable, and present all executable paths, allowing the emergency responders to make decisions and implement them more confidently.

The paper is organized as follows: Section 2 briefly introduces the workflow formalism of WIFA and its state transition rules. Section 3 presents the extended version of WIFA which takes data into account when executing a workflow. In Section 4, an ICS emergency healthcare example is used to illustrate the use of the data-driven workflow model. Finally, Section 5 presents conclusions and ideas for the continuation of this work.

## 2. OVERVIEW OF WIFA MODEL

This section only briefly introduces WIFA model. A detailed description can be found in [15].

A workflow is composed of *tasks* that are executed according to some order specified by *precedence constraints*. The *preset* of a task $T_k$ is the set of all tasks that are immediate predecessors of the task, denoted by $*T_k$; the *postset* of $T_k$ is the set of all tasks that are immediate successors of the tasks, denoted by $T_k*$. If $|T_k*| \geq 1$, then the execution of $T_k$ might trigger multiple tasks. Suppose $\{T_i, T_j\} \subseteq T_k*$. There are two possibilities: (1) $T_i$ and $T_j$ can be executed simultaneously, and (2) only one of them can be executed, and the execution of one will disable the other, due to the conflict between them. We denote the former case by $c_{ij} = c_{ji} = 0$, and the latter case by $c_{ij} = c_{ji} = 1$.

### 2.1 Workflow Definition

In WIFA, a workflow is defined as a 5-tuple: $WF = (T, P, C, A, S_0)$, where

1) $T = \{T_1, T_2, \ldots, T_m\}$ is a set of *tasks*, $m \geq 1$.
2) $P = (p_{ij})_{mxm}$ is the *precedence matrix* of the task set. If $T_i$ is the direct predecessor of $T_j$, then $p_{ij} = 1$; otherwise, $p_{ij} = 0$.
3) $C = (c_{ij})_{mxm}$ is the *conflict matrix* of the task set. $c_{ij} \in \{0, 1\}$ for $i = 1, 2, \ldots m$ and $j = 1, 2, \ldots m$.
4) $A = (A(T_1), A(T_2), \ldots, A(T_m))$ defines *pre-condition set* for each task. $\forall T_k \in T$, $A(T_k)$: $*T_k \rightarrow 2^{*T_k}$. Let set $A' \in A(T_k)$. Then $T_i \in A'$ implies $p_{ik} = 1$.
5) $S_0 \in \{0, 1, 2, 3\}^m$ is the *initial state* of the workflow.

A state of the $WF$ is denoted by $S = (S(T_1), S(T_2), \ldots, S(T_m))$, where $S(T_i) \in \{0, 1, 2, 3\}$. $S(T_i) = 0$ means $T_i$ is *not executable* at state $S$ and *not executed previously*; $S(T_i) = 1$ means $T_i$ is *executable* at state $S$ and *not executed previously*; $S(T_i) = 2$ means $T_i$ is *not executable* at state $S$ and *executed previously*; and $S(T_i) = 3$ means $T_i$ is *executable* at state $S$ and *executed previously*.

By the above definition of state values, at any state, only those tasks whose values are either 1 or 3 can be selected for execution. Suppose task $T_i$ at state $S_a$ is selected for execution, and the new state resulted from the execution of $T_i$ is $S_b$, then the execution of $T_i$ is denoted by $S_a(T_i)S_b$.

At the initial state $S_0$, for any task $T_i \in T$, if there is no $T_j$ such that $p_{ji} = 1$, then $S_0(T_i) = 1$; otherwise $S_0(T_i) = 0$.

Note that tasks that have no predecessor do not need to wait for any other task to execute first. In other words, these tasks are executable immediately. We assume that there are always such tasks in a workflow. They are the initial triggers or "starting" tasks of workflows.

### 2.2 State Transition Rules

The dynamics of a WIFA workflow can be captured by state transitions. The state transitions are guided by the following rules:

If $S_a(T_i)S_b$, then $\forall\ T_j \in T$,

1) If $T_j = T_i$ then $S_b(T_j) = 2$;
2) If $T_j \neq T_i$ then the state value of $T_j$ at new state $S_b$ depends on its state value at state $S_a$. We consider four cases:

*Case A* – $S_a(T_j) = 0$:
If $p_{ij} = 1$ and $\exists A \in A\ (T_j)$ such that $S_b(T_k) = 2$ for any $T_k \in A$, then $S_b(T_j) = 1$; otherwise $S_b(T_j) = 0$.

*Case B* – $S_a(T_j) = 1$
If $c_{ij} = 0$ then $S_b(T_j) = 1$; otherwise $S_b(T_j) = 0$.

*Case C* – $S_a(T_j) = 2$
If $p_{ij} = 1$ and $\exists A \in A\ (T_j)$ such that $S_b(T_k) = 2$ for any $T_k \in A$, then $S_b(T_j) = 3$; otherwise $S_b(T_j) = 2$.

*Case D* – $S_a(T_j) = 3$
If $c_{ij} = 0$ then $S_b(T_j) = 3$; otherwise $S_b(T_j) = 2$.

According to the above state transition rules, for example, a task's state value at a given state other than the initial state is 0 iff one of the following holds:

1) Its state value is 0 in the previous state, and it is not the successor of the task which is just executed.
2) Its state value is 0 in the previous state, and it is the successor of the task which is just executed, but for each of its precondition sets there is at least one task that is not executed.
3) Its state value is 1 in the previous state but it conflicts with the task which is just executed.

Note that a state value can increment from 0 to 1, from 1 to 2 or from 2 to 3; it can also decrement from 1 to 0 or from 3 to 2. But it cannot decrement from 2 to 1.

### 2.3 Well-Formed Workflows

Two important subclasses of workflows are introduced in WIFA: well-formed workflows and confusion-free workflows. Here we only give their informal definitions. A workflow is *well-formed* if and only if the following two *behavior conditions* are met:

1) There is no dangling task.
2) Given any reachable state, there is always a path leading the workflow to finish.

The workflow of Fig. 1 is well-formed, because every task in this workflow is executable, and it is always guaranteed to finish.

To further simplify the workflow structure, we introduce *confusion-free* workflows, which are a class of well-formed workflows. A well-formed workflow is *confusion-free* if and only if the following two *structural conditions* are met:

1) Either all tasks triggered by the same task are in conflict, or no pair of them is in conflict.
2) A task becomes executable either when all of its predecessor tasks are executed, or when any one of them is executed.

From the perspective of triggering condition and relation among triggered tasks, tasks in a confusion-free well-formed workflow can be classified into four types:

1) *AND-in-AND-Out* A task belongs to this class iff it is not executable until all its direct predecessor tasks are executed, and after it is executed, all its direct successor tasks can be executed in parallel.
2) *AND-in-XOR-out* A task belongs to this class iff it is not executable until all its direct predecessor tasks are executed, and after it is executed, only one of its direct successor tasks can be executed.
3) *XOR-in-AND-out* A task belongs to this class iff it is executable as long as one of its direct predecessor tasks is executed, and after it is executed, all its direct successor tasks can be executed in parallel.
4) *XOR-in-XOR-out* A task belongs to this class iff it is executable as long as one of its direct predecessor tasks is executed, and after it is executed, only one of its direct successor tasks can be executed.

Without loss of generality, a task with only one or no direct predecessor is treated as an "And-In" task, and a task with only one or no direct successor treated as an "AND-out" task.

## 3. EXTENDED WIFA MODEL

Now, we extend the WIFA model described in the previous section to take resource constraints and decision into an account. To develop such a model, we modify WIFA by redefining the definitions and state transition rules.

### 3.1 Resource-Constrained Workflow

We assume there are $n$ types of resources in a system, and the quantity of each type of resource can be represented by a natural number. Hence, resources are described by $R = (r_1, r_2, \ldots, r_n)$ with each $r_i$ an integer. $R$ is a global data, which can be accessed and modified by all instances of a workflow model when a task is executed.

A task may consume certain resources and meanwhile release or produce certain other resources when it is executed. We use $R^c(T_k) = \{r^c_{k1}, r^c_{k2}, ..., r^c_{kn}\}$ and $R^p(T_k) = \{r^p_{k1}, r^p_{k2}, ..., r^p_{kn}\}$ to describe the resources consumed and produced when task $T_k$ is executed, respectively. If multiple mutual exclusive tasks are executable at a state, the decision to selection one of them for execution will be based on the evaluation of current resources situation. To the end, each task has a defined *decision function*, which evaluates the resources when the task is executed. Each task is also associated with a *decision value*. The decision functions and decision values jointly decide which task should be selected for execution.

**Definition 1 (Workflow control structure):** A workflow control structure is a four-tuple describing the control flow aspect of a workflow. $WFCS = (T, P, C, A)$, where

1) $T = \{T_1, T_2, \ldots, T_m\}$: a set of tasks in a workflow
2) $P$: Describes a precedence relationship between two tasks
3) $C$: Indicates if tasks are in conflict and competition for resources
4) $A$: Specifies the pre-condition set for a task to become control-ready

In the rest of the paper, we will simply call WFCS a *control flow*.

**Definition 2 (Resource-constrained workflow):** A resource-constrained workflow is defined as $RCWF = (WFCS, R^c, R^p, F, D, S_0)$, where

1) $WFCS$: The workflow control structure as defined in Definition 1.
2) $R^c = (R^c(T_1), R^c(T_2), \ldots, R^c(T_m))$ describes the quantity of each type of resource consumption in a task execution, with $R^c(T_k) = \{r^c_{k1}, r^c_{k2}, ..., r^c_{kn}\}$, where $r^c_{kj}$ represents the quantity of the resource of type $j$ consumed when task $T_k$ is executed.
3) $R^p = (R^p(T_1), R^p(T_2), \ldots, R^p(T_m))$ describes the quantity of each type of resource production in a task execution, with $R^p(T_k) = \{r^p_{k1}, r^p_{k2}, ..., r^p_{kn}\}$, where $r^p_{kj}$ represents the quantity of the resource of type $j$ produced (or released) when task $T_k$ is executed.
4) $F = (f_1, f_2, \ldots, f_m)$: Decision functions for decision support. $f_i$ is associated with task $T_i$ and maps the resource set $R$ to an integer.
5) $D = (d_1, d_2, \ldots, d_m)$:Decision values. $d_i$ is associated with task $T_i$. A decision value describes a range of the acceptable values from evaluating the decision function of the predecessor.
6) $S_0 = (H_0, R_0)$ is the initial state, with $H_0 \in \{0, 1, 2, 3\}^m$ being the state element of control flow and $R_0$,

the value of $R$ at the initial state, being the element representing the availability of resources.

Note that $R^c$ and $R^p$ are all static data; they are specified according tasks' resource consumption and production. However, $R$ is dynamic; it changes with the state transition of any instance of a workflow model. The state element of resources at state $S_i$ is denoted by $R_i = R(S_i) = (r_{i1}, r_{i2},\ldots, r_{in})$. Thus $R_0 = R(S_0) = (r_{01}, r_{02},\ldots, r_{0n})$.

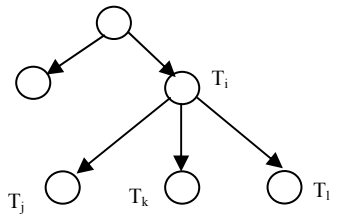



Fig. 1. Path selection.

Before we go further to discuss the state transition rules, let us briefly illustrate how to use the decision functions and values to determine the path to take at an XOR out decision point. Fig. 1 depicts an example scenario. Suppose the execution of $T_i$ brings the workflow to state $S_a$. After $T_i$ is executed, $T_j$, $T_k$ and $T_l$ are triggered. Suppose $C_{jk} = C_{kl} = C_{lj} = 1$ ($T_j$, $T_k$ and $T_l$ are in conflict), we have to determine which branch to take based on the current resource status. Use the current resource state $R_a = R(S_a)$ as the parameters and plug into the pre-defined decision function, $f_i(R_a)$. Then, we compare the result with the decision values of the successors ($T_j$, $T_k$ and $T_l$). If $f_i(R_a) \in d_j$ (the result falls in the range of the decision value specified in $T_l$), then this branch is selected for execution.

### 3.2 State Transition Rules

There are three rules which dictate the executability of a task. The executability of a task simply indicates if a task can be executed at a state. The first rule deals with the control aspect of the workflow. In other words, it describes which task is triggered regardless of the data constraints. If a task is triggered, we call it a *control-ready* task. The WIFA definitions presented in the previous section deals only with this aspect of the workflow. The second rule checks on the triggered tasks to see if sufficient resources are available in order to implement them. If there are not enough resources available to perform the task, then it is not an executable task. If there are enough resources to carry out the task, then we call it a *resource-ready* task. The last rule deals with the selection of a path to take based on the available resources. This rule becomes important when there are tasks in conflict with each other. Hence, a decision has to be made on which path to take. It evaluates the current resources against user defined rules and constraints, and then it selects a task which in terms become an executable task and other competing task would be non-executable.

**Definition 3 (Executability):** Task $T_k$ at state $S_i$ is said to be *executable* if and only if it meets all of the following:

1) *Control-Ready*: $H_i(T_k) \in \{1, 3\}$, which means $T_k$ is triggered in terms of control flow. In another words, from the control flow aspect of the workflow, the task has to be triggered by its predecessor(s).
2) *Resource-Ready*: $R^c(T_k) \leq R_i$, a task, $T_k$, requires a certain amount of resources in order to be executed. If the current set of resources, $R_i$, does not have enough resources required to execute the task, then the task is not executable.
3) *Branch-Selected*: This rule is concerned with decision making when multiple tasks that are in conflict with each other are triggered:
   a. If $c_{kj} = 0$ for any $j \neq k$, task $T_k$ is not in conflict with other tasks, then select this task
   b. Otherwise, if $f_i(R_i) \in d_k$, i.e. the evaluated value of the decision function falls in the range of the decision value range of $T_k$, then $T_k$ can be selected for execution.

**Definition 4 (State transition rules)** State Transition Rules: If a task $T_k$ satisfies all of the three conditions listed in Definition 3 at state $S_i$, it is executable. Once the task is executed, the new state $S_j = (H_j, R_j)$ will be determined according to the following rules:

1. $H_j$ is changed according to the state transition rules of basic WIFA workflow.
2. The resource state $R_j$ is derived from the previous resource state $R_i$, resource consumed by $T_k$, $R^c(T_k)$, and resource produced by $T_k$, $R^p(T_k)$, according the following formula:

$$R_j = R_i - R^c(T_k) + R^p(T_k)$$

### 3.3 Correctness and verification

The correctness of a resource-constrained workflow is defined and verified in three dimensions. There are: control flow correctness, decision model correctness and resource model correctness.

#### *3.3.1 Control flow correctness*

The underlying control flow of a resource-constrained workflow is correct if it is well-formed as defined in Section 2.3. The well-fomedness of the control flow can be verified through reachability analysis.

#### *3.3.2 Decision model correctness*

For each decision construct as shown in Fig. 1, we require:

- No overlap among the ranges of decision values of tasks in conflict.
- The range of decision function of a task is equal to the union of the ranges of decision values of its succeeding tasks in conflict.

For the construct shown in Fig. 2, the two requirements can be mathematically described as

$$d_j \cap d_k = d_k \cap d_l = d_j \cap d_k = \varnothing, \text{ and}$$

$d_j \cup d_k \cup d_l = \{x \mid x = f_i(R)$ for all possible $R\}$

*3.3.3 Resource model correctness*

Note that the resources incorporated in a workflow model are used by all the instances of the model. With other words, they are considered to be global resources. Hence, issues related to resource contention arise. Resource contention can block tasks from being executed immediately because the required resources are held by other instances. It can also cause deadlock in case an instance is holding resource A and waiting for resource B while another instance is holding B and waiting for A. Deadlock can also caused by two concurrent tasks in a single instance which are contending for common resources.

Several approaches can be used to avoid deadlock due to resource contention. One is to book all resources that could be required when an instance is created. This way, the instance can finish without additional resource request in the course of execution. The drawback of this approach is, if all required resources are not available then the instance cannot start execution.

Another approach is to prioritize all tasks in a workflow in terms of their privileges of resources utilization, then apply the existing deadlock avoidance protocol, such as priority-ceiling protocol [5], when allocating resources to tasks. The advantage of this approach is in its good real-time performance. The disadvantage is, when applied to avoid deadlock among multiple instances, that it requires communication of workflow execution status across instances.

## 4 ICS Emergency Medical Care Workflow Modeling

In the event of incident with injuries, the injured people will be taken to emergency rooms for immediate treatment. The example here depicts the workflow of the emergency medical care service. Resources to be modeled include physicians, nurses, and examining rooms, as well as the resource consumers, the patients.

When a patient first arrives at emergency room, he/she proceeds to check in. Then, the receptionist checks the resources and number of current patients to determine a waiting time. Since the waiting time is usually proportional to the number of waiting patients and inversely proportional to the number of doctors, we will use them to determine the patient's waiting time. For this example, we assume if the number of current waiting patients exceed 7 times of the total number of physicians, then the newly arrived patient would not be admitted into the emergency room. Hence, he/she would be sent to other emergency room immediately. Otherwise, the patient would check into the emergency room and be given paperwork to fill out. After the patient completes the paperwork, he/she would wait to be treated and an available nurse would start processing the paperwork. When the nurse completes the process and there is an examining room available for the patient, then the patient would enter the room. When a physician becomes available, he/she would start examining the patient in the examining room. After the completion of the treatment, the patient proceeds to check out and leave the emergency room.

### 4.1 The control flow

Based on the above specification, we build the WIFA model as shown Fig. 2. The legend of the tasks is shown in Table 1. In this figure, tasks with single predecessor and single successor are represented by a circle. Other tasks are represented by intuitive icons according to their types. $T_2$ is a decision point where the number of waiting patients determines the path of the workflow. If there are too many waiting patients, the emergency room would not admit new patients. Hence, $T_2$ is XOR out. In another word, $T_3$ and $T_{11}$ are in conflict with each other. At $T_3$, patient finishes filling out the paperwork, he/she would wait while a nurse is processing the paperwork. Therefore, $T_3$ is AND out. $T_7$ states that if there is a room available, the nurse has processed the paperwork and the patient is waiting, then the patient can enter the examining room. Therefore, $T_7$ is AND in. When a physician finishes treating a patient, the patient will proceed to check out. When a patient arrives at the hospital, it would trigger a new instance of the workflow which accesses the same global data as other instances of the workflow model. For this example, the global data are the number of physicians, nurses, examining rooms and patients.

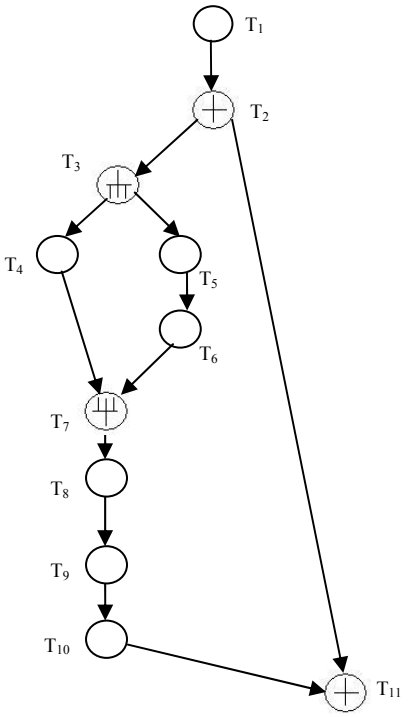



Fig. 2. A healthcare workflow model.

Table 1 Legend of the tasks.

| Task | Description |
|---|---|
| $T_1$ | Patient arrives at the emergency room |
| $T_2$ | Determine whether to admit the patient or not. |
| $T_3$ | Patient checks in and fills out the paperwork. |
| $T_4$ | Patient waits to be treated. |
| $T_5$ | Nurse starts processing the paperwork. |
| $T_6$ | Nurse finishes processing the paperwork. |
| $T_7$ | Patient enters an examining room (occupies a bed). |
| $T_8$ | Physician starts treating the patient. |
| $T_9$ | Physician finishes treating the patient. |
| $T_{10}$ | Patient checks out of the emergency room. |
| $T_{11}$ | Patient leaves the emergency room. |

This workflow is formulated in the WIFA framework as the following:

$$T = \{T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}, T_{11}\},$$

$$P = \begin{bmatrix} 0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&1 \\ 0&0&0&1&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0 \end{bmatrix},$$

$$C = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0 \end{bmatrix}$$

$$A(T_1) = \phi, A(T_2) = \{\{T_1\}\}, A(T_3) = \{\{T_2\}\}, A(T_4) = \{\{T_3\}\},$$
$$A(T_5) = \{\{T_3\}\}, A(T_6) = \{\{T_5\}\}, A(T_7) = \{\{T_4, T_6\}\},$$
$$A(T_8) = \{\{T_7\}\}, A(T_9) = \{\{T_8\}\}, A(T_{10}) = \{\{T_9\}\},$$
$$A(T_{11}) = \{\{T_2\}, \{T_{10}\}\}.$$

### 4.2 Resource model

There are four types of resources: physicians, nurses, examination rooms and patients. Denote by

$r_1$: Number of available physicians

$r_2$: Number of available nurses

$r_3$: Number of available examining rooms

$r_4$: Number of waiting patients

Assume that in the emergency room there are 3 physicians, 6 total nurses, and 5 examining rooms. At any time the number of waiting patients cannot exceed 21, which is 7 times the number of physicians.

When a new patient arrives at the hospital, a new workflow instance is initiated to track the patient's activity. The hospital checks the number of currently waiting patients. If the number is not greater than 21, he/she will be admitted into the hospital and requested to fill out the paperwork. Otherwise, he/she will be declined.

After the patient completes the paperwork, he/she hands it in to be processed by a nurse and waits. After the paperwork is processed and there is an available room, the patient would be asked to enter the examining room. When a physician is available, he/she would start treating the patient, and when the treatment is completed, the patient would proceed to check out and leave.

For the first instance of the workflow, the initial state is specified by

$H_0 = (1,0,0,0,0,0,0,0,0,0,0)$.

$R_0 = (3, 6, 5, 0)$.

Among these eleven tasks, only tasks $T_5$, $T_7$, and $T_8$ acquire resources for execution, and only tasks $T_3$, $T_6$, and $T_9$ release resources when executed. Let $\vec{0} = (0,0,0,0)$. The resource consumption and production vectors are defined as following:

1) $R^c = (\vec{0}, \vec{0}, \vec{0}, \vec{0}, r_5^c, \vec{0}, r_7^c, r_8^c, \vec{0}, r_{10}^c, \vec{0})$, where

- $r_5^c = (0, 1, 0, 0)$, for $T_5$ consumes one nurse to process the paperwork;
- $r_7^c = (0, 0, 1, 1)$, for $T_7$ uses an examining room and takes away one waiting patient;
- $r_8^c = (1, 0, 0, 0)$, for $T_8$ employs a physician to treat a patient;

2) $R^p = (\vec{0}, \vec{0}, r_3^p, \vec{0}, \vec{0}, r_6^p, \vec{0}, \vec{0}, r_9^p, \vec{0}, \vec{0})$, where

- $r_3^p = (0, 0, 0, 1)$, for $T_3$ adds a new patient into the waiting patients queue since the task is that patient checks into the hospital and completes the paperwork;
- $r_6^p = (0, 1, 0, 0)$, for the execution of $T_6$ releases a nurse. When a nurse finishes processing the paperwork, he/she becomes available again. Hence, it increases the number of available nurses by one.
- $r_9^p = (1, 0, 1, 0)$, for the execution of $T_9$ releases a physician.

### 4.3 Decision model

Among all 11 tasks, $T_2$ is the only decision task. We have

$$F = (\phi, f_2, \phi, \phi, \phi, \phi, \phi, \phi),$$
$$f_2 = r_4 / 7.$$

$f_2$ evaluates ratio of the number of waiting patient to 7. $T_2$ has two successors: $T_3$ and $T_{11}$. The decision values vector is

$$D = (\phi, \phi, d_3, \phi, \phi, \phi, \phi, \phi, \phi, \phi, d_{11}),$$

Where, $d_3 = [0, r_1]$, meaning if the number of waiting patient is less than or equal to 7 times of the number of total physicians and grater than or equal to zero, then this task would be selected; $d_{11} = (r_1, \infty)$, meaning if the result of the predecessor's evaluating function is greater than 7 times the number of total physicians, then this task is selected. We use $\phi = (-\infty, \infty)$ for the decision values of tasks which are not involved in any decisions.

### 4.4 State transitions

Now, let us examine the execution of this workflow. We will only show the entire execution of one instance of this workflow.

At $S_0$, $T_1$ is the only executable task. $H_0$ = (1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0). $T_1$ satisfies the rules of executability. Namely, $T_1$ is

- Control ready: $H_0(T_1) = 1$;
- Resource ready: $R^c(T_1)$ = (0, 0, 0, 0) and $R_0$ = (3, 6, 5, 0), $R^c(T_1) \le R_0$; and
- Branch selection: $T_1$ is not in conflict with other tasks, hence it is selected.

Let $S_0(T_1)S_1$, then based on the state transition rules, we have

$$H_1 = (2,1,0,0,0,0,0,0,0,0,0)$$
$$R_1 = R_0 - R^c(T_1) + R^p(T_1) = (3,6,5,0).$$

At $S_1$, $T_2$ is executable because it is

- Control ready: $H_1(T_2) = 1$;
- Resource ready: $R^c(T_2) = (0,0,0,0) \le R_1 = (3,6,5,0)$; and
- Branch selection: $T_2$ is not in conflict with other tasks.

Let $S_1(T_2)S_2$, then based on the state transition rules, we have

$$H_2 = (2,2,1,0,0,0,0,0,0,0,1) \text{ and } R_2 = (3,6,5,0).$$

At $S_2$, $T_3$ and $T_{11}$ are both

- Control ready: $H_2(T_3) = H_2(T_{11}) = 1$;
- Resource ready: $R^c(T_3) = (0,0,0,0) \le R_{2,} = (3,6,5,0)$;

However, $T_3$ and $T_{11}$ are in conflict with each other, $C_{3,11} = C_{11,3} = 1$. Based on Branch selection rule, we have to evaluate $f_2(R_2) = r_4 / 7$. The result, $f_2(R_2) = 0/7 = 0$, falls in the range of $f_2(R_2) \in d_3 = [0, r_1] = [0,3]$. Therefore, task $T_3$ is selected. Hence, $T_3$ is the only executable tasks at $S_2$. Let $S_2(T_3)S_3$, then based on the state transition rules, we have

$$H_3 = (2,2,2,1,1,0,0,0,0,0,0) \text{ and } R_3 = (3,6,5,1).$$

Notice that $T_{11}$ is not executable now because $T_3$ and $T_{11}$ are in conflict. Execution of $T_3$ disables the execution of $T_{11}$. Furthermore, the number of waiting patient increased by one.

At $S_3$, $T_4$ and $T_5$ are both

- Control ready: $H_3(T_4) = H_3(T_5) = 1$;
- Resource ready: $R^c(T_4) = (0,0,0,0) \le R_3 = (3,6,5,1)$ $R^c(T_5) = (0,1,0,0) \le R_3 = (3,6,5,1)$; and
- Branch selection: $T_4$ and $T_5$ are not in conflict with each other or other tasks. Therefore, both tasks are selected.

So at $S_3$, both $T_4$ and $T_5$ are executable. Let's assume $T_4$ is executed first and let $S_3(T_4)S_4$, then based on state transition rules, we have

$$H_4 = (2,2,2,2,1,0,0,0,0,0,0) \text{ and } R_4 = (3,6,5,1).$$

At $S_4$, $T_5$ is the only executable task since it is

- Control ready: $H_4(T_5) = 1$;
- Resource ready: $R^c(T_5) = (0,1,0,0) \le R_4 = (3,6,5,1)$; and
- Branch selection: $T_5$ is not in conflict with any other tasks. Therefore, it is selected.

Let $S_4(T_5)S_5$, then based on state transition rules, we have

$$H_5 = (2,2,2,2,2,1,0,0,0,0,0) \text{ and } R_5 = (3,5,5,1).$$

Notice that the number of available nurses decreased by one. Since one nurse is required to process the paperwork, there would be one less available nurse at this state.

At $S_5$, $T_6$ is executable since it is

- Control ready: $H_5(T_6) = 1$;
- Resource ready: $R^c(T_6) = (0,0,0,0) \le R_5 = (3,5,5,1)$; and
- Branch selection: $T_6$ is not in conflict with other tasks.

Let $S_5(T_6)S_6$, then based on state transition rules, we have

$$H_6 = (2,2,2,2,2,2,1,0,0,0,0) \text{ and } R_6 = (3,6,5,1).$$

Notice the number of available nurses is added by one since the nurse is free up again after processing the paperwork.

At $S_6$, $T_7$ is executable since it is

- Control ready: $H_6(T_7) = 1$;
- Resource ready: $R^c(T_7) = (0,0,1,1) > R_6 = (3,6,5,1)$; and
- Branch selection: $T_7$ is not in conflict with other tasks.

Let $S_6(T_7)S_7$, then based on state transition rules, we have

$$H_7 = (2,2,2,2,2,2,2,1,0,0,0) \text{ and } R_7 = (3,6,4,0).$$

Notice the execution of $T_7$ consumes an examining room. Hence, the number of available examining room returned to zero. It also takes away one waiting patient, so the number of waiting patient decreased by one as well.

At $S_7$, $T_8$ is executable since it is

- Control ready: $H_7(T_8) = 1$;
- Resource ready: $R^c(T_8) = (1,0,0,0) \le R_6 = (3,6,4,0)$; and
- Branch selection: $T_8$ is not in conflict with other tasks.

Let $S_7(T_8)S_8$, then based on state transition rules, we have

$$H_8 = (2,2,2,2,2,2,2,2,1,0,0) \text{ and } R_8 = (2,6,4,0).$$

Notice, the number of available physicians decreased by one since he/she is busy treating a patient at task $T_8$.

At $S_8$, $T_9$ is executable. Let $S_8(T_9)S_9$, then based on state transition rules, we have

$$H_9 = (2,2,2,2,2,2,2,2,2,1,0) \text{ and } R_9 = (3,6,5,0).$$

Notice the number of physicians and rooms both increased by one since the execution of $T_9$ signifies the completion of treatment which frees up those resources, namely a room and a physician.

At $S_9$, $T_{10}$ is executable. Let $S_9(T_{10})S_{10}$, then based on state transition rules, we have

$$H_{10} = (2,2,2,2,2,2,2,2,2,2,1) \text{ and } R_{10} = (3,6,5,0).$$

At $S_{10}$, $T_{11}$ is executable. Let $S_{10}(T_{11})S_{11}$, then based on state transition rules, we have

$$H_{11} = (2,2,2,2,2,2,2,2,2,2,2) \text{ and } R_{11} = (3,6,5,0).$$

***Note***: If multiple instances are created from the workflow model and are running concurrently, then resources $R$ will be updated by every instance. More specifically, we need to check the *global R* to decide if a task in an instance is executable, and update $R$ when a task in an instance is executed.

## 5 CONCLUDING REMARKS

In this paper, we introduced a resource-constraint decision support workflow model. This model enables users to specify resources consumption and production when executing a task, and decision policies to choose a path at a given situation where multiple execution branches are available. The work is particularly important for an emergency response system workflow management, where large quantity of resources, including emergency responders, ambulances, fire trucks, medications, food, clothing, etc., are required. Moreover, at a time of crisis, a decision needs to be made rapidly in order to respond in a timely manner. A data-driven workflow model can help the user make a logical decision by evaluating the present workflow status.

We are currently enhancing our workflow tool [12] to support resources and decision modeling. We are also investigating the best deadlock avoidance strategy for incident command systems due to resource contention.

## REFERENCES


[1] W.M.P. van der Aalst, "Verification of workflow nets", *Proceedings of Application and Theory of Petri Nets*, Volume 1248 of Lecture Notes in Computer Science, pp. 407-426, 1997.

[2] N. R. Adam, V. Atluri and W. Huang, "Modeling and analysis of workflows using Petri nets", Journal of Intelligent Information Systems, pp. 131-158, March 1998.

[3] P. C. Attie, M. P. Singh, A. Sheth and M. Rusibkiewicz, "Specifying Interdatabase Dependencies," Proceedings 19th International Conference on Very Large Database, pp.134-145, 1993.

[4] P. Dourish, "Process descriptions as organizational accounting devices: the dual use of workflow technologies", ACM GROUP'01, Sept. 30-Oct. 3, 2001, Boulder, Colorado, USA.

[5] J.B. Goodenough and L. Sha, "The priority ceiling protocol: A method for minimizing the blocking of high priority Ada tasks," Proc. 2nd Int. Workshop on Real-time Ada Issues, pp. 23-31, 1998.

[6] G. Keller, M. N¨uttgens, and A.W. Scheer. "Semantische processmodellierung auf der grundlage ereignisgesteuerter processketten (EPK)." Ver¨offentlichungen des Instituts f¨ur Wirtschaftsinformatik, Heft 89 (inGerman), University of Saarland, Saarbr¨ucken, 1992.

[7] P. Lawrence, editor, Workflow Handbook 1997, Workflow Management Coalition, John Wiley and Sons, New York, 1997.

[8] Lee, R.C, and W.M. Tepfenhart, UML and C++: A practical guide to object-oriented development (2nd Edition), Prentise-Hall (2000).

[9] M. Reichert, P. Dadam, "ADEPTflex – supporting dynamic changes of workflows without losing control", Journal of Intelligent Information Systems, 10 (2), 1998, pp.93-129.

[10] D. Rosca, S. Greenspan, C. Wild, "Enterprise modeling and decision-support for automating the business rules lifecycle", Automated Software Engineering Journal, Kluwer Academic Publishers, vol.9, pp.361-404, 2002.

[11] M.P. Singh, G. Meredith, C. Tomlinson, and P.C. Attie, "An event algebra for specifying and scheduling workflows," Proceedings 4th International Conference on Database System for Advance Application, pp. 53-60, 1995.

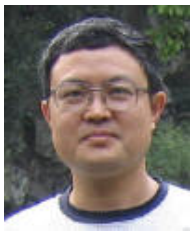
[12] M. Stoute, J. Wang, and D. Rosca, "Workflow management tool support for incident command systems", *Proceedings of ICNSC'06,* Ft. Lauderdale, FL, 2006

[13] J. Wang and D. Rosca, "Dynamic workflow modeling and verification," *Proceedings of International Conference on Advanced Information Systems Engineering*, Luxembourg, June 5-9, 2006.

[14] J. Wang, D. Rosca, W. Tepfenhart, and A. Milewski, "Incident command systems workflow modeling and analysis: A case study," Proceedings of the 3rd International ISCRAM Conference (B. Van de Walle and M. Turoff, eds.), Newark, NJ (USA), May 2006.

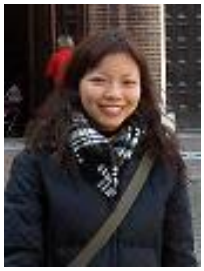
[15] J. Wang, D. Rosca, W. Tepfenhart, A. Milewski and M. Stoute, "An intuitive formal approach to dynamic workflow modeling and analysis," Proceedings of the 3rd Conference on Business Process Management, Nancy, France, Sept. 6-8, 2005.


William Tepfenhart received the PhD in Physics from the University of Texas at Dallas in 1987. His is currently an associate professor of the software engineering department at Monmouth University, West Long Branch, New Jersey, USA. From 1991 to 1999, he was a member of the technical staff with AT&T Bell Labs/AT&T Laboratories in Middletown, New Jersey. Prior to joining AT&T, he was a senior scientist at Knowledge Systems Concepts, Inc in Rome New York. His experience ranges across a broad spectrum of activities. He has performed in the role of instructor, researcher, software developer, and author. Trained as a physicist, his areas of expertise include object-oriented software development, artificial intelligence, and software engineering. His knowledge of modeling physical systems has formed the basis for major contributions in the area of software development. Dr. Tepfenhart is a member of the IEEE.



Jiacun Wang received the PhD in computer engineering from Nanjing University of Science and Technology (NUST), China, in 1991. He is currently an associate professor of the Software Engineering Department at Monmouth University, West Long Branch, New Jersey, USA. From 2001 to 2004, he was a member of scientific staff with Nortel Networks in Richardson, Texas. Prior to joining Nortel, he was a research associate of the School of Computer Science, Florida International University (FIU) at Miami. Prior to joining FIU, he was an associate professor at NUST. His research interests include software engineering, discrete event systems, formal methods, wireless networking, and real-time distributed systems. He authored Timed Petri Nets: Theory and Application (Norwell, MA: Kluwer, 1998), and published more than 50 research papers in journals and conferences. He is an Associate Editor of IEEE Transactions on Systems, Man and Cybernetics, Part C, and has served as a program committee member for many international conferences. Dr. Wang is a senior member of IEEE.

Dr. Daniela Rosca is an Associate Professor in the Department of Software Engineering at Monmouth University. She was a Visiting Scientist at GTE laboratories, Waltham, MA where she investigated issues related to process modeling and business rules. Her current research interests include: Dynamic workflows modeling, inter-organizational workflows, business rules, requirements engineering, web-based environments.



Anni Tsai received the Bachelor of Science in computer science from Rutgers University, New Jersey, USA, in 2004. She is currently pursing the master degree in software engineering at Monmouth University, West Long Branch, New Jersey, USA. She is a member of IEEE and serves as the vice chair and treasurer for the local chapter.