Coordination of Emergency Response Operations via the Event-Based Awareness Mechanism

Bo Yu and **Guoray Cai**
College of Information Sciences and Technology,
The Pennsylvania State University,
University Park, PA 16802, USA
{byu, cai}@ist.psu.edu

ABSTRACT

Emergency response involves collaboration among search and rescue workers, medical staff, transportation coordinators, and others to save human lives and minimize damages. While carrying out local activities, members of the teams must also attend to new events happening elsewhere that may affect their work, and be prepared to adjust their activities accordingly. This paper describes a computer supported coordination system, DACE (Dependency-based Awareness and Coordination Environment), which offers a scalable solution to coordination in emergency response. The system serves as a cognitive aid to human actors in both maintaining a group mental model of the overall collaborative activities and their dependencies, and determining the effects of events as they propagate through the web of dependencies. We demonstrate the principles and utility of the DACE system through a hypothetical scenario of search and rescue exercise. This work contributes to the goal of scaling up awareness-based coordination in emergency response.

Keywords

Emergency response, dependency, coordination, awareness, event processing

INTRODUCTION

Emergency response operations are well recognized as collaborative activities with a high level of complexity (Turoff et al. 2004). In response to an emergency situation, many actors from different organizations form response teams operating in extreme volatile environment. They usually engage in a multitude of activities that are *distributed* in time and space, but strongly *interdependent* (Franke & Charoy 2010). They have to divide, align, and interrelate their individual activities with aspect to spatial, temporal, and resource constraints. They have to explicitly manage situations of conflicts among activities. Due to such interdependencies, even a locally occurring event could potentially have a broader, global impact, requiring coordination in order to steer the work on the right track (Schmidt 1991). Coordination work represents additional cost of attention and effort to response teams (Chen et al. 2008). Hence, supporting coordination of these distributed and interdependent activities has become one of the fundamental design requirements to develop emergency response management information systems (Turoff et al. 2004).

When dependencies are known and predictable, the most efficient mode of coordination is through a prescribed action plan, which directs the choices of actions at a given situation. However, the practices in emergency operations are characterized by a high level of contingency, leading to the fact that the exact actions and responsibilities of actors cannot be pre-determined (Turoff et al. 2004). Traditional models of coordination based on workflows and disaster plans are insufficient to support coordination in emergency response. Coordination in such context is less dependent on a prescribed process structure and is more contingent on knowledge integration and situation awareness (Faraj & Xiao 2006). Recently, Chen et al. (Chen et al. 2008) proposed a life-cycle approach for coordination that emphasizes dependencies arising from sudden and unexpected events, time pressure, and resource shortage. As a result, "coordination by awareness" is becoming a promising trend of coordination technology in emergency response activities (Gonzalez 2008; Faraj & Xiao 2006). The idea of this method is to provide actors an awareness of their current situation and each other's activities so that they can make mutual adjustment in coordination processes.

Emergency response efforts are triggered by an incident, or a disaster, i.e. an "event" happening in the environment. To respond to the initial event, actors perform activities that add more events building up the situation. In addition, unexpected new events can occur in the environment, requiring continuous response. It is

these critical events that cause the actors to identify changing dependency relations, and try to resolve possible problems or conflicts. In this paper, we focus on the role of awareness in coordination and adopt an event-based approach. The application of event-based model offers high potential for coordination support in emergency response operations (Pottebaum et al. 2011). We believe that an event-driven model of coordination is most appropriate to the domain of emergency response characterized by different types of dynamics.

One of the critical concerns about event-driven coordination is that there are potentially a very large number of events that may or may not be relevant to an actor at any given time, causing information overload. To address such cognitive overload, existing event-based systems commonly employ a publish/subscribe interaction paradigm (Eugster et al. 2003), where actors have the ability to express their interests in a set of events or patterns of events, in order to be notified subsequently when any event that matches their subscribed interests is published (Prinz 1999). The subscription mechanisms in these systems are based on the assumption that recipients of these events have the full knowledge and reasoning capacity to judge the relevance of possible events. However, this assumption is not always true in complex situations, such as emergency response operations, where the overall activities are so distributed and complex that actors rarely have the knowledge and expertise beyond their assigned roles and scopes of work. It is even more challenging to maintain the knowledge about how activities depend on each other. At a given state of a response operation, multiple dependency relationships can arise among the various activities and interweave with each other (Shen & Shaw 2004; Franke & Charoy 2010). One activity may have to be performed before another; they may require the use of the same resource; or one activity is to achieve a subsidiary goal that is necessary for another activity. Furthermore, multiple dependencies can co-exist in a single operation and form a web of relationships, leading to cascading propagation of impacts from one activity to another. As a result, it becomes impractical for users to pre-determine the relevance of all possible events and reason about impacts on their activities.

To scale up the event-based model to more complex situations, several existing research projects have integrated the event-driven architecture with complex event-processing (CEP) engines to provide more effective event management in response operations. Examples are PRONTO (Pottebaum et al. 2011) and PLAY (Truptil et al. 2012). An important merit of these complex event-driven systems is the capability to reason about events and their effects. While users may be only knowledgeable about event handling within their local scopes of work, the system can infer implications of events on the whole collaborative activities and derive new events that are relevant to the users, based on the system knowledge about the processes in the application domain. However, given the unpredictable nature of a crisis and its reliance on tacit knowledge being generated by the situation (Turoff et al. 2004), such domain knowledge about processes and derived inference rules tend to be too rigid to handle the high level of contigency in emergency response situations. In this paper, we provide a more flexible approach to event reasoning by utilizing the situated knowledge about activities and dependencies in an emergency response operation. Instead of stipulating how events should be processed following pre-determined business processes in the domain, our system maitains a computational model of the ongoing activities and their dependencies, and use this kind of situated knowledge to reason about effects of incoming events, and produce new derived events for notification.

This paper proposes our solution to supporting event-based coordination in complex emergency response activities. In particular, our approach helps the users to determine the relevance of events by automating the reasoning on how an event impacts the states of activities as its effects propagate across multiple activities. Our model does not rely on a pre-determined process model. Instead, it is based on a situational representation of activities and dependencies in an emergency response operation and uses this situated knowledge to perform event reasoning. In the following sections, we first present our conceptualization of the event-driven coordination process in the context of an emergency response scenario. Then our computational framework for supporting event-driven coordination is described. To validate our design methods, we describe the prototype system DACE (Dependency-based Awareness and Coordination Environment) that has been developed as a proof of concept. Its application to a hypothetical scenario demonstrates the advantages as well as limitations of our method.

A FRAMEWORK FOR EVENT-DRIVEN COORDINATION IN EMERGENCY RESPONSE

Drawn from the research literature on coordination theory (Malone & Crowston 1994), we define coordination as the management of dependencies among collaborative activities. Following this definition, dependency becomes the central concept on which the coordination work is founded. The goal of our framework is to provide the basis for designing cognitive aids to human actors in the event-driven coordination process of complex emergency response operations. The framework has two major components: (1) understanding and modeling collaborative activities and their dependencies, and (2) developing a representation and reasoning mechanism that helps the users to determine the relevance of events by automating the reasoning on how an event impact the states of activities as its effects propagate through the web of dependencies. In order to

elaborate this framework in the context of emergency response, we introduce a fictitious scenario of a large-scale event in response to chemical pollution.

> **Scenario.** *A chemical factory near an urban area was exploded and caused a major pollution. To respond to this critical incident, a response team is formed that includes search and rescue responders, decontamination manager, medical personnel, and transportation manager. The process is made up of several parallel activities, performed by several actors at distributed geographical locations. (1)* ***Searching*** *for victims. First responders patrol within the incident area to search for victims and report their locations and status. (2)* ***Decontamination****. All contaminated victims need to be decontaminated before they can be moved to other facilities. The decontamination manager assigns a station for each victim and the personnel at the station will perform the decontamination. (3)* ***Medical treatment****. The medical manager assigns a proper medical station for each victim and making sure they are treated in time. (4)* ***Transportation****. The transportation manager manages all the transportation activities by assigning a driver for each activity.* See Figure 1 for the spatial situation.

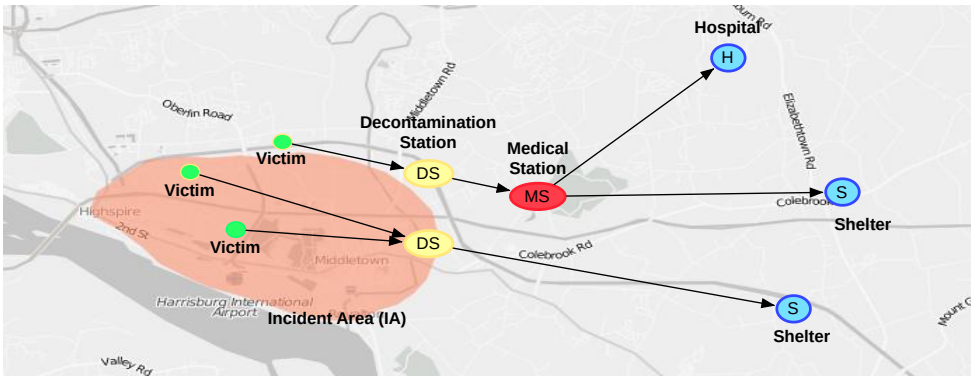


Figure 1. An emergency scenario

In the scenario, although each of these activities operates in parallel in semi-autonomous fashion, there are a large number of dependencies between them that could create situations where coordination is needed. For example, a temporal dependency exists between the decontamination operation and medical treatment, indicating that the victim needs to be decontaminated before he/she is transferred to the medical station to avoid spreading of toxic materials. Meanwhile, both the decontamination operation and medical treatment depend on the transportation activities to make sure the victims are delivered to assigned stations in time, hence they compete with each other for the limited transportation resources. To succeed in this joint effort, all these different types of dependency need to be managed effectively.

Dependencies in an emergency response are rarely static, but rather subjective to changes due to events in environment and human activities. Imagine the occurrence of an unexpected event indicating a traffic jam that blocks the route for delivering a victim to a decontamination station. This event is first reported to the transportation manager, who interprets the event as the cause of a delay for a vehicle that is used to deliver a victim to the decontamination station as scheduled. As a result, the dependency between the decontamination activity and transportation becomes problematic. Furthermore, the delay could further impact the medical manager due to the temporal dependency between decontamination and medical treatment. To handle the problematic dependency, the transportation manager may have to assign another vehicle to perform the task, or the decontamination manager may switch the victim to another station. These responses will then generate new events, which may further impact other dependencies.

In this study, we conceptualize the event-driven coordination process as a cyclic process that involves the interaction between events and dependencies. (1) Initially some interesting event is observed or detected (*event detection*). (2) Based on the gathered information from the event, the actor who detected it needs to understand how the event can impact the dependencies between them (*event interpretation*). (3) The interpreted change on one activity can (or potentially can) lead to changes to other activities due to changing dependencies among them, which needs to be furthered evaluated (*event elaboration*). (4) Based on the results of event interpretation and elaboration, the actor makes decisions on what need to be done (*response generation*). The responses generated from one event may lead to new events that trigger new coordination cycles. Figure 2 shows an instance of the event-driven coordination process in the scenario, where the event about the occurrence of a traffic jam is initially associated with a driver's transportation task, but propagated through multiple dependency relationships to the decontamination manager, who has to act upon the changing situation to assign the victim to a new decontamination station.

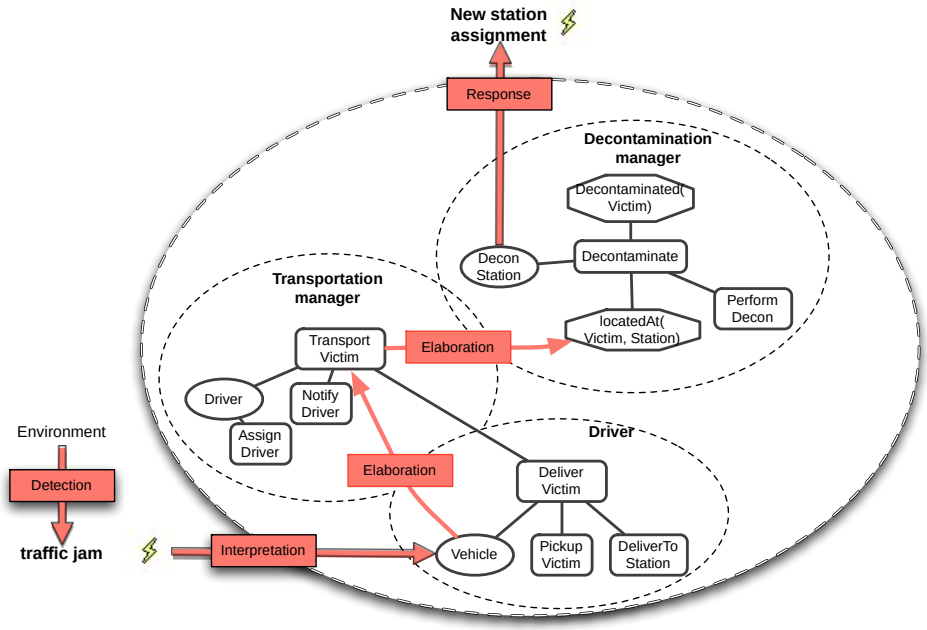


Figure 2. The event driven coordination process

Modeling activities and dependencies

In order to facilitate the coordination of activities in emergency response operations, the system needs to maintain a computational model of dependencies among these activities. Such a model should be based on a comprehensive understanding of activities and dependencies in emergency response. In this study, we extend the *activity-centered* approach to conceptualize activities and dependencies in emergency response. In the *activity-centered* approach proposed by Franke and Charoy (Franke & Charoy 2010), activities are considered as specialized micro processes with dependencies towards other activity micro processes, forming a network that should remain manageable. Instead of considering activities and their relationships as pre-defined business processes, the activity-centered approach models activities and their dependencies as they form and transform in the unfolding process of response operations. It allows actors to create new activities and dependencies during the process. However, their model only considers one particular type of dependencies between activities (i.e. temporal constraints), without connection to other relevant concepts such as resource management, and goal decomposition.

In general, our conceptualization of activities and dependencies in emergency response includes three levels:

Level 1: basic elements of activities. We define activities with their various relationships to other basic elements (i.e. *resources*, *conditions*, other *activities*) in emergence response. An *activity* specifies a particular way of doing something. An *activity* can be either a basic operation, or a complex action that needs to be decomposed into subsidiary ones. A *resource* can be anything that is used in emergency response activities, including both physical and informational objects, such as victims, rescue vehicles, and traffic information. A *goal* is a state of affairs in the world that the actors would like to achieve. How the goal is to be achieved is not specified, allowing alternatives to be considered.

Level 2: local scopes of work. Although various activities can be identified in one single emergency response operation, each actor usually only engages in a small set of them. In fact, one of the fundamental motivations for teamwork is to decouple a complex problem into a set of smaller ones that are much easier to manage and tackle. One result of the decoupling is that the work of actors is distributed and each actor is only interested in a small set of activities, resources, and conditions that are relevant to their current work focus, which we define as the *local scope of work*. For example, the local scope of the decontamination manager in the scenario includes the decontamination stations (*resources*), the goal that all victims need to be located in assigned stations (*goal*), and the operations at each station (*activities*).

Level 3: dependencies. Although activities in emergency response are largely distributed and belong to local scopes of different actors, they cannot be performed without interacting with each other. Because of the dependencies existing among them, activities within different local scopes become strongly interdependent (Franke & Charoy 2010). In fact, dependencies serve as the bridges between local scopes of multiple actors. Although an actor is only responsible for and interested in the activities within her/his local scope of work, the

activities outside the local scope can still potentially impact her/his work through dependencies. For instance, although transporting a victim to the assigned station is out of the local scope of the decontamination manager, it can leads to the success or delay of the fact that the victim located in the station, which is a goal inside the local scope of the decontamination manager. As a result, modeling dependencies becomes an important task to evaluate the implications of remote activities transcending multiple local scopes. Dependencies within a collaborative process can be of various forms and interweave with each other in different ways. In emergency response operations, we can identify several types of dependencies between activities: *resource-related* dependencies consider the management of shared resources involved in multiple activities (Crowston 1994). *Goal-related* dependencies reflect the fact that one activity depends on the other activity to bring about a certain state in the world (Yu & Mylopoulos 1993). *Constraint-related* dependencies reflect the temporal and spatial constraints between multiple activities, such as executing two activities in a certain temporal order, or at the same location.

The SharedPlan-based model of activities and dependencies

To capture the three levels of constructs in a computational environment, we employ the computational theory of SharedPlan (Grosz & Kraus 1996) to represent collaborative activities as shared plans, and then use this model of activities to derive the knowledge about local scopes of work and dependencies. Figure 3 shows an example of the model that captures a portion of major activities and dependencies in the scenario.

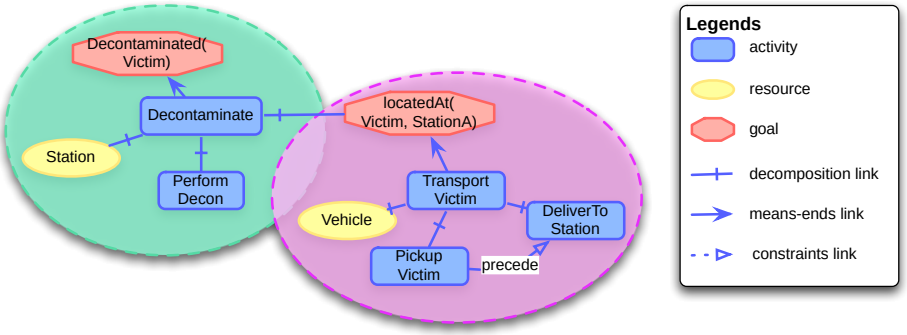


Figure 3. Modeling activities and dependencies

Following the SharedPlans model (Grosz & Kraus 1996), our model of the activity is a collaborative plan that captures a moment-to-moment representation of an unfolding activity and includes not only a hierarchy of actions but also the set of mental states (beliefs, intentions, and commitments) that the participating agents have established towards the plan and its sub-plans. A plan can be partial, meaning that the activity is still ongoing and unfolding. During the performance of an activity, the collaborative plan is updated to reflect the changes. By representing a collaborative activity as shared plans, the major components (*activity*, *resource*, *goals*) of collaborative activities can be matched to corresponding elements in a shared plan (*action*, *parameter*, *pre-condition*). Besides, the hierarchical structure of a shared plan reflects various relationships between these elements. A *means-ends* relationship indicates a relationship between an *end* – which can be a goal to achieve, or a resource to be produced – and a *means* - in the form of an activity - for attaining it. A *decomposition* relationship indicates a relationship between an activity and its subsidiary components. The subsidiary component can be a goal indicating a precondition that needs to be satisfied before the activity can be executed, a resource that is used by the activity, or a subsidiary activity that needs to be executed so as to complete the higher-level activity. A *constraint* relationship indicates a type of constraint between two activities, such as one activity needs to be performed before the other one. Due to space limitations, it is not possible to give here a detailed account of our model of activities as shared plans. Readers can refer to previous studies (Yu & Cai 2010) for such an account.

By modeling the collaborative activities as shared plans, various types of dependency relationships between them can be inferred from the hierarchical structure of the plan, recipes, and constraints. A *shared-resource* dependency can be inferred from two decomposition relationships that start from two different activities, but point to the same resource. A *producer-consumer* dependency is represented as a combination of a means-ends relationship and a decomposition relationship, where one activity uses a resource that is attained by another activity. A *common-output* dependency is two means-ends relationships through which one resource is produced by two activities. *Goal-related* dependencies can be modeled directly by a decomposition relationship between two activities, where one activity depends on the other activity because the latter is a means to achieve a subsidiary goal that must be satisfied in order to perform the former. Alternatively, goal-related dependencies can be indirect and mediated by an intermediate goal. In this case, there exist a decomposition relationship

between an activity and a subsidiary goal and a means-ends relationship from the goal to another activity. Temporal and spatial dependencies are represented as constraint relationships between activities.

Besides the intentional structure of activities, the SharedPlan model also captures each actor's current work focus. The knowledge about each actor's work focus allows us to derive the local scope of work for each user as all the elements in the shared plan that have direct relationships with the user's current work focus. For instance, the *decontaminate* activity itself together with the condition that the victim needs to be decontaminated, and all the subsidiary components of the *decontaminate* activity are in the local scope of the decontamination manager.

Event-driven awareness mechanism

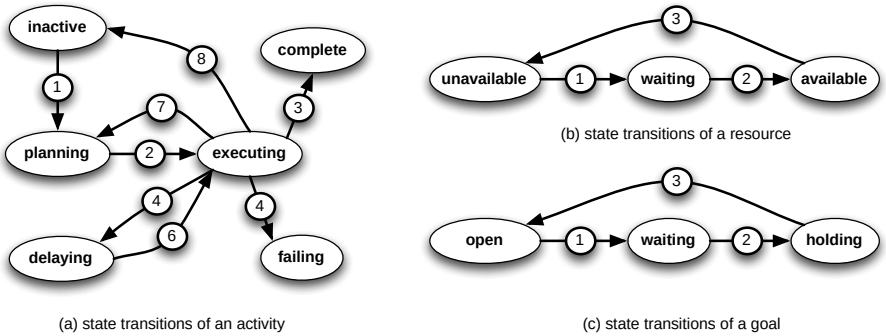
We propose an event-driven awareness mechanism following the complex event processing (CEP) architecture (Etzion & Niblett 2010), and is performed in three major steps: (1) Sensors recognize external events, which are represented as event objects within the system. (2) Event reasoning components reason on top of these initial events to understand their effects, and produce new derived events. (3) All the relevant initial and derived events are notified to the users based on their subscriptions. Here we first discuss the event definition and identification, and then focus on the event reasoning process, especially how the model of activities and dependencies is used in the reasoning. The event subscription and notification is described later.

Definition and identification of events

In this paper, we consider events as representing changes concerning the environment, resources, goals or activities that have impacts on other activities in an emergency response operation. The impacts of events on activities can differ from simply providing a piece of context information to more decisive events, such as triggering a new activity, delaying, disabling or causing re-planning of an ongoing activity.

We make distinctions between *external* (in the environment) and *internal* events (in the response activities). *External* events are changes in the physical environment that have impact on the performance of response activities. For instance, the occurrence of a traffic accident may block the traffic flow, which makes an actor's activity of delivering equipment to a medical station impossible to finish on time. *Internal* events are state changes on the basic elements of response operations, i.e. changes on the state of any resources, goals, and activities, such as the exceeding capacity of a medical station, the completion or delay of an ongoing activity.

Not all the external events in the environment are important for the actors to perform their emergency response activities. Rather, only a subset of the external events that can lead to changes within the response operations (i.e. the internal events) is meaningful. For a given application domain, a set of *external* events can be identified through scenario-based analysis (Friberg et al. 2010). First, a list of representative scenarios in the application domain is generated by utilizing reports from previous incidents. Then by modeling and analyzing the processes in the scenarios, external events that occur and have an effect on the actions and decision-making in the scenarios can be identified (Truptil et al. 2012).



(a) state transitions of an activity

(b) state transitions of a resource

(c) state transitions of a goal

Figure 4. Internal events as state transitions

On the other hand, the *internal* events are identified based on our computational model of activities. We define internal events as meaningful state transitions of activities, resources, and goals in our model. At a given time, an activity can be at one of the following execution states: *inactive*, *planning*, *executing*, *complete*, *failing*, and *delaying*. Each goal can be *open*, *waiting*, or *holding*. Each resource can be *unavailable*, *waiting*, or *available*. Figure 4 shows all the possible internal events that are defined by these execution state transitions.

Dependency-based event reasoning

The event reasoning is performed in two steps. First, the initial events generated by sensors need to be associated with the activity model, by evaluating the effects of these events on each activity represented in the computational model, and generating a set of internal events indicating any changes to impacted activities (*event interpretation*). Then, the set of internal events will be propagated through the network of dependency relationships to understand their chain effects on other activities (*event propagation*).

Event interpretation. The goal of event interpretation is to associate the initial events with the computational model of activities and dependencies, so that they can be propagated within the network of dependencies during the event propagation step. The event interpretation is performed by traversing through each activity node in the SharedPlan-based model, and checking whether the initial event can lead to certain changes of each activity, resource, or goal. The interpretation process is guided by a set of interpretation rules defined on top of the TESLA event specification language (Cugola & Margara 2010). Each interpretation rule includes two parts: an event pattern description that defines the characteristics of incoming events satisfying a set of constraints, and a set of effects that represent the state changes in the activity model when the event pattern is matched.

Event propagation. After the initial events are associated with internal state changes on some activities in our computational model, they can be further propagated to other activities that depend on them. Because the multiple dependencies in the model form a network, where the nodes represent the basic elements of activities, and the links represent the dependency relationships between them, we can adopt network-based reasoning models to perform the propagation. In this study, we use the Constrained Spreading-Activation model (Crestani 1997) to perform the propagation, which includes a basic spreading activation process and a set of constraints to control the spreading. The basic spreading activation process is a recursive process that starts with the activities that are associated with the initial changes during the interpretation step. In the beginning, all the activities with initial changes are labeled as *active*, and all the activities have direct dependency relationships with them are checked to identify possible state changes. If any new state changes are identified, the corresponding activities are also labeled as *active*, and the spreading process continues. To avoid the activation ending up spreading all over the network, a set of constraints is implemented to control the spreading process. In this study, we apply two types of constraints: *path constraints* and *activation constraints*. Path constraints are applied to control the spreading towards a particular direction. For example, a single path constraint can be: if the state change of an activity *A* is from *executing* to *fail*, and the parent of this activity *B* has no other plan to achieve, i.e. *A* is a required sub-act in every plan of performing *B*, then B is attributed with a state change from *executing* to *fail*. On the other hand, activation constraints are applied on single nodes in the dependency network to indicate whether certain state changes will make them active. For example, the decrease in the quantity of a resource will not spread out until it drops below a certain threshold. The development of constraints for the spreading activation is a heuristic process.

An example. To demonstrate the event reasoning process, we consider an event (*E1*) in the scenario that indicates the occurrence of a traffic jam at a certain location (Figure 5).

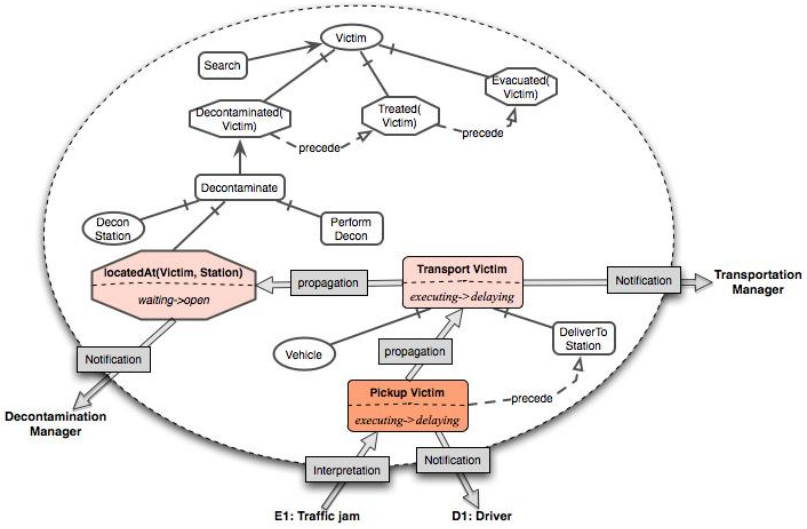


Figure 5. The event reasoning process: an example

When the system detects this event, the interpretation agent first attempts to associate this event with possible changes to current activities. A simple interpretation rule may define that if the traffic jam happens at a location that is on the route of a transportation activity, the transportation activity will be delayed (*Rule1*). By checking all the transportation activities in the dependency model, the interpretation module finds that the current activity of a driver (*D1*) to pick up a victim (*V1*) meets *Rule1*'s conditions. As a result, the interpretation module associate *E1* to *D1*'s activity of picking up *V1* (*Act1*) as a state change from *executing* to *delaying.*

This initial state change (*Act1: executing->delaying*) then triggers the propagation process. It starts with the *Act1* to check all the nodes that have a direct dependency relationship with *Act1*. Following the path constraint that if a subsidiary activity is delayed, the higher-level activity will also be delayed, a new state change to the higher-level activity of *Act1*, i.e. the activity to deliver the victim *V1* to the designated decontamination station (*Act2*) is derived: *Act2: executing->delaying*. The inference continues from *Act2*, and finds out that due to the delay of *Act2*, the condition (*Cond1*) that the victim *V1* must be delivered to the station in 30 minutes cannot be achieved, i.e. a new state change to the condition is derived: *Cond1: waiting->open*. These derived state changes are then further processed until they reach the boundary of the network, or violate any of the activation constraints.

Event notification

Event notification provides the publish/subscribe interaction between the system and users (Eugster et al. 2003). Users can subscribe to a set of events or patterns of events that are interesting to them. Then the system will monitor both the initial and derived events generated in the reasoning process, match them with users' subscriptions, and notify them whenever a match is found. Unlike many existing event subscription methods that ask users to describe their interests on every possible type of events generated in the system, our system only asks users to subscribe to events that are directly relevant to their activities, i.e. only the changes that are within their local scopes of work. Then the system monitors derived events generated by the reasoning agents and filters them out based on the local scopes. If a certain change is within a user's local scope, the user's subscriptions are loaded, and then are matched with the change. If a match is found, the user will be notified in the way as defined in the subscriptions.

IMPLEMENTATION

To validate our computational framework, a prototype system DACE (Dependency-based Awareness and Coordination Environment) is implemented in this study. A screenshot of the web-based client is shown in (Figure 6). The server side is implemented in Python, and contains four functional modules (*interpretation, inference, subscription, notification*) and three data repositories (*dependency, rule, and subscription*) (Figure 7).

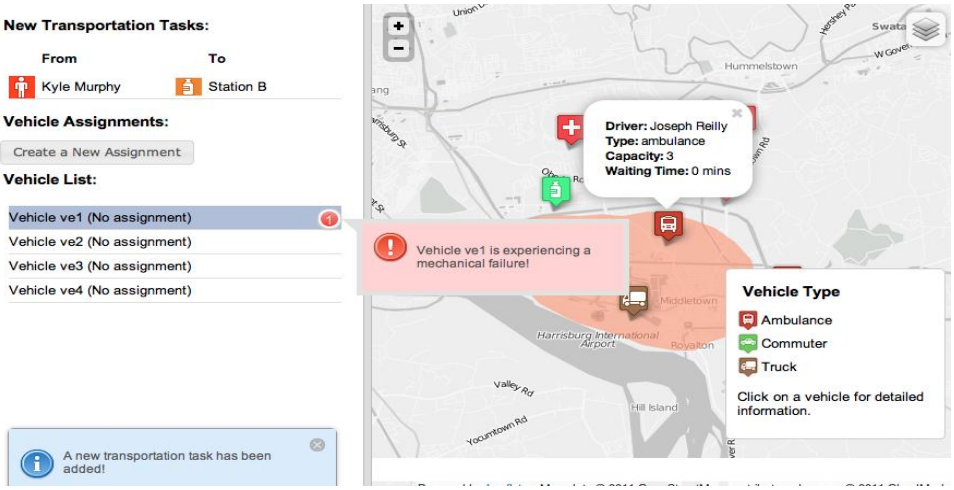


Figure 6. A screenshot of the web-based interface

Interpretation and *propagation* modules implement the event reasoning with the model of activities and dependencies described in the previous section. Both modules use rule-based event processing agents (EPA) (Etzion & Niblett 2010), and are powered by the PyKE (http://pyke.sourceforge.net/) rule-based inference engine to perform the reasoning. The *interpretation* agent monitors all the new events fed into the system and attempts to associate them with certain state changes in the dependency model, which is driven by a set of interpretation rules stored in the rule repository. The output of *interpretation* agent is a set of internal events that are then received by the *propagation* agent. The *propagation* agent follows the spreading activation model to

propagate the state changes through the dependency model, with constraints defined as a set of propagation rules.

The *subscription* and *notification* module are implement as a web-based event notification service providing storage and management for subscriptions and efficient delivery of events. The event subscription module includes a web-based interface to ask the user to identify all relevant events that are within his/her local scope of interest. The user can assign the level of relevance for each event/event type as high, medium, or low. In addition, the user can specify the notification style for each event, from less obtrusive way (e.g. badge, banner) to more obtrusive ways (e.g. sound, alert). The subscriptions made by the users are stored in the *subscription* repository that can be easily queried by the notification module. The long-polling technique is used to provide the push service provided by the notification module. If a match between an event and a user's subscription is found, the notification module will push the event with the notification style to the user's client.

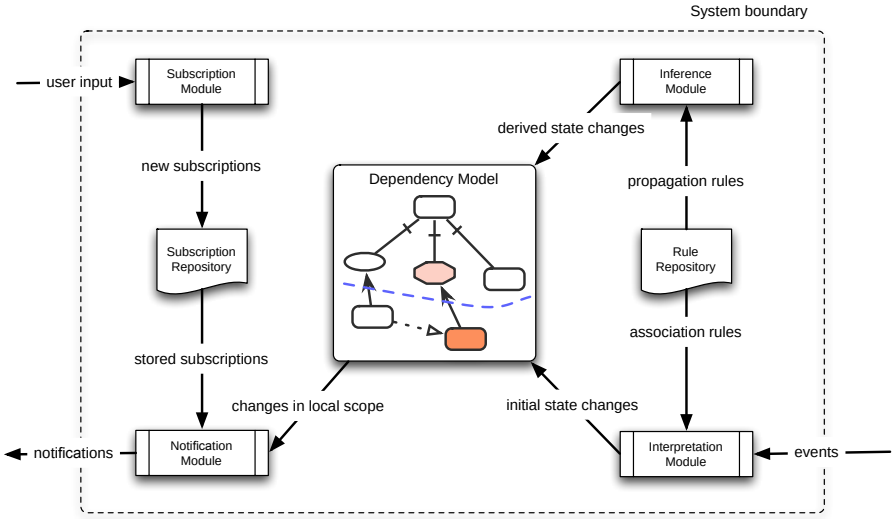


Figure 7. System architecture

DISCUSSION AND CONCLUSION

In this paper, we proposed our solution to scaling up the event-based awareness mechanism to support coordination in emergency response operations characterized by higher level of complexity and contingency. Our approach attempts to balance the efforts of human actors and the computational system in coordination with the support of the SharedPlan-based activity/dependency model. While human actors are responsible to express their awareness interests within their local scopes of work though event subscription, the system is designed to serve as a cognitive aid to human actors in two senses: (1) it represents and maintains a group mental model of the overall collaborative activities and their interdependencies; (2) it helps the users to determine the relevance of events by automating the reasoning on how an event impacts the states of activities as its effects propagate through the web of dependencies. We believe that such a balanced model of awareness support can scale better with the increasing complexity of emergency response operations, without losing the flexibility of awareness-based coordination mechanisms.

Our preliminary scenario-based evaluation has demonstrated the potential of using dependency knowledge in the process of event reasoning. However, a formal user-centered evaluation has not been conducted and therefore how significantly the system can improve coordination in real-life situations is still not clear. We believe that the performance of the system largely depends on the quality of various types of knowledge built into the system, such as the domain knowledge about the activities, dependencies, and the event reasoning rules. As a result, we consider the future evaluation of the system as an iteratively analytical process, where the initial results from the evaluation will allows us to collect and refine these types of knowledge from human experts/users, which is subject to further evaluation.

This study can be further extended in several directions. First, the development of reasoning rules is a heuristic process. The current sets of interpretation and propagation rules in DACE are generated based on a scenario-based analysis. A knowledge elicitation study with domain experts will be conducted in the future to enrich the knowledge base. Also, the constrained spreading activation model is only one of the existing reasoning strategies that can be applied in event propagation. We plan to investigate other methods, such as inference networks, and compare them with our current rule-based model. In current system, we assume the event reasoning process is an automatic process where human users are not active participants. However, we believe

that in complex collaborative situations, human involvement is inevitable. Our next step also involves explicitly focusing on human interaction and collaboration, and emphasizing the role of human actors in the event reasoning process.

REFERENCES

1. Chen, R. et al., 2008. Coordination in emergency response management. *Communications of the ACM*, 51(5), pp.66-73.
2. Crestani, F., 1997. Application of spreading activation techniques in information retrieval. *Artificial Intelligence Review*, 11(6), pp.453-482.
3. Crowston, K., 1994. A taxonomy of organizational dependencies and coordination mechanisms. *MIT Center for Coordination Science Working Paper*. Available at: http://ccs.mit.edu/papers/CCSWP174.html.
4. Cugola, G. & Margara, A., 2010. TESLA: a formally defined event specification language. In *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems*. pp. 50-61. Available at: http://portal.acm.org/citation.cfm?id=1827427 [Accessed November 11, 2011].
5. Etzion, O. & Niblett, P., 2010. *Event Processing in Action*, Manning Publications Co.
6. Eugster, P.T. et al., 2003. The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)*, 35(2), pp.114-131.
7. Faraj, S. & Xiao, Y., 2006. Coordination in fast-response organizations. *Management Science*, 52(8), pp.1155-1169.
8. Franke, J. & Charoy, F., 2010. A Model for Temporal Coordination of Disaster Response Activities. In *Proceedings of the 7th International ISCRAM Conference*. Seattle, USA, pp. 1-11.
9. Friberg, T. et al., 2010. Using Scenarios for the Identification of Real-World Events in an Event-Based System. In *Proceedings of the 7th International ISCRAM Conference*. Seattle, USA, pp. 1-5. Available at: http://www.iscram.org/ISCRAM2010/Papers/164-Friberg_etal.pdf.
10. Gonzalez, R.A., 2008. Coordination and its ICT support in Crisis Response: confronting the information-processing view of coordination with a case study. In *Hawaii International Conference on System Sciences, Proceedings of the 41st Annual*. p. 28.
11. Grosz, B.J. & Kraus, S., 1996. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2), pp.269-357.
12. Malone, T.W. & Crowston, K., 1994. The interdisciplinary study of coordination. *ACM Computing Surveys (CSUR)*, 26(1), pp.87-119.
13. Pottebaum, J. et al., 2011. Event Definition for the Application of Event Processing to Intelligent Resource Management. In *Proceedings of the 8th International ISCRAM Conference*. Lisbon, Portugal.
14. Prinz, W., 1999. NESSIE: An Awareness Environment for Cooperative Settings. *Proceedings of the sixth conference on European Conference on Computer Supported Cooperative Work*, pp.391-410.
15. Schmidt, K., 1991. Riding a tiger, or computer supported cooperative work. In *Proceedings of the second conference on European Conference on Computer-Supported Cooperative Work*. pp. 1-16.
16. Shen, S.Y. & Shaw, M.J., 2004. Managing coordination in emergency response systems with information technologies. In *Proceedings of the Tenth Americas Conferences on Information Systems*. pp. 2110-2120.
17. Truptil, S. et al., 2012. Nuclear Crisis Use-Case Management in an Event-Driven Architecture. *Lecture Notes in Business Information Processing*, 99(6), pp.464-472. Available at: http://www.springerlink.com/index/P2456X5670343467.pdf [Accessed February 20, 2012].
18. Turoff, M. et al., 2004. The design of a dynamic emergency response management information system (DERMIS). *Journal of Information Technology Theory and Application (JITTA)*, 5(4).
19. Yu, B. & Cai, G., 2010. Using Intentions and Plans of Mobile Activities to Guide Geospatial Web Service Composition. In *Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific*. pp. 141-148.
20. Yu, E.S.K. & Mylopoulos, J., 1993. An actor dependency model of organizational work: with application to business process reengineering. In *Proceedings of the conference on Organizational computing systems*. pp. 258-268.