Collaborative Coordination of Activities with Temporal Dependencies

Jörn Franke, François Charoy, Paul El Khoury

Collaborative Coordination of Activities with Temporal Dependencies

Jörn Franke[1,2], François Charoy[2], and Paul El Khoury[1]

[1] Public Security, SAP Research Center (Sophia Antipolis), 805 Avenue du Docteur Maurice Donat, BP1216-06254 Mougins, France
{joern.franke,paul.el.khoury}@sap.com
[2] LORIA-INRIA-CNRS, Université de Lorraine, BP 239-54506 Vandoeuvre-lès-Nancy Cedex, France
charoy@loria.fr

Abstract. Business process management and systems have been proven mature and highly beneficial in many domains. Recent surveys by leading market analysts say that the next challenge for BPM are unstructured processes. Based on a domain study in disaster response management, we identify current shortcomings of business process models and management with respect to unstructured processes. We develop a generic model for flexible temporal ad-hoc coordination of activities. Its focus lies on awareness and feedback as well as loosely structuring the process with temporal dependencies. It is implemented as an extension to the open Google Wave collaboration infrastructure. The approach is commented by commanders in the disaster management domain.

1 Introduction

Highly dynamic scenarios challenge existing technologies for managing processes performed in the real world. According to Gartner and McKinsey the management of activities in unstructured processes becomes more and more important for many organizations [1]. Two key characteristics have been identified by them. Firstly, they are predominantly executed by an individual or group in a dynamic fashion. Secondly, they are highly dependent on the interpretation, expertise and judgment of the humans doing the work for their successful completion. It implies that people work collaboratively on single activities and that dependencies between activities are also coordinated by people in a dynamic and ad-hoc fashion. Different participants of different organizations do not have necessarily common goals, but their efforts need to be unified and synchronized by providing awareness and feedback mechanisms. Disaster management is a special case of a domain that requires the management of highly dynamic unstructured processes to coordinate people and teams from multiple organizations distributed between different command centers and the field. Results can be seen as an important contribution to the BPM community [2,3,1]. Our research is grounded in a study of process management within the SoKNOS project in section 2 [4]. This project mainly aims at integrating the systems of different command centers of public

R. Meersman et al. (Eds.): OTM 2010, Part I, LNCS 6426, pp. 186–203, 2010.

safety organizations in a disaster response. Real world studies have shown that a real-time overview of the activities and dependencies between organizations in a disaster is beneficial [5,6]. We have found out that the traditional view of sequential processes is limited with respect to unstructured processes [7]. This led us to the development of a new model for coordinating and executing disaster response activities in section 3. Its emphasis is different from traditional BPM approaches, which control and structure whole processes. It is about awareness and feedback of activities as well as the management of temporal dependencies. The model can be collaboratively defined and executed ad-hoc by integrating shared activities of different stakeholders cross hierarchies and organizations. We describe the formal foundation of the model using Allen's interval algebra and how it can be verified and executed. In section 4 we explain how our model is implemented in the collaboration infrastructure Google Wave. We discuss comments of domain experts about our approach section 5. Finally, we describe future research directions in the last section.

2 Domain Study

We describe the investigation that we have conducted in the field BPM for disaster management in subsection 2.1. Disasters should be differentiated from emergencies by the scale of the considered event, the number of people and organizations involved, the evolving nature of the events and also the inability of existing plans to cope with them [8]. Recent news gave us examples of such events like the earthquakes in Haiti or Chile. In subsection 2.2 we describe the state of art regarding BPM and crisis management. Finally, we provide a scenario in subsection 2.3, which is used as an example throughout the paper.

2.1 BPM for Disaster Management

To conduct this study, we investigated the requirements of the disaster management domain in general, within the SoKNOS project [4] and participated in workshops and interviews with disaster management stakeholders in Germany. We evaluated with them also the use of business process modeling languages for disaster response management using event-driven process chains (EPC) [9]. It is mandatory to model a process in order to support its coordination by a system. Together with a senior police commander, we modeled the response process to a train accident with hazardous material threatening a residential area nearby. Fig. 1 shows the process model immediately after the interview. We show only a few activities in detail due to space restrictions. In the first half a special organizational structure is created for responding to the disaster. The second half describes the actual response at a very high level. The model still did not provide a useful excerpt of the reality. Basically it shows that many activities are running in parallel. The process model does not deal with dynamically evolving situations requiring creation of new activities by anyone involved in the process (even new participants). For example, a fire fighter gets injured and his colleagues

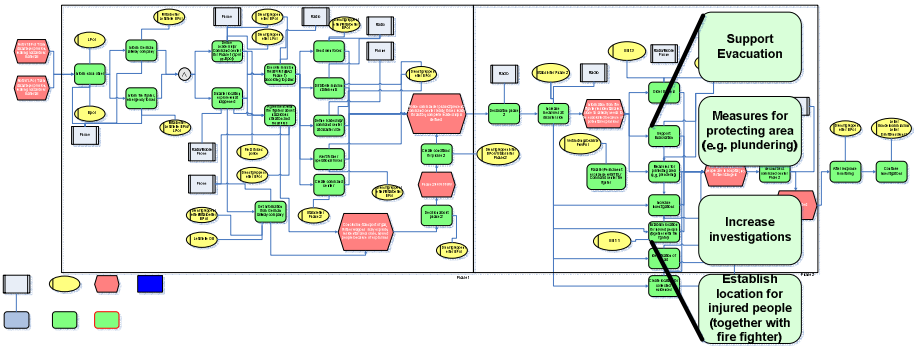

Fig. 1. Results of Modeling Disaster Response Processes using a Business Process Modeling Language

rescue him before continuing fighting the fire. It does not deal with the integration of activities of other hierarchy layers, e.g. teams in the field giving feedback on activities or creating new activities of relevance for the coordination by the command center. It also does not deal properly with the ad-hoc integration of other organizations in the process (illustrated in the first half). Only messages are sent to them with no further explanation (e.g. what they do with it) and feedback. The use of gateways (e.g. XOR-gateway) or other advanced modeling constructs make the model more complex, because many described alternatives are irrelevant in a specific situation. It unnecessarily limits the choices that can be made to the alternatives prescribed in the model and prevents creativity. This is an unrealistic assumptions for dynamic situations involving unstructured processes [6]. In particular, it does not seem to be very useful for supporting the temporal coordination of the activities during the disaster response. Temporal coordination is important to synchronize the execution of different activities of different organizations. Given these results, it is not surprising that the disaster response plans from fire fighters and police from the SoKNOS project do not contain business process models or textual description of activity sequences. They cover mostly available resources, a textual description of generic activities and interfaces with other organizations. However, usually one or more of these things changes during a disaster [5,6] and this requires that the organizations collaboratively coordinate their activities. This way of planning is more adequate for dynamic scenarios where the evolution of the situation cannot be prescribed. This can be seen in current practices of disaster management, but has also be proven by studies (e.g. several studies conducted by the Disaster Research Center of the University of Delaware [8]).

Other modeling attempts with three other public safety organizations showed the same results, even when using other modeling notations, such as the Business Process Modeling Notation (BPMN). This is not surprising, because research has shown that they are used similarly [10].

2.2 Related Research

Georgakopoulos et al. [11] and Fahland et al. [12] propose a scenario based approach for describing and executing processes. Depending on the situation different scenarios can be composed dynamically to represent the process. These are more suitable for managing predictable scenarios, which is not the case of a disaster. De Leoni et al. [13] adapt emergency processes automatically if the context, described in situation calculus, changes. They rely on modeling activity sequences and the context in detail, which makes it more suitable for predictable scenarios. The approach in [14] is similar. Based on our interactions with disaster response stake holders, we consider both approaches as too time and resource intensive for a disaster response. Reijers et al. analyze different resource scheduling mechanisms for workflow systems in emergency management [2]. They assume routine emergencies, standard business process models and no unexpected situations (e.g. failure of activities or unexpected extension of the crisis). Flexible business process management systems have been applied to the emergency management domain (e.g. [15]). They describe predictable routine emergencies. As mentioned before, business process models are not suitable to model a disaster response, which is about collaboration and coordination of activities with temporal dependencies. Our research confirms related research [12,16,13,11].

Ad-hoc or flexible workflow systems (e.g. [17,18,19,20,21,22]) rely on traditional business process models and thus have the same limitations with respect to coordinating disaster response activities mentioned before. Many of these limitations hold also for declarative process management approaches (e.g. [23,24]), but correct modeling requires also a lot of experience and time. They are also still very close to the workflow paradigm by describing a schema and instances. We are more interested in an ad-hoc collaborative approach. We also investigated disaster management software (e.g. the SAP Defense Force & Public Security system [25] or Sahana [26]), but they do not support temporal coordination of disaster response processes explicitly (i.e. by modeling, executing and monitoring them).

All these approaches consider predictable routine scenarios and structured business processes consisting of activity sequences using one centralized system. We argued before and confirmed in our interviews that modeling business processes is not suitable for disaster response processes [7]. This also means existing (flexible) systems using business process models are not suitable for coordinating unstructured processes in the disaster response. There is a need for a model for coordinating activities more closely to reality and not focusing on isolated processes. It should be able to (re-)combine activities in an ad-hoc manner crossing hierarchies and organizations [27], management of temporal dependencies and provide activity awareness for all stakeholders.

2.3 Scenario

The scenario in Fig. 2 is part of a larger case study derived from real flood disasters, established together with end users, such as fire fighter and police, in

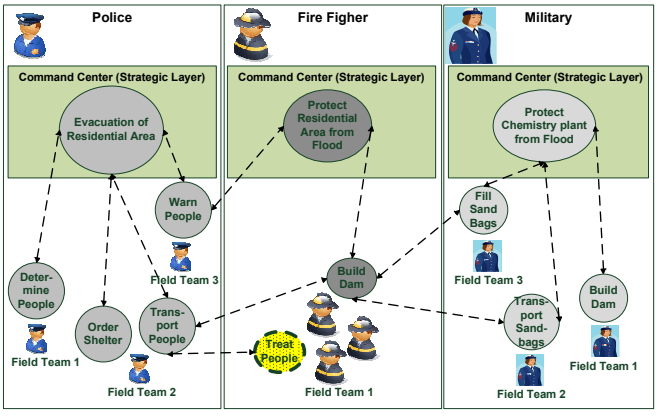

Fig. 2. Scenario

the SoKNOS project [4]. It will be used throughout the paper as an example. A residential area and a chemistry plant are threatened by a flood. Three organizations are responding to the disaster: police, fire fighter and military. Each organization has a command center and one or more teams in the field performing activities. In more complex disasters (cross regions/state/countries) more command centers are established (e.g. command center of the state) supporting the coordination of command centers of other organizations. These command centers do not build a hierarchical structure, but a network of organizations, which need to unify and synchronize their efforts [28]. Command centers communicate with each other by exchanging messages about, for example, the situation, what activities they are aware of or what the status of these activities is. In this use case, the police is responsible for evacuating the residential area. The fire fighters are building a dam to protect the residential area from the flood. The military is filling sandbags and transporting them to the disaster side. These sandbags are used by the fire fighters and the military to build a dam. Different kinds of activities are managed differently. For example, managing logistics activities, such as transporting sandbags, is different from evacuation activities, such as warning people. The relationship between activities can be described using temporal relationships (e.g. overlapping of activities). Some typical problems occurring when there is lack of coordination are, for example, no or too many sandbags arrive at the disaster site, transports fail, double efforts or no efforts etc. New activities can occur to cope with the situation. For example, it might happen that the fire fighters have to treat injured people of an evacuated area. Thus, there is a need for a better support for coordination to distribute and control the activities as well as their temporal relationships. It needs to provide the different stakeholder with enhanced situational awareness and to overcome the typical limitations of message-based exchange of the current situation/status. The proposed solutions from the BPM field do not seem to cope with these requirements. This is why we are proposing a new model that aims to answer this need.

3 A Coordination Model for Activities

We describe in this section the model for ad-hoc coordination of activities with temporal dependencies. The core idea of this model is that it is not imposed by the command center, but that all parts of a hierarchy (e.g. people in the field) and different organizations have influence on it. It has its foundation in Weick's sense-making theory [5] as well as Klein's work on naturalistic decision-making [6]. Both theories are grounded in real disasters and interviews with domain experts in disaster management and other domains, such as military, health care or complex engineering processes. In the sense-making theory people make sense out of what is going on and when they detect inconsistencies or ambiguity they start to interact to create a mutual understanding of the situation. We focus in our model on activities and temporal dependencies between activities as well as their violation during execution. Violation of dependencies needs to be managed by the users and/or the system. The idea for focusing on activities is also motivated by other disaster researcher, who define disasters as social constructions [29]. There, a disaster is defined by the activities humans execute to respond to a disaster. In the first subsection, we provide a description of the model. Then, we describe how this description can be verified to avoid inconsistent models. In the third subsection, we explain how activities are executed and how violation of temporal dependencies can be detected and managed during the execution. Although we describe this in a sequential manner, it is possible to do all this at same time, i.e. there is not a distinguishable modeling, verification and execution phase. There is a continuum of activity creation and execution that may occur all along the disaster. Finally, we show on an example based on our scenario how a system built on this model could actually be used.

3.1 Modeling

We distinguish between three model elements: activity type, activity and temporal dependency. The activity type describes the states of a management lifecycle of an activity and governance roles. For example, a logistic activity has different states in the management lifecycle in comparison to a more simple activity in the field (e.g. fighting fire). The activity, based on an activity type, describes disaster response activities. A temporal dependency can be established between states of different activities. It is qualitative (e.g. when activity "'Protect Area" is executed then the activity "Treat injured People" in this area can be executed). This is easier to define and agree on than providing concrete times (e.g. 5 hours and 5 minutes). Deadlines can still be integrated in our model, but this does not change the general concept. We do not use gateways, because they are difficult to model correctly ad-hoc and introduce unnecessary complexity. Different execution paths can be defined using the activity type instead. Our dependency model is able to reflect the temporal relations in a disaster response more adequately than typical sequential dependencies found in business process models, because it provides a greater variety of dependencies.

Definition 1. An *activity type* $at_d = (S, st, se, f, G)$ represents the management lifecycle of an activity with S is a finite set of activity states, $st \in S$ describes the start state of an activity type, $se \in S$ describes the end state of an activity type (i.e. a state where no further transition is possible), $st \neq se$ a start state is not an end state and $f : S \rightarrow S$ is a transition function defining the possible transitions from one state to another for one activity type. The lifecycle must not contain strongly connected components, because they can lead to confusion (e.g. an activity is re-executed although it has been finished before).

The specification of the activity type can be extended by governance rules G. They describe who can transit from one state to another. For example, an activity can be created by the command center that will be accountable for its execution. The responsibility to execute the activity will be given to someone on the field. Decision for cancellation or failure will be based on these roles. Modeling the activity type allows to intuitively specify deviations and to plan them in advance (e.g. activity "Build Dam" failed and this starts the execution of activity "Evacuate Area") as we will see later. Different kinds of activity types can be used in a given setting. They differ mostly by their life cycle and their governance rules. Some can be very simple (start, execute, terminate), some can be more complex and require more detailed planning and approval phases. Fig. 3 illustrates an example for such an activity type. The white circle describes the start state and the black circle describes an end state. Other states are "Plan", "Execute", "Idle", "Fail", "Cancel" and "Finish".

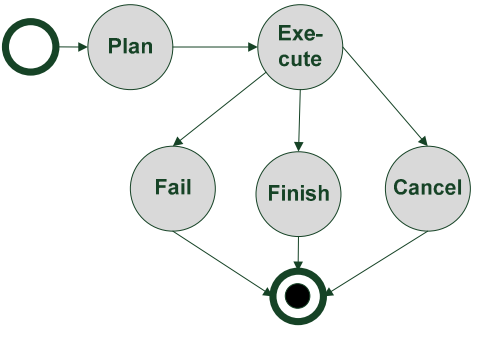

Fig. 3. Example for an activity type without governance rules

Definition 2. An *activity* is defined as $a_i = (uid, name, cs, cat)$ where uid is a unique identifier of the activity, $name$ describes the activity, $cs \in SA$ is the current state of the activity. On creation it must be the start state st of an activity type. $cat \in AT = (at_1, .., at_n)$ one activity type in the set of existing activity types.

An activity is by default independent from other activities. That means they can change their state in parallel without affecting other activities. However, a

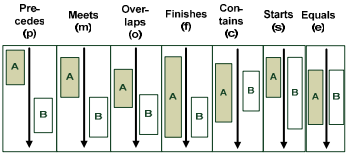

Fig. 4. Temporal dependencies between states of activities

dependency can be established between activities, if it is perceived by the user as important and if the user is aware of this dependency.

Definition 3. A *temporal dependency* is defined as $d_i = (a_s, s_s, a_d, s_d, type)$ with a_s is the source activity, s_s is the state of the source activity, a_d is the destination activity, s_d is the state of the destination activity and *type* is the type of temporal dependency. A temporal dependency can be established between two states of two different activities. Temporal dependencies are the core mechanism used to represent explicitly coordination between activities in our model. It is not necessary to connect every activity directly or indirectly via dependencies, i.e. the users of the system can focus on the most important ones as perceived by him/her. We use Allen's proposed time interval relationships for describing different types of temporal dependencies [30]. This provides a great level of flexibility regarding the kind of relationship that can be established between two states of an activity. A *type* describes the dependency (i.e. describes one or more of Allen's time interval relationships). This will be explained later in the subsection Execution.

Fig. 4 illustrates seven of them, because the other six are just the inverse of the first six. The dependencies have some interesting characteristics: they are qualitative, exhaustive and distinct. Temporal dependencies in highly dynamic scenarios are easier to define when they are qualitative, i.e. we do not need to specify exact times (e.g. 5 hours and 5 minutes). It is not always possible to define exact times, because this information is simply not available or subject to continuous change. As mentioned before, our model does not prevent to specify exact times. In our case, this means a temporal dependency of the state of one activity is relative to the state of another activity. This is different from BPM, where an activity depends only on the output of one or more activities. For instance, a rescue activity can only be executed during the execution of an activity for protecting an area. When we apply the notion of Allen's time interval relationships to the use case then we can describe the temporal dependency between the state "Execute" of the activity "Warn people" and the state "Execute" of the activity "Protect area" as "contains".

Further type of dependencies exist (e.g. resource or data dependencies), but we focus here on temporal dependencies, because they can be applied to all kind of activities and are important. The model can be extended by further dependencies, if there is a need.

3.2 Verification

It must be ensured that all inconsistent specifications of the model, which can be detected before adding a dependency to the model, are forbidden. An example for an inconsistent model, would be a simple model with three activities "A", "B" and "C". A dependency "precedes" is established between "A" and "B". Another dependency "precedes" is established between "B" and "C". Finally a dependency "precedes" is established between "C" and "A". This basically means that activity "A" (or any other activity) precedes itself.

Allen proposed in his work the path consistency algorithm for reasoning about a network of interval relationships [30]. Reasoning means that the algorithm is able to derive all other temporal constraints from a given temporal constraint network. This algorithm can be also used to detect an inconsistent network of temporal constraints. A constraint network can be constructed from our model by representing the transition of the states within one activity using the meets (m) constraint. Temporal dependencies between activity states of different activities can be represented using the corresponding constraints (i.e. all basic time interval relationships). A network is inconsistent if it is not possible to find a temporal constraint between nodes (or states) which fits with the other dependencies in the model. The path consistency algorithm is not complete when considering all possible compositions of time interval relationships between two nodes [30]. However, it is complete for some subsets of them. We restrict our model to the Ord-Horn-Class, which is such a subset of composition of time interval relationships (more details and detailed analysis of this class can be found in [31]). It contains all the basic relationships mentioned before (i.e. it allows verification of our model). The composed constraints it does not support are not seen as relevant for the practice [30] and are not needed for our model since we only rely on the basic dependencies mentioned above. A complete version of the path consistency algorithm for verifying a constraint network is a NP-hard problem [30]. Our model only requires the compositions of time interval relationships defined in the Ord-Horn-Class. This means we can use the path consistency algorithm by Allen and it is complete for this subset [31]. This path consistency algorithm has a computational complexity of $O(N^3)$ assuming a given constraint network, whereby N is the number of connected nodes (states of activities). Adding a new connected node (e.g. by establishing a dependency in our model) and verifying the model leads to a computational complexity of $O(N)$ [30]. This is acceptable for our case. Another interesting aspect of the formalism is that it allows to change (i.e. add activities and dependencies) and verify the model during execution, because it does not depend on a specific execution state. This is very difficult to achieve with other well-known formalisms, such as Petri nets [32].

3.3 Execution

Execution of activities is about changing the state of an activity. This may violates temporal dependencies connected to the activity. We describe how the violation of temporal dependencies can be managed and how it can be detected.

Managing Violation of Dependencies. A dependency can be violated, if one or more activities changes their state contradictory to the dependency (e.g. activity "Build Dam" changes into a state "Execute", although activity "Transport Sandbags" was supposed to change into the state "Execute" before). There are two options to manage this: (1) State changes that violate dependencies can be prevented, (2) violated dependencies can be displayed to the users. Although usually the latter treatment is preferred, when managing disaster response activities, we plan to support also the other case, so that our approach can be applied to more different domains and also work with automated activities. Option (1) can lead to a situation, where the system cannot continue execution, although not all activities are in end states. We ensured in the verification section that this cannot happen by defining all dependencies as conflict-free. Option (2) requires detection and displaying of violated dependencies. If a violated dependency is visualized then user has to deal with it outside the system. After resolving the issue, the user can deactivate the violated dependency.

Sometimes it is the case that dependency violation should be avoided. This means that one or more other activities should change into a desired state. The user might be able to do this or ask other users (or machines in case of automated activities) to do this, because he/she has not the right to do so. For example, if the activity "Build Dam" changes into state "Execute" before the activity "Transport Sandbags" changes into the state "Execute" then the system can trigger itself (if possible) or ask the corresponding role of activity "Transport Sandbags" to do a state change into state "Execute". If this is not possible then the corresponding users are notified that the dependency is violated or if the dependency needs to be enforced then it is not possible to perform the initial state change. This can be supported by a protocol in our system.

Detecting Violation of Dependencies. We describe the detection of violation of dependencies with algorithm 1. The algorithm checks all dependencies associated with all activities performing a state change (described in L). It allows, for example, detecting a violation if two or more activities have to be in the same state at the same time (this corresponds to the dependency

```
input  : List L of state changes of one or more activities
output: A set V of violated dependencies

dependencylist ← GetAllDependencies(L)
for i ← 0 to dependencylist.size - 1 do
    CheckDependency(dependencylist[i],L);
    if GetState(dependencylist[i] == violated) then
        V ← dependencylist[i]
    end
end
```

Algorithm 1. Detect violation of dependencies when executing activities

"equals"). Execution has a complexity of $O(M)$ whereby M is the number of dependencies associated with the activities of which the state needs to be changed. This is also acceptable for our use case. Our approach does not rely on specifying quantitative time (e.g. [22]), inflexible enforcement of the constraints (e.g. [33]) or defining fixed workflows (e.g. [21]) to execute the model. The first one contradicts Allen's idea of qualitative constraints and it is very difficult in a disaster with a dynamic evolving situation to define exact times (e.g. activity "Build Dam" will be executed 5 hours and 5 minutes). They would be also subject to continuous change and disagreement between different organizations. The latter ones require defining sequential business process models, which do not work well for disaster response processes as we have shown before.

Detecting violation of dependencies (CheckDependency) works as follows: Different types of dependencies are represented as different finite state machines. State changes are input for the finite state machine. Depending on the input they change into the state "Violate" or "Neutral".

Definition 4. The type of a dependency is defined as $type = (\Upsilon, \Omega, s, t)$ with Υ is the input alphabet (all accepted state changes of the activities involved in the dependency), Ω is the finite set of states of the dependency, s is the current state and $t : \Omega \times \Upsilon \rightarrow \Omega$ is the transition function.

Definition 5. The transition function t supports the following constructs and their combination: $A : Sa$ activity A changes into the state Sa, $\neg(A : Sa)$ activity A changes into any other successor state of Sa, $A : Sa \wedge B : Sb$ activity A changes into the state Sa and activity B changes into the state Sb or $else$ any other state change of activity A or B.

Example. Fig. 5 provides an example for a finite state machine representing the dependency "overlapped by". The dependency is established between an activity "A" in the state "Sa" and an activity "B" in the state "Sb". Initially, the finite state machine is in the state "Neutral". This means the dependency is not violated. If activity "B" changes in the state "Sb" then finite state machine of the dependency changes into the state "Violated". If activity "B" changes now to any other state than "Sb" then the finite state machine of the dependency transits to the state "Neutral".

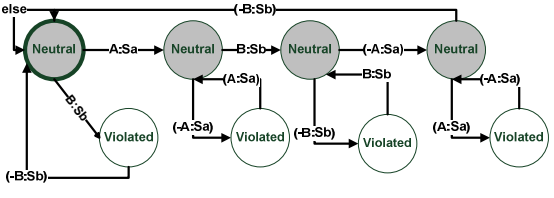

Fig. 5. Finite state machine describing the dependency "overlapped by"

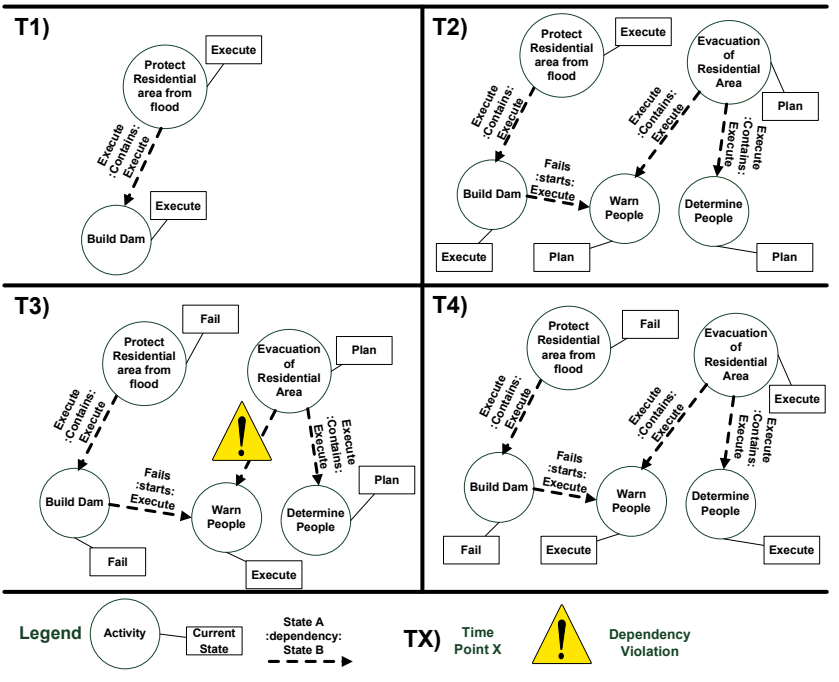

Fig. 6. Example for modeling and executing the scenario

3.4 Example in Context of the Use Case

We demonstrate an example of the evolution of our model in Fig. 6 based on the use case described before (cf. Fig. 2). The evolution of the situation is illustrated in four phases. In phase one, the fire fighters have created the activities "Protect Residential area flood" and "Build Dam". The first activity is on the strategic command center level and the second activity is an activity in the field. Both have a dependency "contains" between them in state "Execute". Both activities are currently in the state "Execute". In the second phase, the activities "Evacuation of Residential Area", "Warn People" and "Determine People" added to the model in the state "Plan" by the police. A dependency "starts" is established between the state "Fail" of activity "Build Dam" and the state "Execute" of the activity "Warn people" of the police. In the third phase, the activity "Build Dam" enters the state "Fail" and "Warn people" enters the state "Execute". This leads to violation of the dependency "contains" between the activity "Warn people" and "Evacuation of Residential Area", because the latter is still in the state "Plan". In the fourth phase, this is corrected by changing the state of activity "Evacuation of Residential Area" to the state "Execute". The modeling is usually not done using one system, but there can be many different systems

and models exchanging activities and establishing dependencies in a decentralized manner as illustrated in [7]. This means that different organizations are allowed to define different dependencies to activities and not all dependencies and activities are shared. This is scope of another paper. We do not envision one large model where all organizations model their activities and dependencies. Several models can exist (e.g. one by each organization) and one shared activity can be part of several models [7]. This fits to the requirements of the disaster management domain. There we have several independent organizations and each organization has their own tools, but they need to coordinate with each other in a de-centralized setting.

4 Solution Design in the Open Wave Federation Context

We have implemented our model as an extension to the collaborative infrastructure Google Wave [34]. It allows for a decentralized infrastructure (e.g. every organization can have its own server) and it enables real-time interaction between the participants of different organizations (i.e. servers) based on the OpenWave Federation Protocol [35]. This provides us also a solution for sharing of activities. A web-based interface, which can be accessed by any standard compliant browser, is another important requirement of the end users. Google Wave has been used in previous disasters (e.g. in the Haiti earthquake [36]) and other software is able to inter-operate with it (e.g. SAP Streamwork). In the following subsections, we introduce briefly Google Wave and how it can be extended. Afterwards, we describe how we implemented our model as part of this.

4.1 Preliminaries

The Open Wave Federation Protocol supports communication between different Wave servers. Participants register to one of these servers. Wave servers host documents, which can be collaboratively edited by different participants. A collaborative document is called a "Wave". At any point in time participants can be added to a "Wave". It is replicated to all the servers of the participants. The server, which created the wave, holds the reference copy of the "Wave" and manages the distribution of updates to it using Operational Transformations [35]. A "Wave" can contain one or more "Wavelets". They have similar characteristics like a "Wave", but they can have their own set of participants. The reference copy of the "Wavelet" can be managed by a different server than the one managing the reference copy of the "Wave". Google Wave and the Open Wave Federation Protocol support two extensions: "Gadget" and "Robot". A "Gadget" can be inserted into a "Wavelet" and it is basically a web-based graphical user interface to support new collaborative functionality (e.g. modeling of processes). A "Robot" is added to a "Wave" or "Wavelet" the same way as a participant and they link the outer world (e.g. a stock market feed, social networks or other "Waves") with a "Wave" and/or "Wavelet". It represents automated behavior.

4.2 Implementation

We represent one activity as a "Wave", the so-called "Activity-Wave". Participants can be added to it. For example, if a fire fighter in the field is rescued then an "Activity-Wave" can be created for this activity. People in the command center can be invited to it and they can integrate it in their models (see below). An "Activity-Wave" contains at least two "Gadgets": The "Gadget-Activity-Specification" and the "Gadget-Activity-ParticipantView". The "Gadget-Activity-Specification" allows to specify the activity name, the activity type and governance roles (This is part of the governance concept we did not describe in detail here). Each "Model-Wave", the activity is modeled in, has an own "Wavelet" in the "Activity-Wave" where participants of this "Model-Wave" define the current state of the activity in the model using the "Gadget-Activity-ParticipantView". This allows detection of conflicting perceived states and allows participants continuing to change the state of a shared activity in case of disconnection. This requires special synchronization mechanisms not detailed here.

A "Model-Wave" is a "Wave" containing the "Gadget-CurrentModelView" which allows modeling activities and dependencies. Activities modeled in this "Gadget" are linked to an "Activity-Wave". By clicking on these modeled activities we can switch to the linked "Activity-Wave".

The "Robot-UProMan" is participant in the "Activity-Waves" and the "Model-Waves". It verifies the model using the path consistency algorithm described before. It displays a warning in the "Gadget-CurrentModelView" if the model is inconsistent. It manages violation of dependencies during execution of activities as described above. Logging of the execution of activities is also done by this robot.

Fig. 7 illustrates a "Model-Wave" (left) as well as an "Activity-Wave" (right) of one activity ("Transport Sandbags") in the model. The "Model-Wave" is managed by the commander of the fire fighter. It contains the activities "Protect Area from Flood", "Fill Sandbags", "Build Dam" and "Transport Sandbags". Initially, all activities are in the state "Plan". There are three dependencies "contains" from the state "Execute" of the activity "Protect Area from Flood" to the three other activities in state "Execute". The activity "Transport Sandbags" has been changed into state "Execute", which violates the dependency "contains" to the activity "Protect Area From Flood", because it is still in state "Plan". It turns out that the person responsible for transport already initiated it, although the situation is still assessed. The violation is visualized by a red dependency and a warning sign. For example, the commander can now change the activity "Protect Area from Flood" into the state "Execute" or start a discussion in the "Wave" of the activity "Transport Sandbags". Participants can enrich models or activities by adding text or other gadgets to the "Wave". This is illustrated in the "Model-Wave" where a user inserted a map of the area.

The "Robot-UProMan" also supports further functionality such as execution of activities part of several models (e.g. in an inter-organisational setting) as well as re-synchronization of activities and models after disconnection or conflicting states of an activity, which we did not detail in this paper.

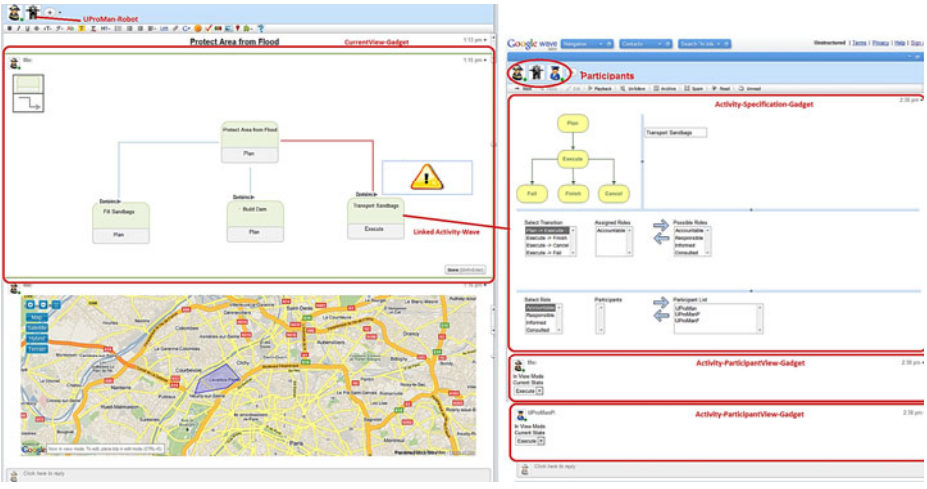

Fig. 7. Screenshot of our Prototype

5 Discussion

We let domain-experts in disaster management comment our approach, by presenting it to them, to find out how our approach is seen in comparison with other systems or concepts currently used for activity management and what the limitations of such an approach are. End users currently use means such as whiteboards, web-based mission diaries or email. The problem is that they can become even in smaller disasters quickly counterproductive (e.g. many mails with different context and not related with each other). We evaluated the model together with commanders of three fire fighter departments in France and in the US as well as with an international humanitarian aid organization in Germany. The interview with the fire fighters in France was a face to face interview of about three hours. The other interviews were one hour phone interviews, but we shared our screen via Internet to illustrate the approach.

The model of activities and temporal dependencies seems to be a concept familiar to the end users: "we use time lines as markers for future action, we have what we call trigger points, when the incident advances to a certain point, it triggers other things, so that would fit into your model as well, using time lines, connecting inter-dependencies" (Fire Fighter Commander Southern California).

It was highlighted that this model can overcome the limitations of web-based mission diaries used at the moment: "we have an incident action plan which each entity utilizes and keeps track of their system and activities, more or less on a manual basis and entering who is command and what actions are taken [..] it outlines future actions and intended actions for the next twelve hours operational period [..] (but it) does not alert you to inter-connection failures [..]" (Fire Fighter Commander Southern California). In a mission diary entries are

sequential and not related (e.g. by temporal dependencies). This leads to cases where someone might read outdated information and is not aware of this.

The interviews confirmed that the model and the implementation could be useful in the following situations: complex situation, large geographical area, many organizations involved and long enduring response. Examples given by the end users were low-pacing floods or snow storms. They also described the limitations of information system support for the disaster response. There are situations, such as wildfires, where it is very difficult for teams in the field to use any kind of information technology. In these situations they hardly use any communication devices and communication occurs infrequently. Nevertheless the approach can still be useful for command centers in these situations. These are positive indicators, but a detailed study with a stable version of the prototype including some more usability features is part of future research.

6 Conclusion and Further Research

Unstructured processes introduce new technical challenges for supporting their coordination by a system. However, such support is seen as beneficial to be able to manage them. We analyzed the requirements for process management in the domain of disaster response management to challenge the concept of unstructured processes. In the beginning we invalidated together with end users the use of business process models for disaster response processes. We developed a generic model for coordination of activities with temporal dependencies. We demonstrated how the model can be verified using Allen's interval algebra and managed by a system. The model can be integrated with further views, e.g. resource view or geographical view. The model puts different emphasis on the benefits of BPM than traditional BPM approaches. Structuring of the process is more loosely fitting to the requirements of unstructured processes. Managing dependency violation, activity awareness and feedback are more important than controlling and enforcing a process within the system. Not all possible dependencies need to be modeled, but only the ones perceived as important by the users. The main difference is that it allows collaborative coordination of activities with temporal dependencies. All these features are reflected in the implementation leveraging the Google Wave collaboration infrastructure based on open standards. Although the results of our interviews look promising, we are going to conduct further evaluations. This will help us to identify further human and organizational limitations (e.g. scalability) of the model implemented in the prototype. The literature, describing the underlying human aspects of our model, suggests that results from disaster management can be applied to other domains with similar characteristics [6,5], e.g. military, project management or ad-hoc supply chains. From a technical perspective we want to investigate how our approach can work on the distributed level and how automated activities or coordinators (i.e. systems that create and manage models) can be integrated.

Acknowledgements. The research was partially funded by the German Federal Ministry of Education and Research under the promotional reference 01ISO7009

and by the French Ministry of Research within the RESCUE-IT project [37]. The authors take the responsibility for the content. We thank the domain experts for providing us detailed insights.

References

1. Olding, E., Rozwell, C.: Expand your bpm horizons by exploring unstructured processes. Technical Report G00172387, Gartner (2009)
2. Reijers, H.A., Jansen-Vullers, M.H., zur Muehlen, M., Appl, W.: Workflow management systems + swarm intelligence = dynamic task assignment for emergency management applications. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 125–140. Springer, Heidelberg (2007)
3. Bunker, D., Smith, S.: Disaster management and community warning systems: Inter-organsational collaboration and ict innovation. In: Pacific Asia Conference on Information Systems, PACIS (2009)
4. SoKNOS: Soknos (soa zur unterstützung von netzwerken im rahmen oeffentlicher sicherheit) (2009), `http://www.soknos.de` (retrieved 03.06.2009)
5. Weick, K.: Making Sense of the Organization. Blackwell, Malden (2000)
6. Klein, G.: Sources of Power: How People Make Decisions. MIT Press, Cambridge (1999)
7. Franke, J., Charoy, F.: Design of a collaborative disaster response process management system. In: 9th International Conference on the Design of Cooperative Systems (2010)
8. Quarantelli, E.: Emergent behavior at the emergency - time periods of disasters. Technical report, Disaster Research Center, University of Delaware (1983)
9. Scheer, A.W.: ARIS - Business Process Modeling, 3rd edn. Springer, Heidelberg (2000)
10. Recker, J.: Understanding Process Modelling Grammer Continuance - A Study of the Consequences of Representational Capabilities. PhD thesis, School of Information Systems, Queensland University of Technology, Brisbane, Australia (2008)
11. Georgakopoulos, D., Schuster, H., Baker, D., Cichocki, A.: Managing escalation of collaboration processes in crisis mititgation situations. In: 16th International Conference on Data Engineering (2000)
12. Fahland, D., Woith, H.: Towards process models for disaster response. In: Process Management for Highly Dynamic and Pervasive Scenarios (2008)
13. de Leoni, M., Mecella, M., De Giacomo, G.: Highly dynamic adaptation in process management systems through execution monitoring. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 182–197. Springer, Heidelberg (2007)
14. Zahoor, E., Perrin, O., Godart, C.: An integrated declarative approach to web services composition and monitoring. In: Vossen, G., Long, D.D.E., Yu, J.X. (eds.) WISE 2009. LNCS, vol. 5802, pp. 247–260. Springer, Heidelberg (2009)
15. Rüppel, U., Wagenknecht, A.: Improving emergency management by formal dynamic process-modelling. In: 24th Conference on Information Technology in Construction (2007)
16. Denning, P.J.: Infoglut. Communications of the ACM 49(7), 15–19 (2006)
17. Huth, C., Erdmann, I., Nastansky, L.: Groupprocess: using process knowledge from the participative design and practical operation of ad hoc processes for the design of structured workflows. In: 34th Annual Hawai International Conference on System Sciences (2001)

18. Grigori, D., Charoy, F., Godart, C.: Anticipation to enhance flexibility of workflow execution. In: Mayr, H.C., Lazanský, J., Quirchmayr, G., Vogel, P. (eds.) DEXA 2001. LNCS, vol. 2113, Springer, Heidelberg (2001)
19. van der Aalst, W., Weske, M., Grünbauer, D.: Case handling: A new paradigm for business process support. Data & Knowledge Engineering 53, 129–162 (2005)
20. Dadam, P., Reichert, M.: The adept project. Computer Science - Research and Development 23(2), 81–97 (2009)
21. Lu, R., Sadiq, S., Padmanabhan, V., Governatori, G.: Using a temporal constraint network for business process execution. In: 17th Australasian Database Conference (2006)
22. Kafeza, E., Karlapalem, K.: Gaining control over time in workflow management applications. In: Ibrahim, M., Küng, J., Revell, N. (eds.) DEXA 2000. LNCS, vol. 1873, Springer, Heidelberg (2000)
23. van der Aalst, W.M.P., Pesic, M.: Decserflow: Towards a truly declarative service flow language. In: Bravetti, M., Núñez, M., Zavattaro, G. (eds.) WS-FM 2006. LNCS, vol. 4184, pp. 1–23. Springer, Heidelberg (2006)
24. Leymann, F., Unger, T., Wagner, S.: On designing a people-oriented constraint-based workflow language. In: 2nd Central-European Workshop on Services and their Composition (2010)
25. SAP: Defense force & public security (dfps) system, `http://help.sap.com/content/documentation/industry/docu_is_ds.htm` (retrieved 03.06.2009)
26. Sahana: Free and open source disaster crisis management system (2009), `http://www.sahana.lk/` (retrieved retrieved 03.06.2009)
27. Franke, J., Charoy, F., Ulmer, C.: A model for temporal coordination of disaster response activities. In: 7th International Conference on Information Systems for Crisis Response and Management (2010)
28. Tierney, K., Trainor, J.: Networks and resilience in the world trade center disaster 2003-2004. In: MCEER: Research progress and accomplisments, Multidisciplinary Center for Earthquake Engineering Research (MCEER), pp. 157–172 (2004)
29. Perry, R.W., Quarantelli, E. (eds.): What is a Disaster? New Answers to Old Questions. Xlibris Corp. (2005)
30. Allen, J.F.: Maintaining knowledge about temporal intervals. Communications of the ACM 26(11), 832–843 (1983)
31. Nebel, B., Bürckert, H.J.: Reasoning about temporal relations: A maximal tractable subclass of allen's interval algebra. Journal of the ACM 42(1), 43–66 (1995)
32. Ellis, C., Keddara, K., Rozenberg, G.: Dynamic change within workflow systems. In: Conference on Organizational Computing Systems (1995)
33. Raposo, A.B., Magalhaes, L.P., Ricarte, I.L.: Petri nets based coordination mechanisms for multi-workflow environments. International Journal of Computer Systems Science & Engineering 15(5), 315–326 (2000)
34. Google: Google wave, `http://www.googlewave.com` (retrieved 27.05.2010)
35. Lassen, S., Thorogood, S.: Google wave federation architecture, `http://www.waveprotocol.org/whitepapers/google-wave-architecture` (retrieved 27.05.2010)
36. gwtips: Wavers collaborating for haiti, `http://gwtips.com/wavers-collaborating-for-haiti/` (retrieved 04.05.2010)
37. Schaad, A., Ulmer, C., Gomez, L.: Rescueit - sécurisation d'une chaine logistique internationale. In: Workshop Interdisciplinaire sur la Sécurité Globale (2010)