Integrating Business Processes into GIS-based Simulations

Marcelo G. Metello, Eduardo David, Marcelo T. M. de Carvalho, Marco A. Casanova

Grupo de Tecnologia em Computação Gráfica – Tecgraf
Departamento de Informática – PUC-Rio
Rio de Janeiro, RJ – Brazil

{metello,david,tilio}@tecgraf.puc-rio.br, casanova@inf.puc-rio.br

Abstract. *This work proposes a mechanism to integrate different types of processes into a common gis-based simulation engine and, in particular, how to handle business processes described as workflows. In this sense, the paper presents the simulation process approach adopted, describes a method for mapping a workflow into the simulation process and it identifies which information needs to be added to the workflow description to make it possible to simulate it. To illustrate the ideas proposed, the aspects regarding management of an emergency situation are discussed. The discussion considered the different processes involved with it and the use of computational simulation to help manage the emergency situation.*

1. Introduction

In the Geographical Information Systems (GIS) field, a considerable amount of research has been devoted to developing dynamic models of man-driven and earth phenomena, such as physical processes, land use cover change, traffic control and socio-economic dynamics, among others. Many formalisms were developed to express these dynamic models. Among the most popular, we may quote cellular automata [von Neumann 1966], system dynamics [Forrester 1961] and multi-agent systems [Michel et al. 2009]. These models are used to predict the outcome of given scenarios and can be used to help planning, for example, urban activity, logistics, emergency response or marketing strategies.

Organizations have also worked out ways to formally express their activities. The adoption of the so-called Business Process Management (BPM) systems has grown significantly over the last few years [Weske 2007]. The most common practice of BPM tools is to express business processes as workflows. However, most BPM solutions focus on analyzing, publishing and monitoring business processes. There has been little research on simulating these processes, which help verifying and improving them.

Sometimes it is relevant to consider interaction between external processes, such as physical simulations, with organizations business processes, such as technical procedures. One example occurs during an emergency situation, such as an oil leak accident on the sea, where the behavior of the oil spill is influenced by the response team actions and the way the command of the emergency handle the situation depends on how big the oil spill is. In order to simulate such situations and obtain more realistic results, this type of interference must be considered.

Based on this idea, a simulation engine has been developed to provide such mechanism to deal with different processes that can interact with each other. In a previous work [Metello et al. 2008], it was presented an earlier version of this engine, which could simulate some kinds of processes in a GIS environment, such as physical dispersion of leaked products and transportation of equipment. Even though this engine could simulate a large variety of processes commonly modeled in GIS, it could not simulate workflows based processes. This paper addresses a newer version of the simulation engine, which can simulate business processes described as workflows. The paper has two main objectives. First, it describes a method for mapping a workflow into a simulation process to be executed in the engine. Second, it identifies the information required to be added to the workflow description that enable it to be simulated.

This paper is organized as follows. Section 2 illustrates in more detail the motivating example of handling emergency situations. Section 3 describes a formal base for integrating processes of different nature together in a simulation environment. Finally, section 4 describes how a process described by a workflow can be simulated in that environment.

2. A Motivating Example: Emergency Planning

An emergency occurs when a sudden and unexpected situation may cause damage to human health and/or to the environment and properties and which, therefore, requires immediate action and specific resources. In this case, it is not only important to respond quickly, but the response must be conducted in a well organized manner, in order to minimize the damages. The complexity of emergency management, coupled with the growing need for multi-agency and multi-functional involvement on these situations, has increased the need for methodologies that follow standards and that can be used by all emergency response disciplines. In this sense, the use of the *Incident Command System* (ICS) [Bigley and Roberts 2001] standard by public safety and private sector organizations around the world has been increasing.

When the incident begins, the first steps are notifications, initial assessment, command meeting, initial response and incident briefing using specific ICS forms. After this initial response period, the emergency handling process becomes cyclic. Each cycle, called *Planning "P",* consists of a planning and an operational phases. The planning phase include meetings for assessing the current situation, update objectives, tactics definition, preparing and approving the incident action plan (IAP). The operational phase is where the response plan is executed and its progress is evaluated, after which starts a new cycle.

Use of information systems, such as InfoPAE [Carvalho et al. 2001] help to improve the management of this process. InfoPAE is an automated system designed to improve the response to emergency situations. It also proved to be a valuable training tool. The system offers sophisticated action plans and easy access to vital information and resources allocated for different types of scenarios. The system is developed by Petrobras (the Brazilian oil company) and the Computer Graphics Technology Group (Tecgraf) at PUC-Rio.

One of the difficulties of such systems is that, even though it is possible to describe an emergency action plan at a reasonably detailed level, this is somewhat limited with respect to the representation of dynamic aspects. The action plan is usually

described using workflows and most workflow representations do not specify the time at which an action should be executed, how long it will take to finish and how the environment is affected by its execution. For instance, if a plan has instructions to install barriers to contain an oil spill at some point, it should take into account where and how the spill moves and how long it will take to have the barriers installed at the correct point. Otherwise, the plan is useless.

Testing the quality of an action response plan and the performance of the emergency response team is mandatory to ensure minimum impact of the incident. Beside others initiatives such as field exercises, use of computational simulation can be a cost-effective mechanism to improve it. Simulation can be used for different purposes as to estimate, in advance, if there will be enough resources and time for executing a specific action or support complex decisions during the planning phase of emergency response. One of the common used for it is to simulate the behavior of physical phenomena such as those involving dispersion of chemical products in the environment. However, the pure simulation of physical dispersion does not take into consideration the effects of contingency actions. Therefore, the use of purely physical simulation is somewhat limited in a strategy which cycles between evaluation and planning.

Hence, a more suitable computational environment to test the emergency management would be one that could combine simulation of physical phenomena with others processes such as those related to *Planning "P"*.

3. Process Simulation

3.1 Motivation

In the long-lived field of simulation, many different formalisms were developed to model processes in time [Eker et al. 2003, Vangheluwe 2000, Zeigler et al. 2000]. Each formalism suits better some specific class of processes. For instance, the dispersion of some leaked product into the environment can be modeled as a cellular automata (CA), while the corresponding contingency action plan can be modeled as a workflow. These two types of processes are depicted respectively in Fig. 1 (a) and (b). The first process defines a sequence of states over time while the second represents a set of actions from a workflow being executed in time, where each individual action has a defined start and end time instants. The problem of simulating these two processes independently of each other is obvious. Each of them only represents a partial view of the problem. The physical dispersion simulation would not take into consideration the effects of contingency actions. Likewise, the elaboration of plans as mere workflows does not guarantee that the actions will have their preconditions for execution met and, even if executed, that they will produce the desired results. Again, one illustrative example would be a plan that contains one action for placing contention barriers for oil leaked into the sea without considering whether the teams will have the necessary time to do so. These processes represent partial views precisely because they only consider a subset of all the elements involved.

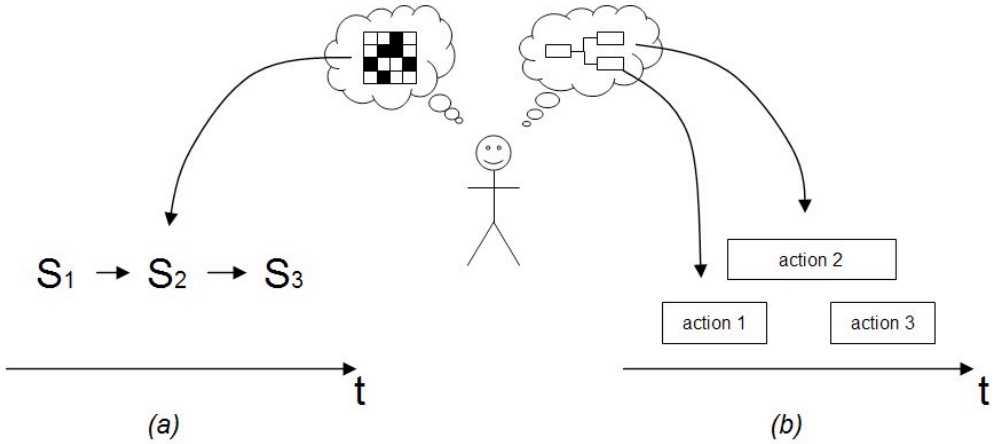

Figure 1. (a) A cellular automaton process represented by a succession of states over time. (b) A **workflow process represented by a set of actions in time**.

Our approach for dealing with this problem is to integrate these isolated processes into a single simulation execution environment, even if they are expressed in different formalisms. Naturally, the simulation must consider all the interferences between them. In order to achieve this, it is necessary to provide a common high-level formalism capable of representing different kinds of processes, such as plans and physical simulations. Also, the simulation environment must be capable of executing interfering processes in parallel.

The most straightforward way to integrate a number of different processes in a simulation environment probably is to run all of them in parallel, propagating causality every time a process generates an event that will possibly affect another process. Fig. 2 depicts two interfering processes being simulated in parallel. The sinuous arrows indicate causality being propagated between those processes.

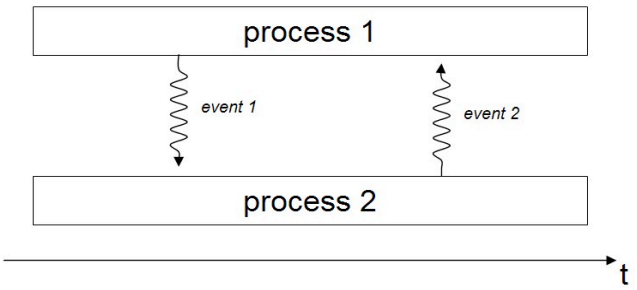

Figure 2. Two interfering processes being executed in parallel.

Three problems must be addressed to integrate processes of different natures. First, we need a common formalism which must be general enough to express the behavior of all individual processes as well as the interferences between them. Second, we need to map the formalisms in which the individual processes are described to this general formalism. Lastly, we need a simulation environment to execute all processes together.

3.2. On the Process Definition Formalism

Abstractly speaking, a process represents change in time. In the simulation field, a wide variety of formalisms to model change in time have been proposed. These formalisms may differ from each other even in the most fundamental aspects such as time representation. Some of them use a continuous numeric scale while others are restricted to discrete time values. They also differ with respect to how they represent changes in the world state with time. Again, some represent changes as discrete instantaneous events while others are able to represent continuous changes by means of differential equations. A complete discussion however is out of the scope of this work. Instead, we shall just point out that generality and the ability to integrate different formalisms is what is important to achieve our goals. Previous research about simulation formalisms have suggested that the *discrete event* approach has good potential for serving as the basis for integrating different simulation formalisms [Vangheluwe 2000, Zeigler et al. 2000]. Therefore, the principles of the discrete event approach were used as basis for the formalism designed to describe processes in a general way.

Each process evolves in a continuous timeline in which instantaneous events cause changes to the internal state of the process. Additionally, a process can generate events that are sent to other processes that are coupled to it. The coupling structure is defined separately from the definition of the processes behavior, thus increasing modularity and reuse. The coupling structure is also dynamic and can be changed during the execution of the simulation. Changes made to the coupling structure are also instantaneous in time, so it can be treated in the same way as a regular event.

Another aspect that was included in the approach is the lifetime of a process. During the course of a simulation, new processes can be started and existing ones can finish their executions. Each process must have definite start and end times. A process can only alter the state of the world within its lifetime. In consonance with other types of changes in the simulation, the start of a new process and the finish of an active one are also treated in the same way as other instantaneous events.

In order to allow GIS-based simulation, a notion of a *GIS environment* is created as a spatio-temporal data repository which represents the physical world. It has a state that changes instantaneously at some points in time, as the simulation advances and new events alter some geographical feature. Fig. 3 shows two processes and their interaction with the GIS environment, as well as with each other.

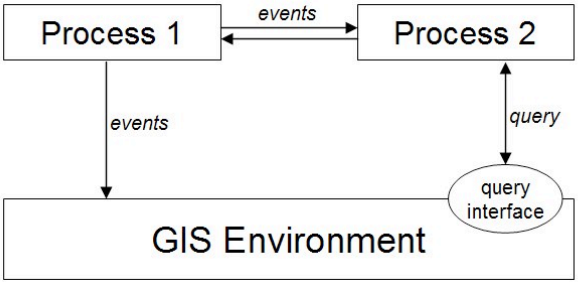

Figure 3. Two processes interacting with the GIS environment.

Because of the generality of the discrete event approach to simulation, it is usually possible to map other dynamic models into it [Zeigler 2000]. Discrete time models such as cellular automata, continuous models such as system dynamics and other hybrid approaches can be implemented on top of the discrete event approach [Vangheluwe 2000]. This makes it possible to reuse most models developed in the GIS field such as physical phenomena, social and economical behaviors and transportation, among others. The idea is to integrate all the relevant dynamic models into one synchronized set of processes and, on top of that, add some extra processes representing the business processes of an organization in that scenario. Then, it will be possible to the organization to foresee the effects of any given course of action.

4. Mapping Workflows to Discrete Event Simulation Processes

In this section, we describe in detail how to map workflow processes into a discrete-event process formalism. This is relevant since workflows are the most common way to represent business processes, as mentioned before, and since we are particularly interested in how organizations may create action plans. It is out of the scope of this paper to discuss how other types of processes are mapped into the discrete event process formalism.

4.1. Workflow Definition

Since there are many different languages and representations for workflows, this section starts by describing the workflow definition used here.

A workflow defines a set of actions and a control flow which imposes restrictions on the order at which the actions should be executed. With respect to their control flow, workflow representations may differ from each other in some of the patterns they use. For the sake of simplicity, we shall consider only the basic patterns defined in [van der Aalst et al. 2003]. Those are *sequence*, *parallel split*, *synchronization*, *exclusive choice* and *simple merge*. Fig. 4 depicts these patterns.

Each basic pattern connects some *preceding actions* to some *following actions*. In Fig. 4, all preceding actions are represented as rectangles on the left side of each pattern and all following actions are placed on the right side. In some cases it makes sense to connect a pattern directly to another pattern instead of an action. However, considering a direct connection between two patterns, placing a dummy action between them will not alter the behavior of the process described by the workflow. In fact, the two will be equivalent. Therefore, it is enough to describe the behavior of the patterns considering that they are only connected to actions. Table 1 lists some of the characteristics of each basic pattern.

The number of preceding and following actions indicates the number of action connections each pattern may have. The trigger time indicates the condition upon which the pattern is triggered. Finally, the following activity describes which of the following actions should be started when the pattern is triggered.

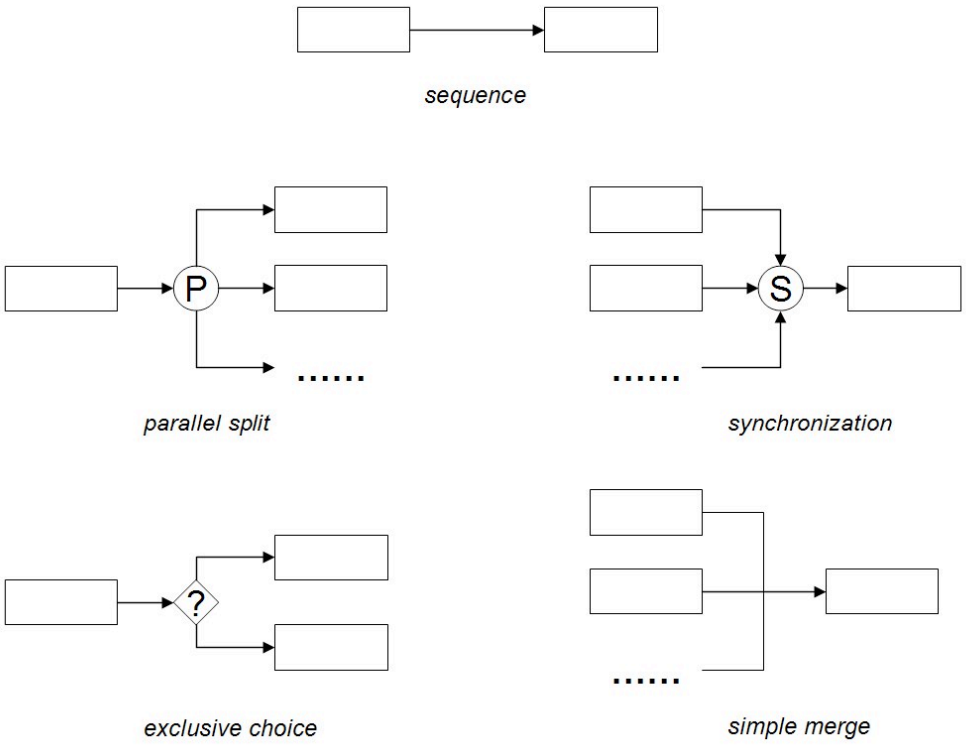

Figure 4. Basic workflow patterns.

Table 1. Properties and behaviors of basic patterns.

pattern	number of preceding actions	number of following actions	trigger time	following activity
sequence	1	1	completion of preceding action	start following action
parallel split	1	n	completion of preceding action	start all following actions in parallel
synchronization	n	1	completion of all preceding actions	start following action
exclusive choice	1	2 (or n)	completion of preceding action	start one of the following actions
simple merge	n	1	completion of any preceding action	start following action

Optionally, the parameters or inputs of the individual actions are also represented. Likewise, an action may also produce some data as output. That data could be consumed either by another action executed after it or by some conditional split operator in the control flow. Therefore, in order to represent action input and output, a

data flow must also be represented. It is important to notice that the data flow does not follow the same paths as the control flow. However, it should obviously obey the ordering restrictions imposed by it since an action cannot consume output data from another action that has not yet been executed.

The simplest way to model the data flow is to define a set of variables that will compose the internal state of the workflow process. These variables are accessible by any action. Every time an action or a control operator needs some input data, it can get it from one or more variables. Whenever an action produces some data, it should store it in variables so that later actions can read it. Information stored in variables can also be used by exclusive choice operators to evaluate their conditions when they are triggered. Fig. 5 illustrates a workflow with the basic patterns and some variables.

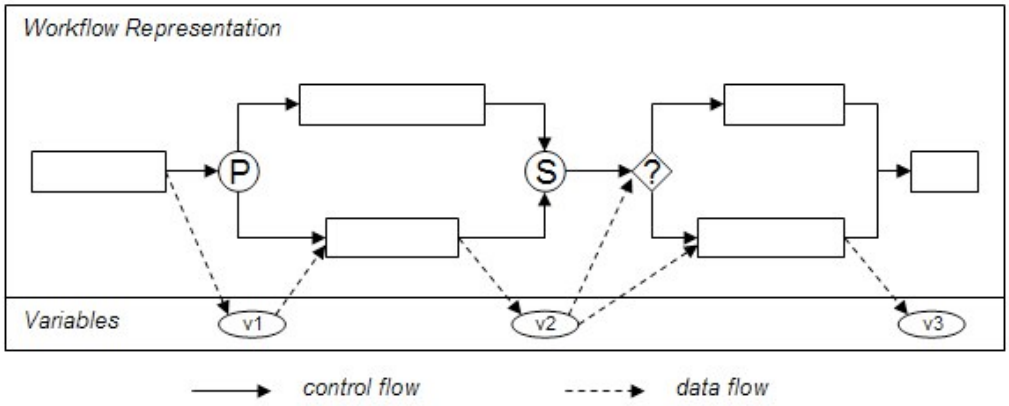

Figure 5. Workflow example with control and data flows.

4.2. Workflow as a Simulation Process

In our approach, we introduce a simulation process to represent the workflow, which we simply call the *workflow process*. This process will fork one individual *action process* every time it executes one action, as illustrated in Fig. 6. For that reason, the workflow process must be able to indicate the start time for each of its actions. Likewise, it needs to receive back the exact finish time of each action process so that the workflow can continue executing the next steps.

For any of the basic patterns, the start time of its following actions is given by a function that takes as input the trigger time of the pattern and, optionally, some values from variables. Actions that are not following actions of any pattern are started at the beginning of the workflow execution. When an action finishes its execution, it must inform the workflow process that it has finished by sending an event to it. That event may cause another pattern to trigger and so on until there are no more actions to execute. The finish time of the workflow process is the same as the finish time of its last executed action. The flow of all this timing information is illustrated in Fig. 6 as dotted arrows. As in the previous section, continuous arrows indicate the control flow and the dashed arrows indicate the data flow.

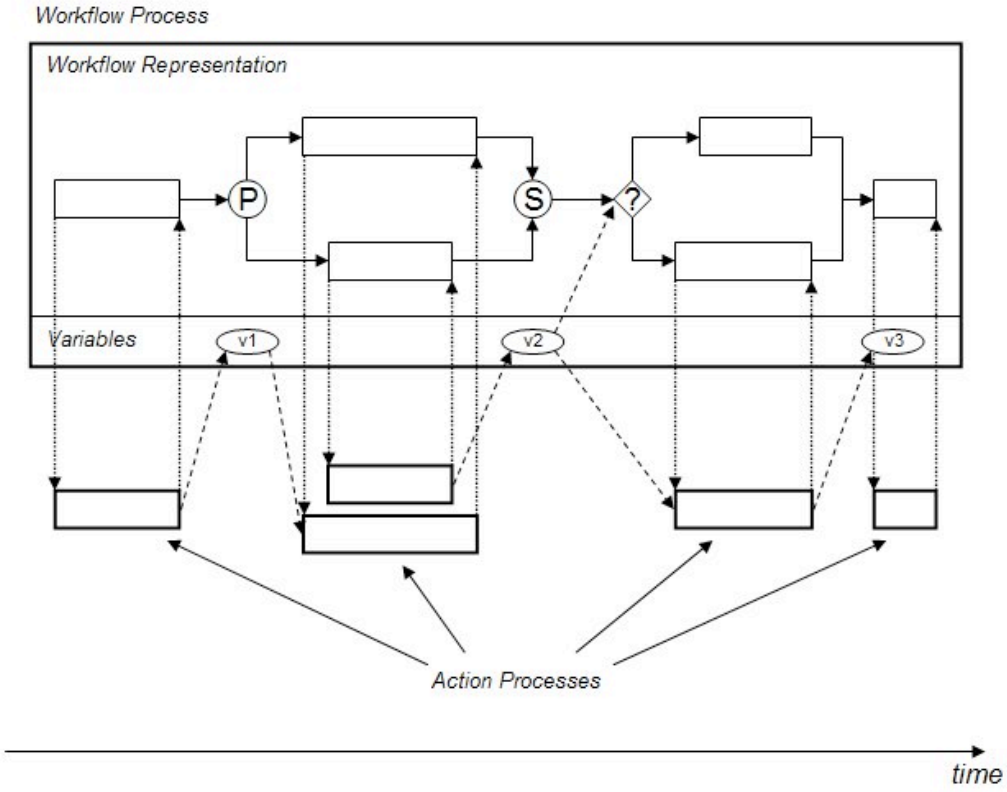

Figure 6. Processes involved in the simulation of a workflow.

Now that the processes to simulate a workflow are defined, we can combine them with processes of a different nature and consider how they interfere with each other. It is worth noting that the workflow variables could be updated by events sent by some process other than the action processes. This is actually a way for a process of a different nature to interfere with a workflow process. Likewise, an action process may send events to processes other than the workflow process, therefore interfering with them.

With the definitions presented in this section, it is possible to identify all the requirements of a workflow representation necessary to simulate a workflow together with other interfering processes. The requirements are summarized in the following list:

- **Provide all actions with typing information so that they can be associated with simulation processes.** Some human-readable workflow representations define the information about what an action does textually. In order to allow automatic simulation, actions must be more strongly typed. Besides, some additional information may be needed providing more detail for enabling simulation. For example, describing the weather conditions as “low tide with south wind” may not be precise enough for a physical simulation.
- **Provide functions to specify the start time of actions and condition evaluation.** As seen before, this is essential to simulate the situation correctly.

- **Provide a way to receive the end times of the actions.** Specific events may be created for providing the workflow process with information about when its executed actions have finished their execution.
- **Provide the workflow process with all necessary information so that it can keep its variables up to date.** By sending events to it, other processes may provide the workflow process with information it needs for correct simulation.

5. Conclusion and Future Work

The paper discusses aspects regarding the integration of different types of processes into a common simulation engine and, in particular, how to handle business processes described as workflows. In this sense, it presents the simulation process, describes a method for mapping a workflow into the simulation process and it identifies which information needs to be added to the workflow description to make it possible to simulate it.

In order to illustrate the ideas proposed, the aspects regarding management of an emergency situation are discussed. The discussion considered the differents processes involved with it and the use of computational simulation to help manage the emergency situation and . Section 3 discussed the use of discrete event approach to simulation, which proved to be a good way of integrating different processes and Section 4 showed how to map workflows into the simulation processes. Finally, it is discussed what a workflow representation must provide in order to enable the engine to simulate it.

As for future work, one possibility is to consider other wofkflow patterns. Indeed, Section 4.1 considered only five patterns, whereas [van der Aalst 2003] defines another fifteen patterns.

Another interesting challenge is that some workflow representations allow the definition of sub-workflows in a hierarchical composition. The approach could be extended to simulate a workflow hierarchy.

The requirements listed in Section 4.2 provide a roadmap to add some simulation functionality to an existing BPM system. For example, the InfoPAE system features a workflow language called XPAE [Casanova et al. 2002] that facilitates defining action plans. However, the current version of the language is not able to consider the dynamic aspect of a real emergency. Extend the language with the requirements presented would be extremely valuable.

References

Bigley, G.A., Roberts, K.H. (2001) ”The Incident Command System: High-Reliability Organizing for Complex and Volatile Task Environments“, The Academy of Management Journal, 44(6), pp. 1281-1299.

Carvalho, M.T., Freire, J., Casanova, M.A. (2001) “The Architecture of an Emergency Plan Deployment System”, In: Proc. III Brazilian Symposium on Geoinformatics, Rio de Janeiro, Brazil.

Casanova, M.A., Coelho, T.A.S., Carvalho, M.T.M., Corseuil, E.T.L., Nóbrega, H., Dias, F.M., Levy, C.H. (2002) “The Design of XPAE – An Emergency Plan

Definition Language", In: Proc. IV Brazilian Symposium on GeoInformatics, Caxambu, MG, Brazil.

Eker J., Janneck J.W., Lee E.A., Liu J., Liu X., Ludvig J., Neuendorffer S., Sachs S., Xiong Y. (2003) "Taming heterogeneity - the Ptolemy approach", In Proceedings of the IEEE, 91(2).

Forrester, J. (1961) "Industrial Dynamics", Cambridge, MA: MIT Press

Metello, M., Casanova, M.A., Carvalho, M.T.M (2008) "Using Serious Game Techniques to Simulate Emergency Situations", In: Proc. X Brazilian Symposium on GeoInformatics, Rio de Janeiro, Brazil.

Michel, F., Ferber, J., Drogoul, A. (2009) "Multi-Agent Systems and Simulation: A Survey from the Agent Community's Perspective", In A. Uhrmacher and D. Weyns (Eds.), Multi-Agent Systems: Simulation and Applications, Taylor and Francis, pp 3-51

van der Aalst, W.M.P., der Hofstede. A.H.M., Kiepuszewski, B., Barros, A.P. (2003) "Workflow Patterns", Distributed and Parallel Databases, 14(1): 5–51(47).

Vangheluwe, H.L. (2000) "DEVS as a common denominator for multi-formalism hybrid systems modeling", IEEE International Symposium on Computer-Aided Control System Design, pp 129-134, Anchorage, Alaska.

von Neumann, J. (1966) "Theory of self-reproducing automata", Illinois: A.W. Burks.

Weske, M. (2007) "Business Process Management - Concepts, Languages, Architectures", Springer.

Zeigler, B.P., Praehofer, H., Kim, T.G. (2000) "Theory of Modeling and Simulation", Academic Press, San Diego, CA.