Precise yet Flexible Specification of Emergency Resolution Procedures

Manuel Llavador[1], Patricio Letelier[1], Mª Carmen Penadés[1], José H. Canós[1], Marcos Borges[2] and Carlos Solís[1]

[1]Information Systems and Computation Dept.
Technical University of Valencia
46071 – Valencia (Spain)
{mllavador | letelier | jhcanos | mpenades | csolis} @dsic.upv.es

[2]Graduate Program in Informatics
NCE & IM
Universidade Federal do Rio de Janeiro
Brasil
mborges @nce.ufrj.br

ABSTRACT

Emergency Managers face a number of critical problems related to the compilation, validation, and use of Emergency Procedures. Traditional approaches do not provide enough expressiveness to accurately specify emergency procedures covering each possible scenario. As a result of this situation, Emergency Procedures are not as useful as they should be, neither in prevention nor during resolution of an emergency. In this work, we present an approach that merges two techniques to provide the broad expressiveness required when specifying Emergency Procedures. To represent sequences on actions performed by different participants we use workflow techniques. On the other hand, we use rules to represent available or mandatory actions according to the state of the system during the emergency. These rules are expressed in dynamic logic as the underlying formalism. Our approach provides more expressiveness and precision for the specification of Emergency Procedures, offering better conditions for their verification and validation. As a case study we have used part of a city subway Emergency Procedure.

Keywords

Emergency Response Systems, Emergency Plans, WorkFlows, Rule-Based Processes, Dynamic Logic

INTRODUCTION

Procedure planning is an important issue in the emergency domain, because procedures must guide emergency managers through the resolution of emergency situations, under critical and stressful conditions. The so-called Emergency Plan serves both as a legal document and a manual that describes those procedures. It contains information about the kind and location of available safety equipment (such as emergency exits and fire barriers) and the detailed procedures to follow in a number of eventualities (for example, a fire in a subway tunnel, crashes, and equipment malfunctions).

Despite the critical settings where they are applied, most Emergency Procedures are represented using either a natural language description (which is ambiguous, informal, and incomplete) or a general workflow model (without explicitly including all scenarios). This information, combined with additional actions from experienced managers (based on their tacit knowledge) and context information, can be sufficient to generate an appropriate response for simple scenarios, but not for complex ones. Process modeling based on workflow technologies has been used to represent procedures in different domains (Jablonski and Bussler, 1996), but it has several disadvantages when dealing with dynamic and context-aware scenarios. Rule-based workflow (Ellis and Keddara, 2000) is an alternative but also presents some drawbacks; for example, it is difficult to test and visualize the process definition and/or state when the number of rules is very high.

In this paper, we propose a formal approach that combines the good properties of both workflow and rule-based approaches. As an underlying formalism (not visible from the user point of view), we use a variant of Dynamic Logic (Harel, Kozen and Tiuryn, 2000), which makes our approach more expressive and also provides advantages

for validation, verification, simulation, and other desirable features. We use workflow models where emergency procedures are stable; and use rules whenever some flexibility is needed. To unify the expressiveness and provide executability of our mixed models, both the workflow models and the rules are automatically transformed to a Dynamic Logic specification.

As a guide for the emergency plan designers, we categorize the space of actions using a four-quadrant conceptual space. There, process and rules are crossed with obligations and permissions, and explain how all the possible categories can be expressed in dynamic logic. We illustrate our approach using a case study extracted from the Emergency Plan of MetroValencia, the company running the public underground transportation system of Valencia.

The remainder of this paper is structured as follows. The following section introduces a set of requirements for the definition of Emergency Procedures, presenting the advantages and disadvantages of Workflow and Rule-based approaches with respect to their expressiveness. We then describe our new approach, which combines good properties of both workflow and rule-based approaches using Dynamic Logic. This new approach is illustrated with a case study. And finally, we present our conclusions and directions for future work.

MODELLING EMERGENCY PROCEDURES

In most cases, the emergency response procedures are described using natural language, as the example given in Figure 1 (an excerpt of the Metro Valencia Emergency Plan (Metro Valencia Safety Office, 1998)). However, the intrinsic ambiguity of natural languages, long recognized by the Requirements Engineering community, is a serious obstacle to establish precise processes making more formal approaches necessary.

A) The driver or station manager, after being aware of the fire, must notify both the type of incident and the train's location to the Safety Manager (SFT) at the Traffic Control Office (TCO). The SFT must quickly evaluate the situation before making the subsequent decisions: signal the emergency level (1 or 2) of the incident to the staff in the destination station and the next station; request the traffic controllers to stop the circulation of trains in the affected subway line.

B) In order to evaluate the magnitude of the emergency, the SFT must immediately request more information to the following actors:

   a. The Station Manager (STN)
   b. The drivers of the trains (TRN) that were in the affected subway line

C) If there is not enough information, the SFT can send a team with emergency equipment (flashlights, oxygen tanks, etc.) to evaluate the affected zone.

D) When a zone has Emergency Level 2, the installation managers must turn the electric power off as soon as the train has stopped and the passengers have disembarked. The SFT must request Fire Extinction Services using the Emergency Coordination and the Communication Center alert systems.

Figure 1. Fragment of the Metro Valencia Emergency Plan corresponding to "Fire in a train"

Workflow Management Systems

A Workflow Management System (WfMS) is one which provides procedural automation of a process by handling the set of work activities and invocation of humans and resources associated with the various activity steps (Workflow Management Coalition, 1995). Despite the variety of WfMSs, all of them exhibit a number of common features. A process definition tool is used to create the process description using a workflow definition language that captures the behavior of actors and the temporal flow of activities. Normally, a workflow model is specified using a formal process definition language, though in simpler systems, a script or a set of routing commands can be used to provide basic coordination capabilities.

In recent years, many workflow description languages have been proposed. Van der Aalst describes and analyzes existing workflow description languages according to five perspectives: functional, process, organizational, informational and operational (Aalst and Weskez, 2003). The functional perspective characterizes the activities to be performed during a workflow execution. The process perspective specifies the starting conditions and the control flow defining the order of execution of the tasks, as well as their interrelationships. The organization perspective

describes the information of the organizational structure and the population in which the workflow is executed; typically, the structure of an organization is defined by roles and groups which participate in a set of process activities. The information perspective covers data, partitioned in control data and production data; the former are used for workflow management purposes, mainly routing the control flow of execution, whereas the latter are represented by information objects (e.g., documents, forms, and tables) whose existence does not depend on workflow management. Finally, the operation perspective describes elementary operations performed by resources and applications.

A workflow schema is the specification covering all these perspectives. Workflow schemas have been traditionally represented using directed graphs, whose nodes represent tasks and whose edges represent control (and data) flow between tasks. While control flow specifies an execution order, the conditions under which a given workflow instance is executed are defined by its start condition. This simplifies the definition of processes where the order of activities is predefined.

Figure 2 shows the workflow model corresponding to the emergency plan in Figure 1. It includes some actions removed from the text-based description. We use the notation of UML 2.0 Activity Diagram (Object Management Group, 2005). The process begins when the Safety Manager (SFT) receives an emergency warning. According to sentence A of the text-based emergency plan, once the warning is received, the SFT should stop the trains and quickly evaluate the situation, setting an initial emergency level (as a value for the *emergencyLevel* variable). To do this, he may use the Control Panel (CTRL) and the Expert System (EXPRT). At this point, the control flow will follow different paths depending on the value of *emergencyLevel*.

When the value of *emergencyLevel* is 1, a sequence of steps begins to confirm that it is a small emergency. As indicated in the sentences B and C, the SFT will request information from the Station Managers (STN) and Train Drivers (TRN) affected by the emergency to allow him to evaluate the magnitude of the emergency. If the information obtained is not enough, an Intermediate Manager (INTR) must make an "in situ" evaluation of the affected area. Finally, depending on the final magnitude of the emergency the SFT proceeds to confirm a Level 1 if it is small emergency or change to Level 2 for those serious emergencies.

Once Level 1 is confirmed, all the actors will proceed to solve the emergency situation with their own available means (extinguishers, first-aid kits, etc.). This coordination process involves relaying appropriate instructions to the respective authorities to assure that passengers safely evacuate. The train driver (TRN), for instance, is in charge of the passengers until the train arrives at the station. At that moment, the station manager on duty is responsible for the second phase of the evacuation. Other actions are also triggered, but we omit them for space reasons.

For Level 2 emergencies, the SFT should use the control panel to do the following: turn the ventilation systems of the affected tunnels on or off (depending on the emergency level); turn off the power; request the intervention of external agents (firefighters, police, etc.). Each one of the actors should act in accordance with Level 2 emergency procedure. This information is reflected in sentence D in Figure 1.

The main disadvantage of the workflow based approach is that workflow languages cannot deal with dynamic and flexible processes. For example, it is not possible (or it is very difficult) to define activities that can be started at any time depending only on data values, regardless of the control flow. This is not very common in the business process field, but it is important in emergency response, where an incident can change its behavior, requiring new actions that are not included in the normal procedure under execution. Current workflow–based languages like BPML or BPEL (Owen and Raj, 2003; Andrews et al., 2003), introduce start conditions based on events that simulate this behavior for certain kinds of activities like process-start activities; however, these conditions are not allowed for many cases and have many semantic restrictions.

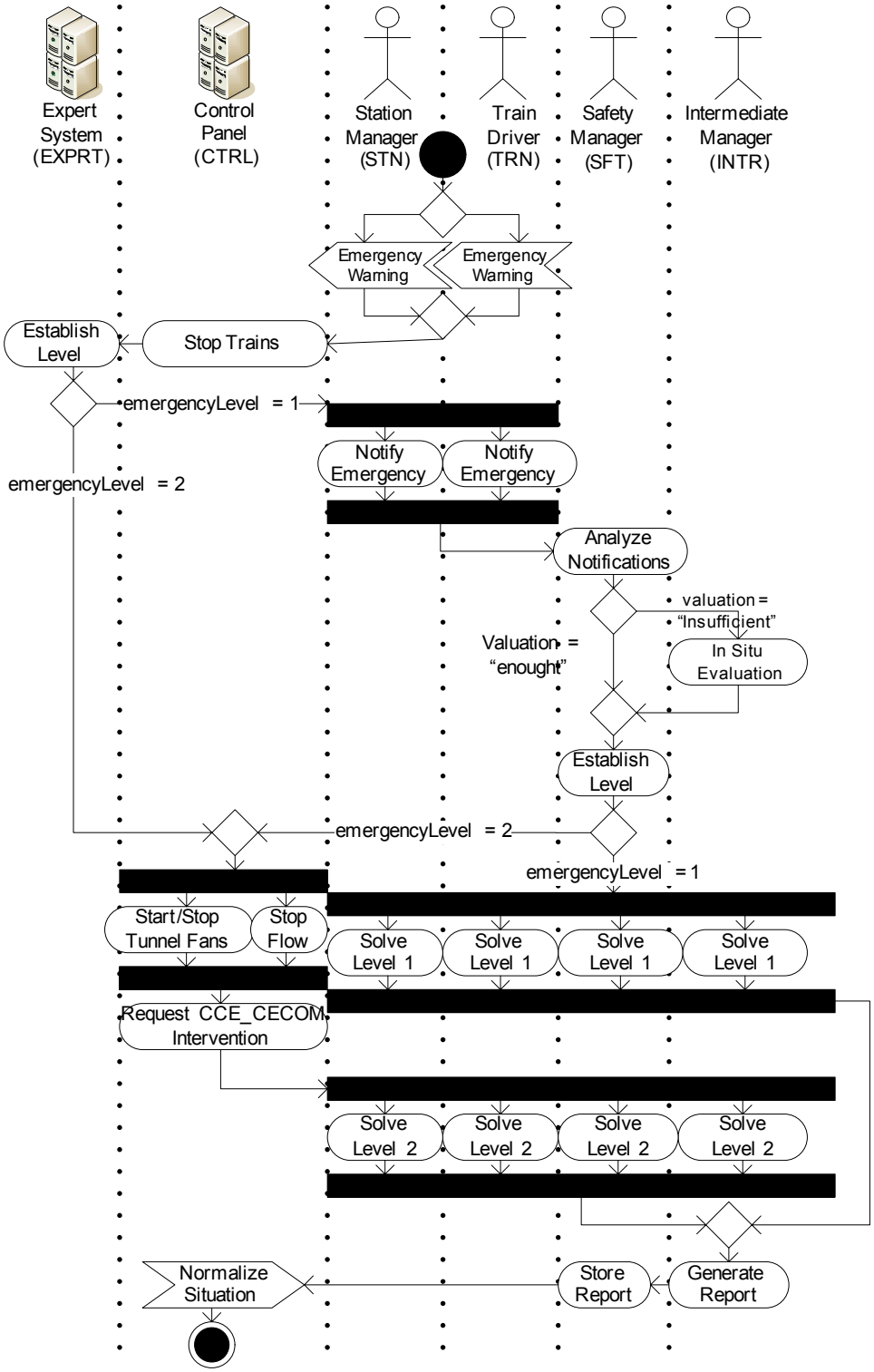

Figure 2. Requested set of ordered activities involved in the resolution of fire in a train emergency

Rule-Based Systems

Propositional Logic was one of the first ways developed to represent knowledge. It is still relevant due to the fact that more sophisticated knowledge representations used by modern knowledge management systems (KMS) can be reduced to that form. Basically, a rule-based system contains an inference engine, a set of data facts, and a set of if-then rules. The inference engine embodies one of the most basic forms of logical deductions, *modus ponens*, which states that if *fact1* is true and the rule *If fact1 then fact2* is true, then *fact2* is also true. The inference engine triggers rules by matching facts to the premises in order to add new data to the facts, until no more rules can be triggered.

Rule-based systems can be viewed as a flexible way to write computer software and, as a particular case, to define dynamic and flexible processes. Many rule-based process definition languages have been proposed recently (Ellis and Keddara, 2000; Wainer, 2000; Kappel, Rausch-Schott and Retschitzegger 2000). In such languages, a process is defined by a set of rules that match the form *state1 then state2,* where *state1* represents the start condition for an activity and *state2* represents the state reached after the activity is executed.

Despite the flexibility of rule-based systems, they are very difficult to manage and frequently long chains of rules often contradict each other. Another disadvantage with rule-based systems is that it is very difficult to control the order in which rules are triggered. If-then paradigms are useful but also have limitations; actually, humans do not use them as a significant mode of thinking. Finally, such systems are difficult to test and visualize, especially when the number of rules is high.

A MIXED APPROACH FOR FLEXIBLE PROCESSES SPECIFICATION

Workflow and Rule-based approaches have limited expressiveness for the case of emergency response. Specifically, both assume that an activity is immediately executed if its start condition is reached. This is commonly the case, but it is not the only possibility; sometimes we may need to describe an activity that *can*, but not *must*, be executed after a condition is reached, or we may need to describe activities that are forbidden in certain states during the emergency. Similarly, sometimes we may need to describe a process fragment with optional execution, that is, a set of activities that must follow certain order of execution and *can* but not *must*, be executed, or we may need to describe sequences of activities that are forbidden.

Since contextual information can influence an emergency response. Flexibility is a key requirement. We propose a solution based on a combination of both workflow and rule based languages that allow the description of flexible processes without loss of precision. A flexible process can be constructed using the following elements:

- **Actor:** any of the entities, either human or resources, that participate in the process
- **Action:** an atomic and instantaneous event that occurs during the process
- **Activity:** an atomic unit of work that is done by an actor. From the point of view of the process management system, an activity involves two actions, one for the activation of the activity and another one for the response of the actor that executes the activity.
- **Process:** set of activities that represent the steps to be followed during the process.
- **Process Variables:** set of name-value pairs that are used to control the process.
- **Process State:** set of values for the process variables at a given moment.

In terms of expressiveness, we consider that a process can be defined from two complementary points of view. At any given moment, there can exist *required* activities (that is, for mandatory execution) and *allowed* activities (that can be executed, although not required). Both types of activities can be specified considering the state of the process (rule-based approach) and/or considering the order of execution of the activities (workflow approach). This is summarized in Figure 3, where four complementary perspectives to define activities in a process are depicted.

Quadrant I: Obligation Processes

Obligation Processes are composed by a set of activities that must be executed in a certain order. This is the common expressiveness of traditional workflow languages and is the common semantics used in natural language emergency plans: "First execute activity A1. Next, execute A2…". Figure 2 shows the graphical representation of the obligation process that represents the emergency plan response procedure (corresponding to the first quadrant).

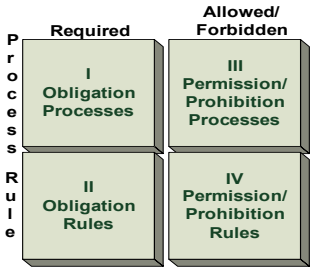

Figure 3. A conceptual framework for Process-Definition

Quadrant I: Obligation Processes

Obligation Processes are composed by a set of activities that must be executed in a certain order. This is the common expressiveness of traditional workflow languages and is the common semantics used in natural language emergency plans: "First execute activity A1. Next, execute A2…". Figure 2 shows the graphical representation of the obligation process that represents the emergency plan response procedure (corresponding to the first quadrant).

Quadrant II: Obligation Rules

Obligation Rules define activities that must be executed when the process reaches a given state. The typical use is the definition of alternative control flows that must be executed under certain situations like a sudden complication in the development of an incident. These kinds of rules are frequently attached to obligation processes to include additional constraints. For example, in the case of a fire above a train the firefighters must be called; Using the process depicted in Figure 2, we can express this situation by adding one obligation rule without having to add this behavior for each activity of the obligation process.

Quadrant III: Permission/Prohibition Processes

This kind of process defines sets of activities that must be executed in a certain order although the execution of the process is not obligatory. As an example, before the execution of the activity "In Situ Evaluation" (see Figure 2), it might be interesting to execute the two ordered activities "Retrieve Historical Information about Incidents in the Affected Zone" and "Send Historical Information to Evaluation Team", whose execution is allowed. If a process is under execution the actions included must follow the established order. This means that actions that are not executed in that order will not be allowed.

Quadrant IV: Permission/Prohibition Rules

These rules allow the definition of actions that are forbidden (or allowed) for execution in certain states of the process without being restricted by any order of execution. With these rules, restrictions or conditions that must be guaranteed throughout the process can be easily defined. For example, in the process in Figure 2, it might be interesting to add a prohibition rule to guarantee that the air pumps in the tunnels are not shut down while the evaluation teams are in the affected zone. By default, an action is allowed if there is not a rule associated to it.

Hence, an emergency plan response procedure could be fully described with the expressiveness provided by these four quadrants that cross obligation/permission/prohibition with processes/rules. It is important to remark that the expressiveness provided by quadrants III and IV is not provided by workflow nor rule-based traditional approaches. The formal language that supports these four quadrants is presented in the following subsection.

Using Dynamic Logic

Dynamic Logic (DL) is a formalism introduced by David Harel for reasoning about programs correctness (Harel, Kozen and Tiuryn, 2000). Its name emphasizes the main feature that distinguishes it from classical predicate logic. In the latter, truth is *static*, that is, the truth value of a formula is regarded as immutable and is determined by a valuation of its free variables over some structure. In DL, however, there are explicit syntactic constructs called programs, whose main role is to change the values of variables, thereby changing the truth values of formulas.

We have used a variant of the Dynamic Logic that incorporates deontic operators as the underlying formalism for our proposal. Deontic Logic (Meyer, 1988) is a modal logic that contains operators that apply potential behaviours

or actions (Vonwright, 1968). This way, we can express the behaviour of the system using three kinds of formulas: state changes, prohibitions, and obligations. In the following description, *a* represents the occurrence of an action, and ψ and ϕ are first order logic well-formed formulas that define part of a process state. Furthermore, ¬*a* represents the non-occurrence of action *a*, in other words, the occurrence of any other action but *a*. Finally, *false* is an atom that represents an unsatisfiable state or, in other words, a state which cannot be reached.

State Change: ψ [a] ϕ

"When a state satisfies ψ, ϕ must be satisfied immediately after the occurrence of action *a*"

Prohibition: ψ [a] *false*

"The occurrence of action *a* is forbidden for the states where ψ is satisfied"

Obligation: ψ [¬a] *false*

"The occurrence of action *a* is required in the states where ψ is satisfied"

In our combined approach with the Dinamic Logic, obligation and permission/prohibition rules (quarters II and IV) are automatically mapped to obligation and prohibition formulas, respectively. A permission rule is equivalent to a negated prohibition formula. In (Letelier, Sánchez and Ramos, 1998) we defined an automatic way to obtain a set of obligation (prohibition) and state change formulas that are equivalent to an obligation (permission/prohibition) process corresponding to the quadrants I and III. The expressive capabilities provided by the framework described above can be uniformly interpreted and executed as Dynamic Logic formulas, providing us a formal framework for the description of more expressive procedures than other current approaches; moreover, it allows the validation, verification, simulation and other features that are needed in order to manage emergency scenarios.

The following section describes a case study taken from the emergency plan of Metro Valencia's subway Emergency Procedures and shows how easy the description of the resolution procedure is following this new approach.

A CASE STUDY

In this section, we show the specification of the response to a fire in the subway based on the Emergency Procedures partially shown in Figure 1. After identifying the actors and the variables of the process, we specify the process using the framework presented above.

Actors and State Variables

From the description in Figure 1, we identified four human actors involved in the resolution of the emergency. The Safety Manager is responsible for the coordination of the actors that participate in the resolution of the emergency. The Station Manager (STN) is in charge of communicating the incidents that occur in the areas under his or her control, as well as collaborating with the Safety Manager (SFT) in the evaluation of the level of severity of the emergency. The Train Driver (TRN) should report on the train status, as well as collaborate in the assessment of the emergency if required. Finally, the Intermediate Manager (INTR) is responsible for an area and collaborates in the evaluation "in situ" when the SFT does not have enough information in order to evaluate the severity of the emergency.

Besides the above-mentioned actors, it is assumed that the SFT has two automatic emergency support systems: the Control Panel (CTRL) and the Expert System (EXPRT). The CTRL controls the electric systems available in the subway network (the ventilation of the tunnels, the emergency stop system of the trains, the flow of electric power, and others). This panel is integrated with the Fire Extinction and Communication Center systems network. The EXPRT is trained to classify the level of severity of the emergencies depending on whether external resources (Fire Extinction Services) are required or not.

We will use a variable named *emergencyLevel* to represent the severity of the emergency; the possible values are 1 for small emergencies and 2 for serious ones.

Regular Process (Quadrant I)

The regular process or obligation process in our approach was presented in Figure 2 using a UML 2.0 Activity Diagram. This process is obtained from de Emergency Procedure (Figure 1).

Following subsections present three situations (one for each of the other quadrants) which are not included in the specification of the Figure 2. These situations can be represented as complementary specifications using our approach. To integrate the specification of obligation processes with the rest of the specifications, we will use an internal representation (hidden from the analyst) based on a state diagram. In this state diagram, each activity is represented by two states, one reflecting that the activity is started and another indicating that the activity is finished. Figure 4 is the state diagram corresponding to the specification in Figure 2. This state diagram is obtained automatically. The complexity of the state diagram can be hidden by means of graphic facilities in a visual editor based on the process model in the Figure 2. This way, permission and prohibition rules can make reference to the states derived in Figure 2 to define the starting conditions.

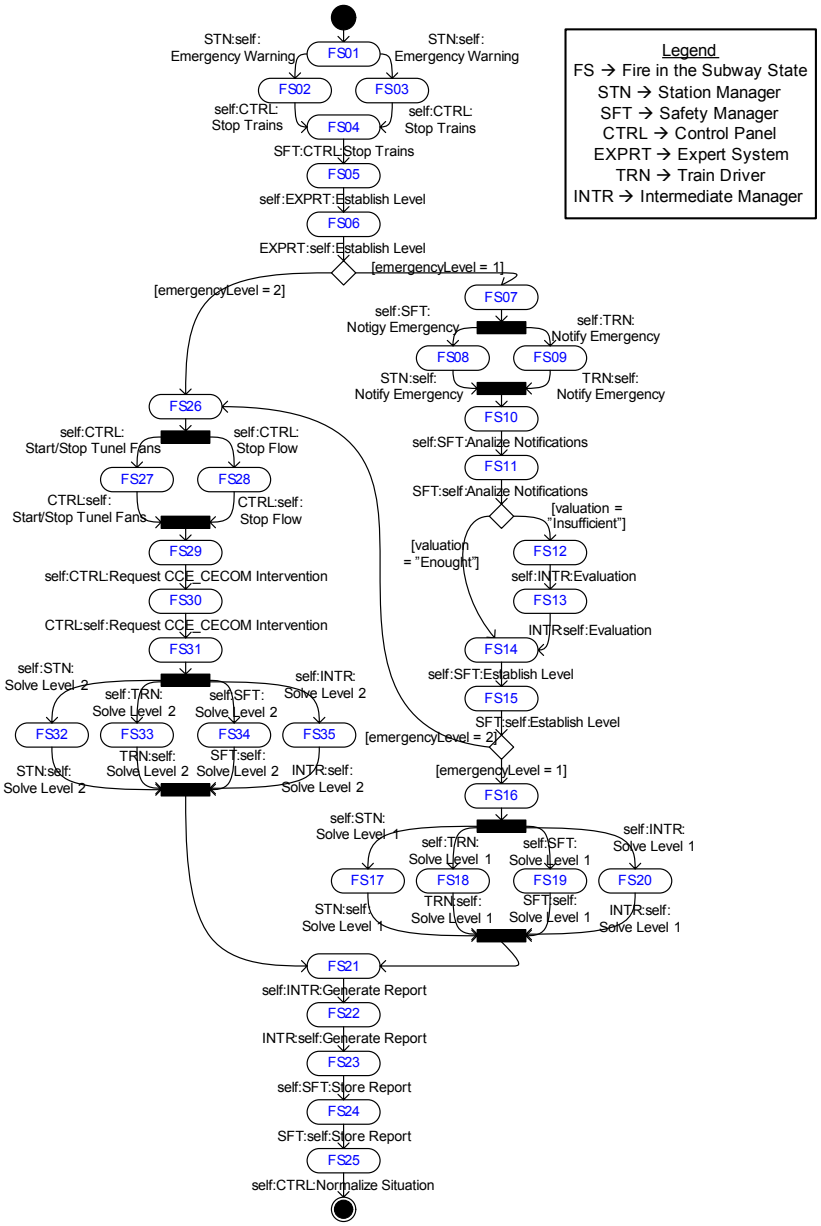

Figure [0]**4. Equivalent State Diagram to the Process definition in Figure 2**

Emergency Level Correction (Quadrant II)

After analyzing the description of the emergency, if Expert System (EXPRT) establishes an initial Emergency Level of 1, a group of activities begins to verify that it is a small emergency. What would happen if after receiving more information from the Train Drivers (TRN), the Safety Manager (SFT) detects that it is in fact a more serious emergency? As described in Figure 2, after establishing an initial emergency level with a value of 1 the SFT cannot change to Emergency Level 2 until the control flow activates "Establish Level" activity and therefore the SFT is forced to carry out all the activities in order to verify that it is a small emergency even though it is known to be a serious one. To respond correctly, immediately after detecting an error in the prediction of the expert system, the SFT should correct the Emergency Level starting the "Start/Stop Ventilation" and "Shut down Electric Flow" activities to manage the level 2 emergency.

To define this behaviour following the workflow approach, we would have to add a bifurcation in each activity with a conditional activity that validates whether or not a change in level has been made or we would have to add complex and very restricted exception handling, complicating the specification. Using the rule-based approach it's mandatory to add auxiliary variables that greatly complicate very much the specification.

Following our approach, starting from a representation like that shown in Figure 2, the following steps should be followed:

1. The analyst defines the state change formulas that represent the correction carried out by SFT:

   [CorrectToLevel2] emergencyLevel = 2;

2. The analyst indicates the activities in which a change to level 2 can take place; in this case, "Notify Emergency", "Analyze Emergency", "In Situ Evaluation" and "Establish Emergency Level". The start condition is that the emergency is in one of the states corresponding to these activities (states FS07..FS15) and that the condition emergencyLevel = 2 is satisfied. This way, the obligation formula that would be added would be the following:

   (FS in (FS07, FS08, FS09, FS10, FS11, FS12, FS13, FS14, FS15)
   and emergencyLevel = 2)

   [¬ ChangeToLevel2] *false*;

3. The analyst indicates the destination activity, to which the control flow should change when the initial condition is satisfied, just before the parallel execution of the activities "Start / Stop Ventilation" and "Shut down electric flow". The corresponding state change formula is:

   [ChangeToLevel2] FS = FS26;

Recognition without electric flow (Quadrant IV)

In other situations, it might be interesting for us to limit the execution of certain activities depending on the value of the state variables. For example, we must ensure that the Intermediate Manger (INTR) carries out evaluations of the affected area when the electric flow is down since, otherwise, there is danger of electric shock to the operatives.

These cases belong to quadrant IV and are modelled with prohibition rules. In a way similar to the one used with the obligation rule above, the analyst would indicate: the activities for which we want to specify the prohibition, in this case "In Situ Evaluation" (FS12 of Figure 4); and the condition that should be accomplished (electricFlow = off). The final rule is:

¬ (FS = 12 and electricFlow = off) [InSituEvaluation] false;

In this case, InSituEvaluation can only be executed when the process is in state FS12, following the workflow approach, it might be possible to get the same behaviour by adding a precondition for this activity. However, in the case that InSituEvaluation could be executed in several places during the process, this rule will ensure that all occurrences of this activity will be executed without electric power.

Optional Ordered Activities (Quadrant III)

During the resolution of the emergency, any actor can request information on the state of the emergency. This application of information implies the sequence of activities that is illustrated in Figure 5: Initially an actor requests information on the emergency; the SM should obtain the required information and return the answer to the actor that requested the information.

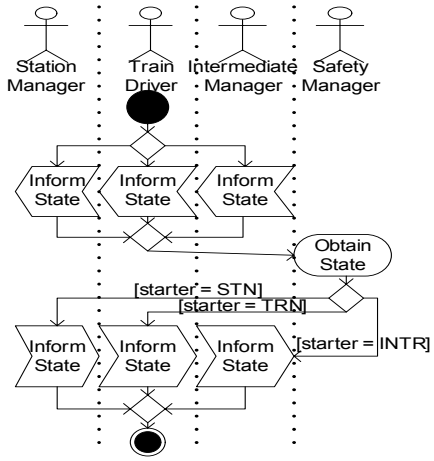

Figure **[0]**5. Optional Execution Process to inform about emergency state

The Process in Figure 5 is specified separately from the process shown in the Figure 2. Its execution is optional using the permission/prohibition processes of quadrant III. If the restriction of the execution of the process in Figure 5 is desired, prohibition formulas (like in Recognition without flow) could be added based on the states to be restricted.

CONCLUSION

The response to an emergency is the enactment of a process. Unlike other domains, the high probability of occurrence of exceptional and complex scenarios requires some flexibility which is not provided by current workflow management systems. Rule-based solutions provide flexibility, but complex models are difficult to represent and manage.

In this work, we have presented an approach that combines workflow and rule-based languages to provide the expressiveness needed to specify flexible Emergency Procedures. We use workflow languages to specify well understood and stable procedures, whereas we use rules just to define flexible behavior. Both workflow models and sets of rules are translated to a variant of Dynamic Logic. This formal support offers better conditions to validate and verify the specification. This also facilitates the application of more sophisticated operations like automated simulation or prototype generation.

We are developing a tool for the specification and validation of emergency response procedures based on our proposal. This will be integrated in a complete environment for the development of emergency management systems.

One of the problems of our proposal is that Dynamic logic is a formal language that is not adequate for human processing. A suitable visualization of the plan by the emergency experts is required. Further work should include visualization models for understanding and validation of the plan.

REFERENCES

1. W.M.P. van der Aalst and Weskez, M. (2003) Advanced Topics in Workflow Management: Issues, Requirements and Solutions, Journal of Integrated Design and Process Science.
2. Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Laymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S. (2003) Business Process Execution Language for Web Services.
3. Canós, J.H., Alonso G. and Jaén, J. (2004) A Multimedia Approach to the Efficient Implementation and Use of Emergency Plans, IEEE Multimedia. Vol. 11, no. 3, pp.106-110.
4. Ellis, C. and Keddara, K. (2000) ML-DEWS: Modelling Language to Support Dynamic Evolution within Workflow Systems, Computer Supported Cooperative Work.
5. Harel, D., Kozen, D. and Tiuryn, J. (2000) Dynamic Logic, MIT Press.
6. Jablonski, S. and Bussler, C. (1996) Workflow-Management: Modelling Concepts, Architecture and Implementation, International Thomson Computer Press.
7. Kappel, G., Rausch-Schott, S. and Retschitzegger, W. (2000) A framework for workflow management systems based on objects, rules and roles, ACM Computing Surveys Symposium on Object-Oriented Application Frameworks.
8. Letelier, P., Sánchez, P., Ramos, I. (1998) Especificaciones de proceso para objetos y su representación en Lógica Dinámica, Proc. III Jornadas Ingeniería del Software (in Spanish)
9. Metro Valencia Safety Office (1998) Emergency Plan, Internal Document (in Spanish).
10. Meyer J. (1988) A different approach to deontic logic: Deontic logic viewed as a variant of dynamic logic, Notre-Dame Journal of Formal Logic, vol. 29.
11. Object Management Group (2005) UML Specification Superstructure 2.0, www.uml.org.
12. Owen, M. and Raj, J. (2003) BPMN and Bussines Process Management, Popking Software.
13. Vonwright, G. H. (1968) An essay in deontic logic and the general theory of action, In Acta Philosophica Fennica, vol. 21.
14. Wainer, J. (2000) Logic representation of processes in work activity coordination, Proc. ACM Symposium on Applied Computing, Coordination Track (1).
15. Workflow Management Coalition (1995) The Workflow Reference Model, www.wfmc.org.