Framework for Coordination of Activities in Dynamic Situations

Jörn Franke, François Charoy, Paul El Khoury

Enterprise Information Systems
Vol. 00, No. 00, February 2008, 1–26

RESEARCH PAPER

Framework for Coordination of Activities in Dynamic Situations

Jörn Franke[a]*, François Charoy[a] and Paul El Khoury[b]

[a]*LORIA-INRIA-CNRS, Université de Lorraine, Nancy, France*;[b]*SAP AG, Germany*

(*v2.2 released September 2008*)

Recent disasters, such as Hurricane Katrina in 2005, have shown several issues for coordination of human activities in these dynamic situations. Contemporary tools for coordination used in the disaster response, such as e-mail, Whiteboards or phone, only allow for unstructured coordination, which can cause coordination problems. Hence, we discuss current information systems for coordinating activities in a structured manner and identify their weaknesses in context of a process modeling effort conducted together with experienced disaster managers. Afterwards, we propose a framework for coordination of activities in dynamic situations. The framework presented in this paper has been implemented as an extension to an open collaboration service. This shows how it can be used in context of other tools required for disaster response management, such as maps, pictures or videos of the situation. The work described here is the foundation for enabling inter-organizational coordination of activities relevant in other domains, e.g. enterprise support processes, production processes or distributed software development projects. Furthermore, comments by disaster managers show that the concepts are relevant for their work. The expected impact is a more effective and efficient coordination of human activities in dynamic situations by structuring what needs to be coordinated.

Keywords: evolving,process,activity,temporal,dependency,framework

*Corresponding author. Email: loria@joernfranke.net

ISSN: 1751-7575 print/ISSN 1751-7583 online

DOI: 10.1080/1751757YYxxxxxxxx
http://www.informaworld.com

1. Introduction

We have seen over the last years a rise of disastrous events. Disasters can be related to natural hazards, such as Hurricane Katrina in 2005 or the Haiti Earthquakes in 2010. However, they can also be man made, for example, the 9/11 terrorist attacks. During a disaster many organizations respond with the goal to save people and enable them to live a normal life again. The situation is dynamic and the response processes evolve according to shifting goals of these organizations. They are driven by humans based on their judgment of a complex situation as described by Gartner and McKinsey (cf. Olding and Rozwell (2009)).

Since the organizations involved in a disaster response have different skills, resources and capabilities, they need to coordinate their efforts to help the people as much as possible. Examples for problems related to coordination are that nothing is done, double efforts are made or contradictory actions are performed. For example, during Hurricane Katrina search and rescue operations were conducted by different organizations (Unknown (2006)). This led to cases where rescued people were left on highways with no sheltering or food provision as well as nobody took care of them. Some areas were searched several times, but other areas not at all. The problem of shifting goals was also evident in the case of Hurricane Katrina: People were first brought to a location where food and sheltering was provided, but have been later moved to another location without notifying the other organizations, so that food and sheltering was not provided at the new location.

We present in this paper a framework for temporal coordination of activities in dynamic situations. The underlying idea is that people can model what has been done, what is currently going on and what can be done next. This model needs to be syntactically correct, otherwise typical coordination errors may occur. Furthermore, in order to deal with shifting goals and dynamically evolving processes, the users need to adapt the model correctly at any point in time. If this is not possible due to work overload or incomplete information then deviations from the model need to be highlighted to the users. We use the domain disaster response management as a critical example for dynamic situations. However, our approach is not limited to this domain. Dynamic situations can occur in enterprises, for example, when responding to a complex security incident covering several departments or complex support requests from customers to a software vendor. They may also occur in the area of public services. For instance, large events, such as the Olympic games, involve also many stakeholders that need to coordinate. Another example can be found in the area of humanitarian supply chain management (cf. Franke *et al.* (2011)).

Our approach is based on the framework provided in Franke *et al.* (2010). This framework allows modeling activities and their temporal dependencies. We provide in this paper more details on the translation of the model to a temporal constraint network (see Allen (1983)) to ensure its correctness. Furthermore, the framework is able to detect violated dependencies to highlight shifting goals not taking into account the dependencies between different activities. We describe in this paper, how dependencies that cannot be synchronized due to alternative choices by the user can be detected. Additionally, we provide insights from disaster managers about our approach. The main benefit of the framework is that it supports the user to avoid some of the coordination problems occurring during a disaster using traditional tools.

The paper is organized as follows. In the second section, we describe the dynamics of the domain crisis management in more detail by providing a scenario derived from the SoKNOS project together with experienced disaster managers. Afterwards, in the third

section, we investigate the state of the art and the limitations of existing approaches with respect to information system support for structured coordination of activities in dynamic situations. We also explain why contemporary business process modeling notations are difficult to apply in our scenario based on a process modeling effort conducted together with end users, such as police men or fire fighters. We then present the framework and our extensions for temporal coordination of activities in the fourth section. The implementation of the framework in a distributed collaboration service is explained in the fifth section. The goal of the implementation was not only to demonstrate technical feasibility, but also to demonstrate how our concepts work in context of other tools required for the disaster response. The sixth section is dedicated to comments by disaster managers about our framework.

2. Dynamic Situations - The Case of Crisis Management

Crisis response management is a critical example for dynamic situations. Results from this domain are beneficial for business information systems research, because globalized enterprises need to act more quickly and agile in dynamic economic environments (cf. Olding and Rozwell (2009)). During our research in the SoKNOS (Service-oriented ArchiteCtures Supporting Networks of Public Security - Döweling *et al.* (2009)) project, we found out that contemporary tools have limitations with respect to the coordination of response activities. For example, the disaster managers, such as police commanders or fire chiefs, use phone, whiteboards e-mail or fax for coordinating their activities. Although these means allow coordinating in an ad-hoc manner, they have limitations: They only allow for very unstructured coordination. It is very difficult to detect conflicts in what needs to be coordinated. For instance, the relations between the response activities are usually not described or clear to the people. Another problem is to find deviations from what has been done and what was expected to be done. These deviations may occur, because goals of organizations shift. There may be the goal to protect a residential area from a flood, but this shifts towards evacuating the area. This has to consider the dependencies to other activities that are currently executed, otherwise typical coordination problems may occur. In order to understand these issues better, we present an illustrative scenario.

2.1. *Scenario*

The following scenario describes a response to a flood disaster (see Figure 1). It is based on previous flood disasters and it has been developed together with end users, such as fire chiefs and police commanders, in the SoKNOS project. Many organizations respond to the flood. We illustrate three of them in more detail: police, fire brigade and military. The fire brigade is building a dam to protect a residential area from a flood. It relies on the provision of sandbags by the military. The military fills and transport sandbags. It is also responsible for protecting a chemistry plant from a flood by building another dam. Each of the activities may be related to other ones. For instance, building a dam requires the provision of sandbags. Activities themselves are also different: There are simple activities, such as warning the people that the area is going to be evacuated, but also more complex ones. For example, building a dam may involve governance arrangement, such as approval procedures. Another example for governance arrangements is that the fire chief can cancel transporting sandbags by the military, because they do not need them anymore. Similarly,

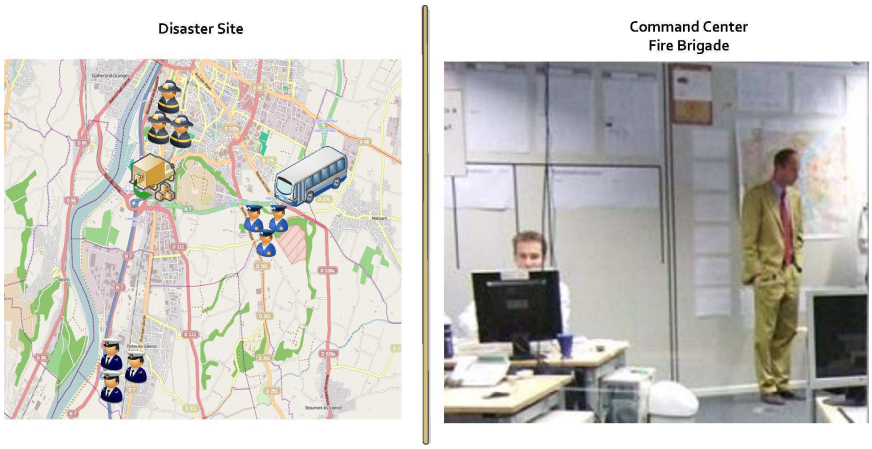

Figure 1. Scenario

the military may declare that transporting sandbags for the fire brigade has failed.

The situation is dynamic and this means that goals of the organizations shift. For instance, the police evacuate a residential area, because building a dam by the fire brigade fails. This has to take into account the relations between activities. If building a dam fails then there is no need anymore to fill and transport sandbags for building it.

2.2. *Conclusion*

We argue that a more structured approach is beneficial for coordination in dynamic situations. A more structured approach means that activities and their relations are explicitly modeled by the users. We expect that typical coordination problems occurring in these scenarios. when using tools, such as mail, paper, fax, mission diary or Whiteboard, can be addressed partially by such a structured approach. For example, process management approaches seems to be suitable to overcome the limitations of the contemporary means. They allow modeling activities and their dependencies in a structured manner. This enables reasoning about correctness of the model of activities and dependencies. Furthermore, deviations from the model and what has been done can be detected. Users can thus deal with the situation at the right time when they are available. We investigate in the next section how more structured approaches can be utilized in dynamic situations, such as disaster response management.

3. State of the Art

We have explained before that information system support can be beneficial for coordinating response activities. In the literature, we find several approaches for coordinating activities. More specifically, we identified four categories of approaches: process-based, artifact-based, rule-based and integrated. We explain in the following subsections related technologies to these categories and their suitability for dynamic situations.

3.1. *Process-based Approaches*

Process-based approaches have been proposed by various scholars to manage business processes (cf. van der Aalst *et al.* (2003)). In a process-based approach, users model

explicitly parallel or alternative activity sequences. They may further describe human resources, data or applications relevant for execution of a process. This process model has to be fully specified with all possible alternatives in advance. The idea is that the process can be executed frequently in a standardized and cost-efficient manner. Thus, processes have few exceptions from the model. Workflow or process management systems enable the coordination of such a modeled process fully controlled by the system. They track which activity needs to be executed next and make the application, data as well as resources required for it. Nevertheless, one of the main criticism of process modeling and management systems is that they are not very flexible as it has been shown in practice (cf. Bowers *et al.* (1995), Grinter (2000)). The reason is that they are mostly addressing administrative routine processes (see Becker and Rosemann (2010)).

However, more flexible process management systems have been researched intensively (cf. Grigori *et al.* (2001), Dadam and Reichert (2009), Huth *et al.* (2001), Medina-Mora *et al.* (1992), Gaaloul *et al.* (2010)). Such systems have also been proposed for routine emergency scenarios (cf. Mak *et al.* (1999), Rüppel and Wagenknecht (2007), Georgakopoulos *et al.* (2000), Fahland and Woith (2009), de Leoni *et al.* (2007), Reijers *et al.* (2007)). They have in common that they are still focusing on alternative and parallel activity sequences that need to be fully specified in advance. They are more flexible in the sense that the process models can be adapted to deal with exceptional situations. For instance, activities can be correctly inserted into a sequence or resources can be dynamically assigned to activities without introducing errors in the process execution controlled by a system. Thus, they can represent more accurately the executed process. Nevertheless, they only address situations with few exceptions, but in a crisis everything is an exception. This motivated us to try modeling a typical response process together with end users, such as police commanders and fire chiefs. We illustrate one result in Figure 2. It is about a train accident with hazardous material from the point of view of the police. Several organizations, such as police or fire brigade, are responding to the disaster. In the first part, the organizational structure of the response is created, it is defined who is in charge and other organizations are informed about the accident. In the second part, some typical response activities and their relations are described.

Based on our end-user interviews and the process modeling effort, we identified the following challenges with the process-based approach:

- *Full specification of all activities and their relations.* Describing a full specification and maintaining it has been difficult during our process modeling effort. In a realistic response, the model is subject to continuous change and it gets very complex. Additionally, the required changes are difficult to anticipate (cf. Klein (1999), Landgren and Nulden (2007)).
- *Using a workflow or business process system to enforce the process.* This means that the model needs to be adapted instantaneously when goals shift to reflect a reassessment of activities and their relations. In a disaster response, this won't be always possible due to the situation. Furthermore, a system should support the user when reassessing activities and their relations. It cannot enforce a process out of control of the system. A human-driven process, such as a disaster response process, is a typical example for a process out of control of a system.
- *The limited expressivity of relations.* Process-based approaches consider only parallel or alternative activity sequences. From our process modeling effort, we conclude that it is very difficult to decide if a relation between activities should be sequential or parallel. Moreover, there needs to be further relations, such as overlapping activities, as it can be derived from the building a dam example. There, sandbags are filled and

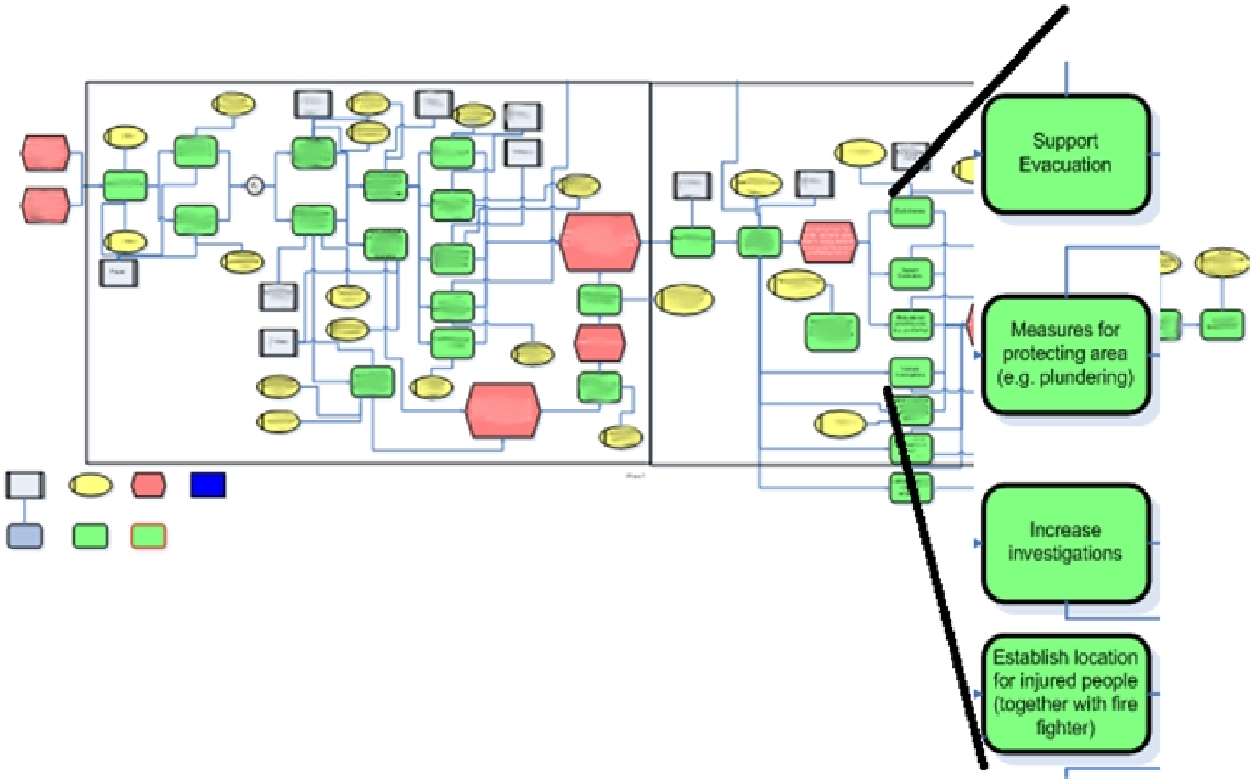

Figure 2. Example model of a response process for responding to a train accident with hazardous material (from the perspective of the police)

some of them are already transported to the disaster site. This problem has already been identified in the BPM community by Dumas et al.: "[..] even today's workflow management systems enforce unnecessary constraints on the process logic [..] processes are made more sequential than they need to be" Dumas *et al.* (2005).

Further processes have been documented with three other organizations, but have shown similar results. We conclude that process-based approaches have some advantages, such as explicit modeling of activities and their relations. The user has a good understanding what has been done, what is currently going on and what can be done next. There are many different frameworks for ensuring correctness of business process models (cf. Fahland *et al.* (2009)). However, there are major limitations, such as requiring full specification of the process and enforcement of the process by the system. Deviations from the process caused by shifting goals cannot be detected. Thus, all changes to the model have to be done in time which seems to be an unrealistic assumption in a disaster response. Finally, the possible relations between activities are not very rich. It is only possible to model parallel or alternative activity sequences.

3.2. *Artifact-based Approaches*

Recently, research scholars have proposed an alternative approach to deal with dynamic ad-hoc processes: the artifact-based approach (see Bhattacharya *et al.* (2007a)). It may also be known as adaptive case management (see van der Aalst *et al.* (2005)). This approach is centered around artifacts that mediate the relations between activities. Exam-

ples for artifacts are data objects, such as an invoice, an insurance case, a bill of material for a product or an order list. Each artifact is described by a state machine describing the possible state transitions of the artifact. For instance, an insurance case can change from the state "Initial" to "Activated". Depending on the state of an artifact, certain activities can be executed. The outcome of the execution of activities is a state change of an artifact. For example, an insurance can only be paid out when all necessary data of the customer has been entered before. Examples for these approaches can be found in Bhattacharya *et al.* (2007b), Gerede and Su (2007), van der Aalst *et al.* (2005), Müller *et al.* (2007), Vanderfeesten *et al.* (2011). They also explain how an artifact-centric model can be verified for correctness.

The artifact-based approach has similar advantages as the process-based approach. Activities and their relations are modeled explicit via the artifact. However, we found several challenges with this approach, which can also partly be found in process-based approaches. There needs to be a full specification of artifacts and their relations. The possible relations between activities are not very rich. Basically, only sequential and parallel relations can be specified via the state transitions of the artifact and the pre-and post-conditions of activities. Finally, it was difficult to find relevant data objects in a crisis that should be the focus of the activities and their relations. For instance, it was not clear if a house in a residential area, a fire truck or a dam should be modeled. The approach seems to be more relevant for digitalized business artifacts, but less for crisis response situations.

3.3. *Rule-based Approaches*

Rule-based approaches describe the relations between activities using rules. They may be also known as constraint-based approaches. Different visualizations are possible for rules and activities. Nevertheless, the most common visualization is a textual logic description. An example for a rule can be "IF build_dam fails THEN start evacuation". Various formalisms have been proposed to model the relations between activities using rules. These formalisms ensure a correct specification of activities and their relations. For example, Pesic (2008) present a formalism based on linear temporal logic (LTL). The author of this approach also explains how deviations from the specified model of rules and activities can be detected as well as highlighted to the user. Other approaches using other formalisms can be found in Skaf *et al.* (1996), Zahoor *et al.* (2010), Dayal *et al.* (1990), Dourish *et al.* (1996).

The rule-based approach has the advantage that nearly every relation can be defined between activities. Furthermore, it does not require a full specification of the activities and their relations. It does not require enforcing the relations between activities by system, but is also able to highlight deviations from a specification. Thus shifting goals affecting the relations between activities can be highlighted to the user. Nevertheless, there are limitations. It can be computational complex to ensure correctness of the specification of rules and activities (cf. Pesic (2008)). This means that the specification cannot always be adapted correctly in a timely manner. Furthermore, a rule-based approach makes it very difficult for the user to understand what has been done, what is currently going or what can be done next. The reason is that rules are usually specified in a textual manner and thus difficult to understand in an instant. Although visualizations have been proposed for logic rules, they are still difficult to understand - particularly for non-experts (cf. Pesic (2008)).

3.4. *Integrated Approaches*

The advantages of the previous mentioned approaches led to proposals to integrate two or more of them. For instance, we find in the literature approaches that integrate rules and processes (cf. Müller *et al.* (2004), Sadiq *et al.* (2001), Adams (2007), Pesic (2008)), rules and artifacts (cf. de Man (2009), Rahaman *et al.* (2009), Cugola (1998)) or artifacts and processes (cf. Wang and Akhil (2005)).

While integrated approaches combine the advantages of the previously presented ones, they do not overcome their limitations. For example, specifying processes and rules requires still specification of processes and rules as described before. Furthermore, integrating these approaches introduces new obstacles. It is required to specify more when using an integrated approach (e.g. rules and artifacts). It has to be decided what should be, for instance, specified as a process and what as a rule. Given our process modeling effort with domain experts explained before, we consider this as too complex for our scenario.

3.5. *Conclusion*

We have presented in this section different approaches for supporting the coordination of activities in dynamic situations. Basically all approaches have been developed for the business context. Some of these approaches require well-defined business goals (e.g. the process-based and the artifact-based one) and full specification of activities and their relations. This can lead to complex models and be time consuming. This makes it difficult to understand the effect of shifting goals leading to a reassessment of activities and their relations. However, the user can understand what has been done, what is currently going on and what can be done next, when applying these approaches. Nevertheless, the possible relations that can be specified are very limited. Rule-based approaches can deal with the issues of process-based or artifact-based approaches to some extent. However, ensuring correctness of rules can be computationally complex. This makes it difficult to adapt the specification of rules and activities in a timely manner. The user can hardly understand the relations between activities specified using textual description of rules. Unfortunately, integrated approaches combine not only advantages of the previous mentioned approaches, but also their limitations. They also require more specification effort by the user.

Given the limitations identified in this section with respect to the state of the art, we propose in the following section an alternative framework for coordinating activities in dynamic situations.

4. A Framework for Temporal Coordination

We identified in the previous sections the necessity that users need to be able to model correctly the activities and relations that they want to coordinate. This is important to create an understanding what has been done, what is currently going on and what can be done next (see Klein (1999), Weick (2000)). We describe in the next subsection how they can model activities and their relations. Afterwards, we explain how they can verify this model for correctness in the second subsection. They need to keep track of the execution of activities, so that they can detect deviations between what is defined in the model and how activities have been executed. We argued before that shifting goals of an organization lead to a reassessment of activities and their relations. If this leads to deviations from the model then this reassessment has not taken into account what has

already been executed. The user needs to be aware of this. This is presented in the third subsection.

4.1. *Modeling*

Given the scenario in section two, we assume that a user wants to describe a model how activities are coordinated. For instance, the fire chief wants to articulate that the activities "Fill Sandbags", "Transport Sandbags" and "Build Dam" are planned. However, these activities should only be executed if the overall activity "Protect Residential Area from Flood" is executed. Obviously, this requires that the current state of an activity needs to be described. We have elaborated further in section two that users need to coordinate different types of activities. For example, an activity for warning people is a more simple activity than an activity for building a dam. Furthermore, we focus on temporal dependencies describing the relations between activities. We will see later that we can provide a richer description of temporal dependencies than process-based approaches.

We describe in Definition 4.1 the concept of an activity type. It is usually modeled before any activities or dependencies are described. It articulates the difference between activities by means of the lifecycle describing them. For example, more simple activities, such as warning people that an area is evacuated, can have the following simple lifecycle: Firstly, the activity is planned, then executed and then finished. More complex activities may require additional approval steps or can be canceled. For instance, building a dam is such an activity. The complexity of a lifecycle is determined by the number of states it contains.

Definition 4.1 An *activity type* $at_d = (S, st, se, f)$ represents the management lifecycle of an activity where

- 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
- $f : S \rightarrow S$ is a transition function defining the possible transitions from one state to another for one activity type

The activity type must not contain strongly connected components (i.e. cycles, cf. Tarjan (1972)). The reason is that it can confuse the users viewing or modeling activities and their relations. For example, an activity is changed from the state "Execute" to the state "Fail" and then again from "Fail" to "Execute". This is difficult to display and understood by the user. Particularly, when there are several users or when there is not always time to pay attention to the system. Furthermore, we describe in Franke (2011) how the activity type can be enhanced further by governance roles to be able to describe who can cancel an activity or is responsible for its execution.

Definition 4.2 describes formally an activity. An activity is a human action performed in the real world, such as transporting sandbags or evacuating an area. It is based on an activity type and has a current state. For instance, the activity for filling sandbags is based on the activity type for field operations. The activity "Fill Sandbags" is currently in state "Plan".

Definition 4.2 An *activity* is defined as $a_i = (uid, name, cs, cat)$ where

- uid is a unique identifier of the activity.

- *name* describes the activity
- $cs \in cat.S$ 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

The activity can also contain a user to governance role assignment. Furthermore, any further data can be attached to activities. An activity is independent of other activities. That means it can change its state without affecting other activities. A dependency can be established between activities if the user perceives it as important and if he is aware of it. However, establishing a dependency is optional. A temporal dependency is described formally in Definition 4.3.

Definition 4.3 A temporal dependency is defined as $d_i = (a_s, s_s, a_d, s_d, type)$ where

- a_s is the source activity
- s_s is the state of the source activity, whereby $s_s \neq a_s.cat.st \neq a_s.cat.se$ (no dependencies between start and end states are allowed)
- a_d is the destination activity
- s_d is the state of the destination activity, whereby $s_d \neq a_d.cat.st \neq a_d.cat.se$ (no dependencies between start and end states are allowed)
- *type* is the type of temporal dependency

A temporal dependency is established between two states of two different activities. We use Allen's thirteen time interval relationships to model different types of temporal dependencies (see Allen (1983)). Figure 3 illustrates seven of them, because the other six ones are just the inverse of the first six ones illustrated. For instance the time interval relationship "overlaps" has the inverse time interval relationship "overlapped by". This provides a richer description of temporal dependencies than sequential ones found in many process-based approaches. We do not allow temporal dependencies between start and end states of activities. The reason is that a dependency can only be established after an activity has been created. However, an activity is already in its start state when it is created. Furthermore, an activity is in its end state for an infinite amount of time. Thus, it is difficult to describe that an activity should be executed after another activity has been in its end state. This would imply that another state exist after an end state, which is excluded by definition.

The time interval relationships proposed by Allen have the following properties: qualitative, exhaustive and distinct. Qualitative refers to the fact that it is not needed to specify quantitative points in time (e.g. 5 hours and 5 minutes). We assume that this is very difficult to achieve and they would have to be constantly updated to reflect the current situation. Nevertheless, our framework does not prohibit defining quantitative points in time. The properties exhaustiveness and distinctness reduce the ambiguity what is meant by a temporal relationship. We chose Allen's rule formalism over other temporal rule formalisms (cf. Lutz *et al.* (2008)), such as linear temporal logic, because we assume that they can easier understood by humans and visualized more easily to them.

4.2. *Verification*

We mentioned in sections two and three that it is important that the user can ensure that the model of activities and temporal dependencies is correct. An example for an incorrect model is illustrated on the left side of Figure 4. Three activities "Fill Sandbags",

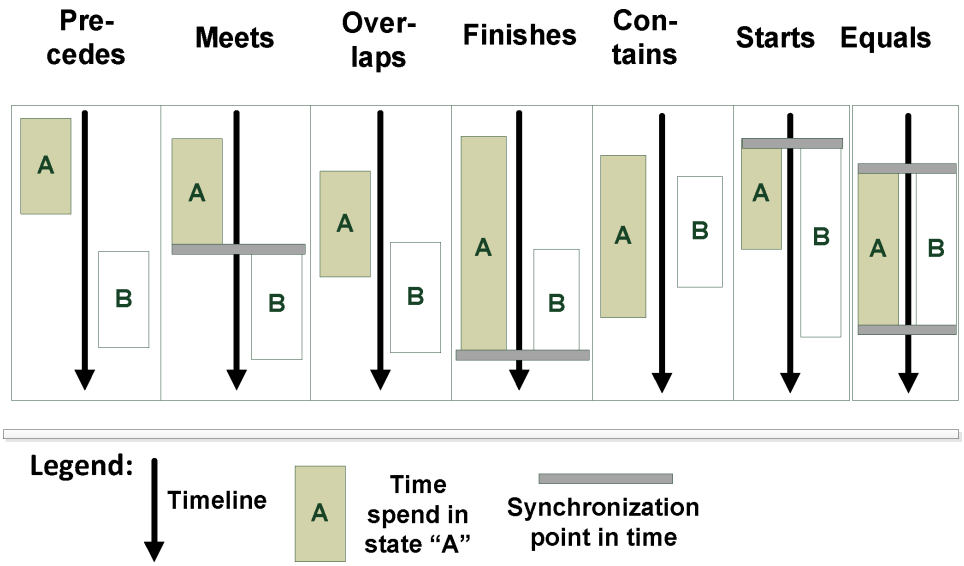

Figure 3. Types of temporal dependencies between states of activities (based on Allen's temporal interval relationships Allen (1983))

"Transport Sandbags" and "Build Dam" are modeled. The activities are based on same activity type illustrated in in the same figure. A dependency "overlaps" is established between the activity "Fill Sandbags" in state "Execute" and the activity "Transport Sandbags" in state "Execute". Another dependency "overlaps" is established between "Transport Sandbags" in state "Execute" and "Build Dam" in state "Execute". Finally a dependency "overlaps" is established between "Build Dam" in state "Execute" and "Transport Sandbags" in state "Execute". This basically means that the activity "Fill Sandbags" (or any other activity in this example) in state "Execute" overlaps itself, which is not possible.

Clearly, it should be avoided that incorrect models can be defined by the user, otherwise typical coordination problems as described before may occur. Furthermore, we cannot expect that the users themselves check manually the model, because they may not have the time in dynamic situations to do this. Thus, we propose an automatic verification procedure. We propose to use existing well-understood formalisms to verify that a model is correct. More specifically, we use temporal constraint networks and the path consistency algorithm to verify correctness of a model (cf. Allen (1983). This means we need to first translate our model into a temporal constraint network and then detect incorrect models using the path consistency algorithm over this temporal constraint network. We discuss these two steps in more detail in the following two paragraphs.

4.2.1. Translating Model into Temporal Constraint Network

In order to translate the model into a temporal constraint network, we need to define a temporal constraint network. It is formally described in Definition 4.4.

Definition 4.4 A temporal constraint network is defined as $CN = N \times N \times C$ where

- N is the set of nodes in the constraint network. A node corresponds to a time interval of a undefined finite length.
- $C \subseteq DT$ is the set of constraints between two nodes. Multiple constraints (e.g. precedes, meets) between two nodes describe that one of them is possible (e.g. precedes or meets).

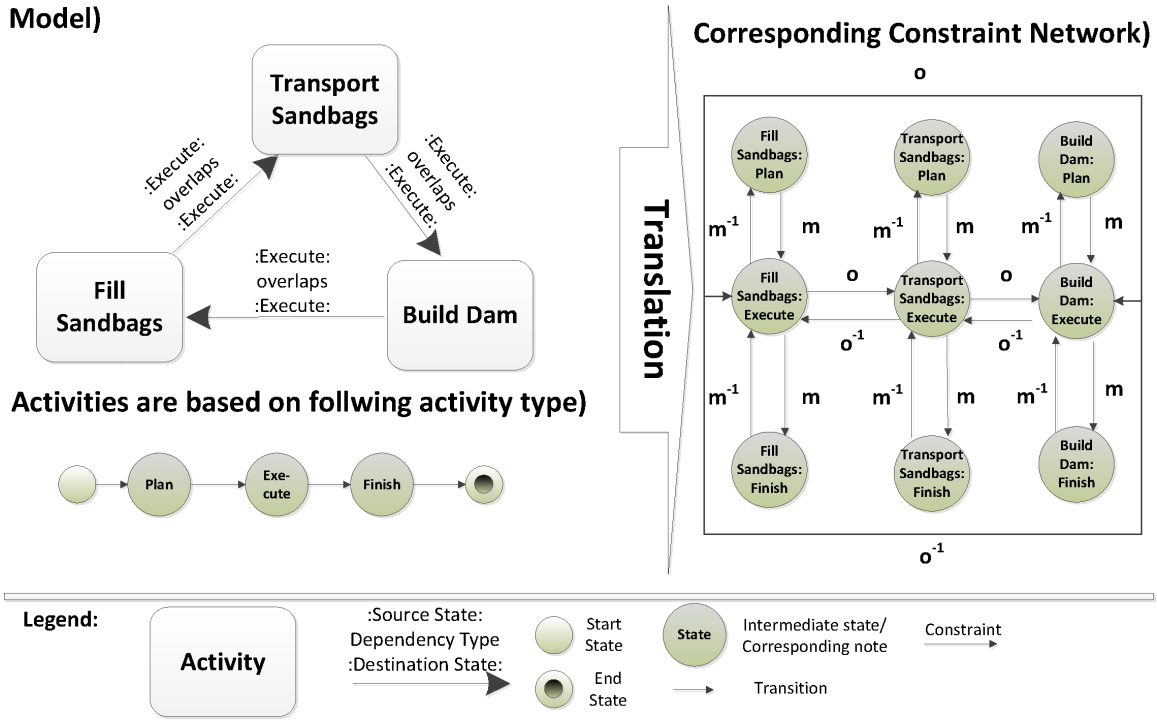

Figure 4. Example for translation of a model into a temporal constraint network

- $DT = \{precedes, precededby, meets, metby, overlaps, overlappedby, finishes, finishedby, contains, during, starts, startedby, equals\}$ is the set of possible constraints based on Allen's interval relationships Allen (1983).

The translation of a model into a temporal constraint network is guarded by the following rules:

(1) States of an activity (as defined in the activity type) are translated into nodes of the constraint network. A state describes a time interval of an undefined finite length. As mentioned, time intervals correspond to nodes in a constraint network.
(2) State transitions of an activity (as defined in the activity type) are translated into the constraint "meets" (m) between the corresponding nodes in the constraint network. The inverse direction is translated into the constraint "met by" (m^{-1}).
(3) A dependency is translated into the corresponding constraint between nodes in the constraint network (representing states of different activities) and the inverse direction is defined by the inverse constraint. For example, the dependency overlaps (o) has the inverse "overlapped by" (o^{-1})
(4) For all other nodes that do not have constraints between them defined, we define that all constraints are possible (precedes or preceded by or meets or met by or overlaps or overlapped by or finishes or finished by or contains or during or starts or started by or equals).

We illustrate an example for this translation in Figure 4. On the left side, we illustrate the model that needs to be translated into a temporal constraint network. On the right side, we describe the resulting temporal constraint network after applying these rules. If the model is changed then the corresponding temporal constraint network can be updated using these rules. It is not necessary to translate the model from scratch into a temporal constraint network again.

4.2.2. Detect Incorrect Temporal Constraint Networks

After obtaining a temporal constraint network from a model, we want to use the path consistency algorithm proposed by Allen (1983) to verify that the temporal constraint network is correct. There are two important aspects to consider when making this choice: performance and completeness.

Firstly, a good performance is required when verifying correctness of a model. The situation is dynamic and dependencies as well as activities are added and removed ad-hoc as needed. Basically, there are two different possibilities to perform verification of a temporal constraint network. The first possibility is using algorithms enforcing local consistency. The path consistency algorithm is one example for an algorithm enforcing local consistency. The second possibility is using search algorithms, such as back-tracking (see Kumar (1992)). Both possibilities return an empty solution for the constraint network, if it is incorrect. The path consistency algorithm has a much better worst case complexity (polynomial complexity) than the back-tracking algorithm (exponential complexity). Thus, it seems that the path consistency algorithm should be used.

However, another important aspect is completeness of the verification algorithm. Given a temporal constraint network translated from a model, we want to be sure that the algorithm is able to detect all incorrect and correct temporal constraint networks correctly. It turns out that the path consistency algorithm is not able to do this for the general case, i.e. all possible combinations of temporal constraints in a temporal constraint network. Allen gave one example for an incorrect temporal constraint network, which is classified by the path consistency algorithm as correct (cf. Allen (1983)). However, the path consistency algorithm does not classify wrongly correct temporal constraint networks as incorrect.

Nevertheless, researchers discovered 18 maximal tractable sub-algebras of Allen's interval algebra (cf. Krokhin *et al.* (2003)). If only combinations of temporal constraints from these algebras are used then it is possible to use the path consistency algorithm or other algorithms with similar complexity for detecting correctly incorrect and correct constraint networks. Maximal tractable means that each of the sub-algebras cannot be extended by further combinations of temporal interval relationships without becoming intractable, i.e. checking satisfiability can have exponential complexity in the worst case. However, there is only one maximal tractable sub-algebra containing all basic relationships as it is required by our approach: The ORD-Horn-Class (see Nebel and Bürckert (1995)). This class contains also further combinations of basic relationships, so that our approach can be extended by dependencies covering some combinations of basic relationships. Finally, this means we can leverage the path consistency algorithm to verify a temporal constraint network translated from a model of activities and temporal dependencies.

4.2.3. Conclusion

We explained in this section how a user can ensure correctness of a model consisting of activities and temporal dependencies. This has to be done automatically to reduce the burden of the user. However, an adequate performance for verifying correctness is required, because the situation is dynamic and the model may be adapted very often. We described a two step-procedure for detecting incorrect models by relying on well-understood formalisms. We showed then that this procedure has (1) sufficient performance and (2) that it is complete with respect to our model.

4.3. *Detecting Deviations from the Model and Activity Execution*

In a dynamic situation, such as in the disaster response, there is usually not always time to directly capture the situation in the system. For instance, goals of organizations may shift. This leads to a reassessment of activities. This has to consider their dependencies. More particularly, what has already been done and what is currently going on. However, we cannot expect that a user can update the model in time to reflect the necessary changes. This means there are deviations from the model and how activities are executed. These deviations should be highlighted to the user, so that they can be taken into account later. This is contrary to the process-based approaches presented in section three, because they enforce dependencies. Thus, they prohibit any deviations, which is not a realistic assumption in a dynamic situation.

We describe in the next paragraph how state changes of activities can be tracked. This is important to detect two different types of deviations in the subsequent paragraphs: violation of dependencies and unsynchronized dependencies.

4.3.1. Tracking State Changes

A state change is initiated by the user. The state change is defined as a set of one or more different activities entering simultaneously from their current state a new state. This is needed, for example, to start two or more activities at the same time. We define this formally as:

Definition 4.5 A state change is defined as $SC = \{s_1, s_i, .., s_n\}, i = 1, .., n$ where

- $s_i = (a_i, a_i.cs, ns)$ where
  - a_i is a activity as it has been described before.
  - $a_i.cs$ is the current state of the activity
  - $ns \in a_i.cat.S$ is the new state of the activity, whereby it holds that there is a transition from the current state to the new state $a_i.cs \rightarrow ns \in a_i.cat.f$
- $a_i \neq a_j \forall i, j = 1, .., ni \neq j$ Each element s_i of the state change has to contain a different activities

A state change must not contain the same activity several times, because this would mean that it can change into different states at the same time.

4.3.2. Dependency Violation

State changes of activities can lead to violation of dependencies. This basically means that the order of state changes is different from what is defined in the model in form of activities and dependencies. These violated dependencies should be highlighted to the users, so that they can be taken into account by them.

We propose to represent temporal dependencies as finite state machines. These finite state machines have state changes of activities as input and depending on the input they transit into a state "Neutral" or "Violated". Figure 5 illustrates an example for a finite state machine of the temporal dependency "overlaps".

We call a finite state machine, which represents a temporal dependency, dependency state machine (DSM). It is formally defined in Definition 4.6.

Definition 4.6 We define a dependency state machine as $DSM = (\Upsilon, \Omega, s, e, tg)$ where

- Υ is the finite set of input symbols
- Ω is the finite set of states
- $s \in \Omega$ is the the current state. On creation it describes the start state.
- $e \in \Omega$ is the end states or acceptance state

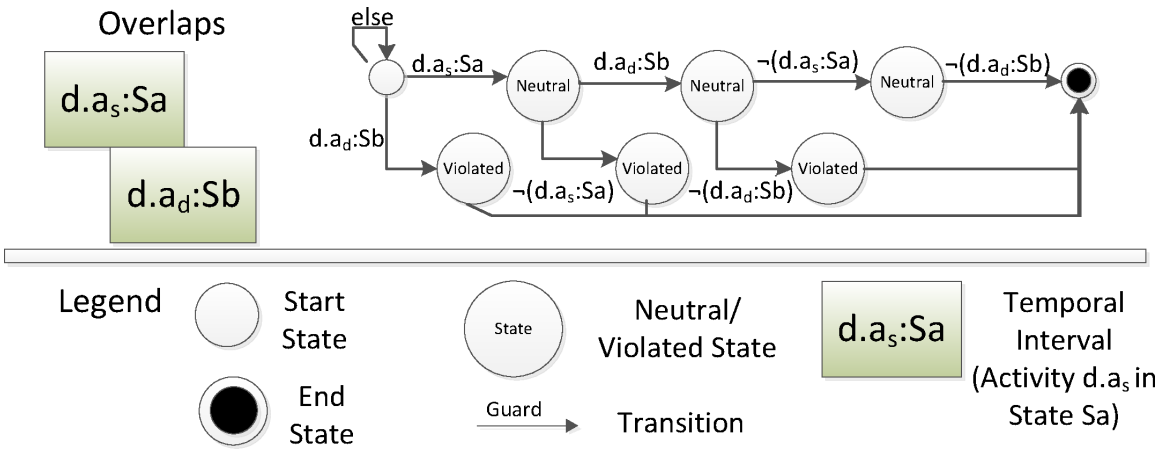

Figure 5. Example of a state machine representing the dependency "overlaps"

- $tg : \Omega \times \Upsilon \rightarrow \Omega$ is the transition function that defines given an input to which state the state machine should transit

The transition function tg of a DSM of a temporal dependency d can be described using the following constructs:

- $d.a_s : Sa$ source activity of the dependency $d.a_s$ changes into the state $Sa \in d.a_s.cat$
- $\neg(d.a_s : Sa)$ source activity of the dependency $d.a_s$ changes into any other successor state of $Sa \in d.a_s.cat$
- $d.a_d : Sb$ destination activity of the dependency $d.a_d$ changes into the state $Sb \in d.a_d.cat$
- $\neg(d.a_d : Sb)$ destination activity of the dependency $d.a_d$ changes into any other successor state of $Sb \in d.a_d.cat$
- $d.a_s : Sa \wedge d.a_d : Sb$ source activity of the dependency $d.a_s$ changes into the state $Sa \in d.a_s.cat$ and destination activity of dependency $a.a_d$ changes into the state $Sb \in d.a_d.cat$
- $\neg(d.a_s : Sa \wedge d.a_d : Sb)$ source activity of the dependency $d.a_s$ changes into any successor state of $Sa \in d.a_s.cat$ and destination activity of dependency $a.a_d$ changes into any successor state of $Sb \in d.a_d.cat$
- $\neg(d.a_s : Sa) \wedge d.a_d : Sb$ source activity of the dependency $d.a_s$ changes into any successor state of $Sa \in d.a_s.cat$ and destination activity of dependency $a.a_d$ changes into state $Sb \in d.a_d.cat$
- $d.a_s : Sa \wedge \neg(d.a_d : Sb)$ source activity of the dependency $d.a_s$ changes into state $Sa \in d.a_s.cat$ and destination activity of dependency $a.a_d$ changes into any successor state of $Sb \in d.a_d.cat$
- *else* any other state change of source activity of the dependency $d.a_s$ or destination activity of dependency $a.a_d$

We present in Listing 1 an algorithm how these DSMs can be used to highlight violated dependencies to the user when a state change occurred.

Listing 1 Detect violation of dependencies when executing activities

```
public V getViolatedDependenciesByStateChange(StateChange SC) {
dependencylist = GetAllDependencies(SC);
for {i=0;i<dependencylist.size-1;i++} {
        CheckDependency(dependencylist[i],SC);
```

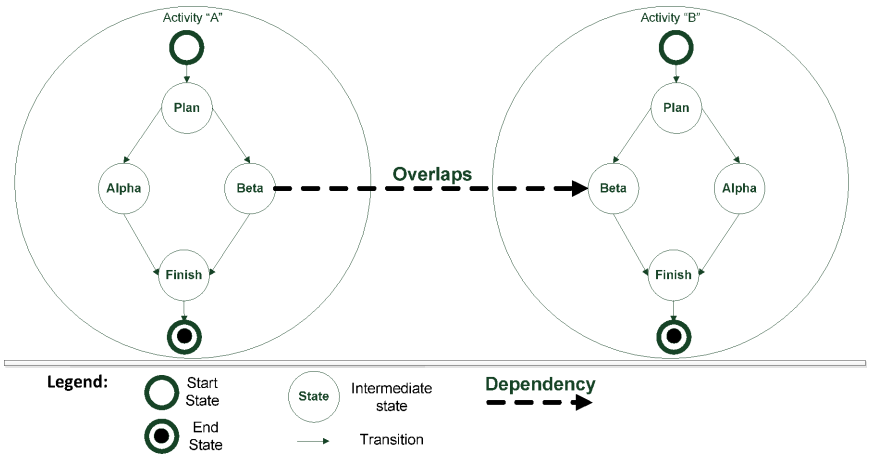

Figure 6. Example for a model that may lead to an unsynchronized dependency

```
        if (GetState(dependencylist[i]==violated) {
                V.add(dependencylist[i]);
        }
}
return V; // list of violated dependencies
}
```

The algorithm gets all dependencies related to the activities occurring in the state change ($GetAllDependencies(SC)$). It then inputs the state change into the finite state machine representation of the dependencies ($CheckDependency(dependencylist[i], SC)$). This results either in "Neutral" or "Violated" state of a dependency. All dependencies in state "Violated" are returned.

4.3.3. Unsynchronized Dependencies

Sometimes it is not possible to detect if a dependency is violated or not. This implies that there needs to be another method to detect deviations. An example for this problem is illustrated in Figure 6. There are two activities "A" and "B" with the same lifecycle. A dependency is established between the state "Beta" of activity "A" and the state "Beta" of activity "'B". If activity "A" changes into state "Beta" and activity "B" changes into state "Alpha" then it is not possible to decide if the dependency is violated or not, because the state "Beta" of activity "B" cannot be reached anymore. Thus the dependency state machine can never reach its end state. We call this an unsynchronized dependency.

The idea of an algorithm for detecting unsynchronized dependencies is as follows. Given a state change of activity "a" and the dependencies related to this activity, find a set of state changes of activity "a", so that the DSMs of these dependencies can reach their end state. In Listing 2, we describe an algorithm for detecting unsynchronized dependencies when performing a state change. Each state change can lead to unsynchronized dependencies connected to the activities changing their state. The algorithm performs for each dependency associated with an activity changing the state the following check:

- Check for the source activity $d.a_s$ of a dependency d if from the current state $d.a_s.cs$ there are only paths through the $d.s_s$ source state of the activity of the dependency. If there are only paths involving $d.s_s$ and there is no paths of the destination activity $d.a_d$ of the dependency d through the destination state $d.s_d$ of the dependency d then the dependency d is unsynchronized.

- Check for the destination activity $d.a_d$ of a dependency d if from the current state $d.a_d.cs$ there are only paths through the $d.s_d$ destination state of the dependency. If there are only paths involving $d.s_d$ and there is no paths of the destination activity $d.a_s$ of the dependency d through the destination state $d.s_s$ of the dependency d then the dependency d is unsynchronized.

The function "GetNumPaths" describes how the number of alternative paths from one activity state to another activity state can be determined. It is based on depth-search (cf. Franke (2011)). It is required that the activity type does not contain strongly connected components, which we excluded by definition.

Listing 2 Detect Unsynchronized Dependencies when Executing Activities

```
public U getUnsynchronizedDependenciesByStateChange(StateChange SC) {
dependencylist = GetAllDependencies(SC);
for (int i=0;i<dependencylist.size-1;i++) {
 Dependency d=dependencylist[i];
 numPathsToDependencySourceActivityState=GetNumPaths(d.as.cs,d.ss);
 numPathsToDependencySourceActivityEndState=GetNumPaths(d.as.cs,d.as.cat.se);
 numPathsToDependencyDestinationActivityState=GetNumPaths(d.ad.cs,d.sd);
 numPathsToDestinationActivityEndState=GetNumPaths(d.ad.cs,d.ad.cat-se);
 // Check if dependency is reachable from the current state
 // of the source activity
 if (numPathsToDependencySourceActivityState>0) {
 if (numPathsToDependencySourceActivityState==
 numPathstoSourceActivityEndState) {
  // Reachable from current state of source activity of dependency
  if (numPathsToDependencyDestinationActivityState ==0) {
   // Not reachable from current state of destination activity of dependency
   U.add(d);
  }
 }
 }
 if (numPathsToDependencyDestinationActivityState>0) {
 if (numPathsToDependencyDestinationActivityState==
 numPathsToDestinationActivityEndState) {
 // Reachable from current state of destination activity of dependency
  if (numPathsToDependencySourceActivityState==0) {
   U.add(d);
  }
 }
 }
}
return U; // List of unsynchronized dependencies
}
```

Raposo *et al.* (2000) proposes an alternative approach involving temporal interval relationships using a petri net formalism. Their approach is only able to enforce temporal dependencies and unsynchronized dependencies are handled using timeouts. This is contrary to the idea of qualitative temporal dependencies. Furthermore, the aforementioned approach may detect wrongly a dependency as unsynchronized if a state change takes longer than the timeout permits. Finally, their approach can only validate petri net

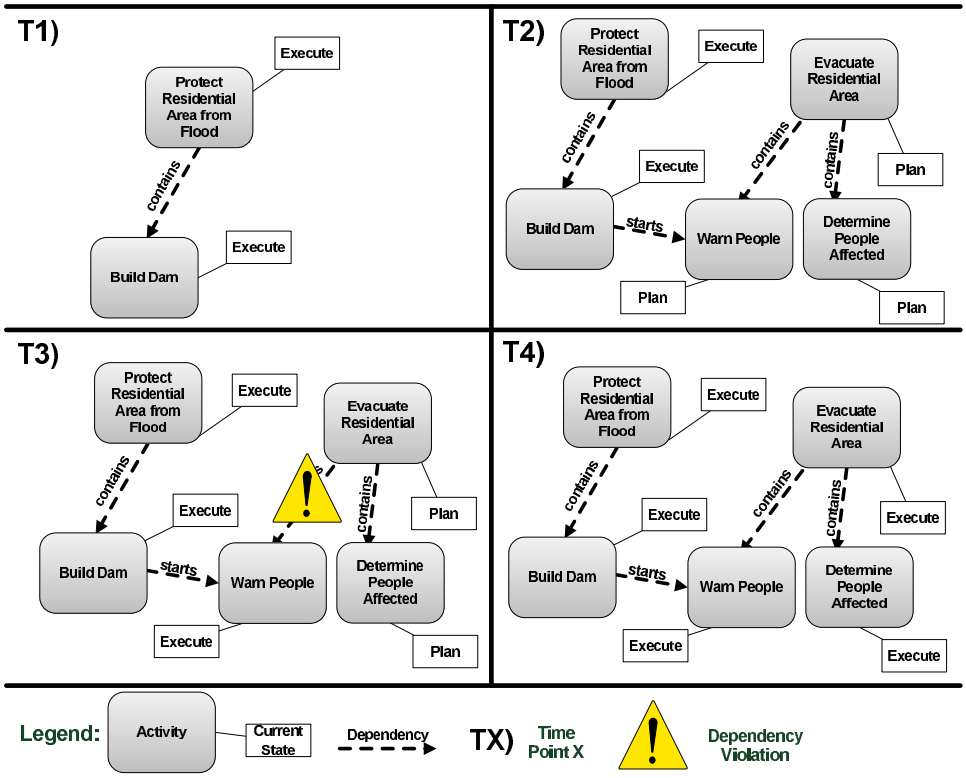

Figure 7. Example for the evolution of a model during a disaster response in four steps

properties, but not temporal ones. This makes it also very difficult to introduce dynamic changes into the model (cf. Ellis *et al.* (1995)).

4.3.4. Conclusion

We presented in this subsection how deviations from the model and how activities have been executed can be detected. Activities can change their state and these state changes may deviate from what is defined in the model of activities as well as dependencies. We described two different types of deviations: violated and unsynchronized dependencies. We explained different algorithms for detecting these deviations, so that they can be highlighted to the user. This is important for the user, so that shifting goals leading to a reassessment of activities and dependencies can be considered later, when there is time and information available to adapt the model to the situation. Since these algorithms only work with a very small subset of the activities and dependencies, there are no performance issues. Furthermore, these algorithms have approximately linear complexity.

4.4. *Example*

We present in this subsection an example of the framework using the scenario described in section two. Figure 7 illustrates the evolution of a model in this scenario in four steps. In the first step, the user models the activities “Protect Residential Area from Flood” and “Build Dam”. The user changes them into state “Plan”. There are no violated or unsynchronized dependencies, because there are no dependencies modeled. The user creates a dependency “contains” between the state “Execute” of the activities. This means that as long as “Protect Residential Area from Flood” is executed, the activity

"Build Dam" can be executed. After adding the dependency, the model is verified. The model is error-free. The user then changes the activity "Protect Residential Area from Flood" into state "Execute". There are no violated or unsynchronized dependencies.

In the second step, the user models further activities: "Warn People", "Evacuation of Residential Area" and "Determine People Affected". The user changes them into state "Plan". There are no violated or unsynchronized dependencies. The user establishes a dependency "contains" between the state "Execute" of the activity "Evacuation of Residential Area" and the state "Execute" of activity "Warn People". The same type of dependency is established between the state "Execute" of the activity "Evacuation of Residential Area" and the state "Execute" of activity "Determine People Affected". Another dependency "starts" is established between the activity "Build Dam" in state "Fail" and activity "Warn People" in state "Execute". This means when building a dam fails then the people are warned that the area is going to be executed. The addition of these dependencies does not introduce errors in the model.

In the third step, the user changes the activity "Build Dam" into state "Fail" and the activity "Warn People" into "Execute". The activity "Protect Residential Area" is also changed into state "Fail". All state changes do not lead to unsynchronized dependencies. However, the dependency "contains" between state "Execute" of the activity "Evacuation of Residential Area" and the state "Execute" of "Warn People" is violated, because the activity "Evacuation of Residential Area" is still in state "Plan". This is highlighted to the user.

In the fourth step, the user decides to resolve this issue by changing the activity "Evacuation of Residential Area" into state "Execute" and afterwards changing activity "Determine People Affected" into state "Execute". The violated dependency is marked by the user as resolved and is not highlighted anymore.

4.5. *Summary*

We presented in this section a framework for temporal coordination of activities in dynamic situations. This framework enables (1) modeling of activities and temporal dependencies, (2) verification of a model of activities and temporal dependencies and (3) detecting deviations between the model and how activities have been executed. We go beyond process-based or artifact-based approaches by providing a richer set of temporal dependencies besides sequential ones. Furthermore, we do not require full specification of all temporal dependencies. This is usually required by formalisms based on petri nets, which are commonly used for verifying business processes. These formalisms have the disadvantage that they do not work very well with dynamically changing processes (cf. Ellis *et al.* (1995)). In fact, the user may also decide to not model temporal dependencies at all or only between selected activities when using our framework. Contrary to existing rule-based approaches, we chose temporal dependencies which are easy to visualize and have similarities with dependencies used in project management. Thus, we assume that the user has a better understanding what has been done, what is currently going on and what can be done next. Additionally, we demonstrated how the user can verify the model with acceptable performance, which is not always the case for rule-based approaches. However, good performance is required for coordinating activities in dynamic situations.

5. Implementation

We provide in this section an overview of the implementation of the concepts described in the previous sections. We decided to implement them as an extension to the collaboration service "Google Wave" (cf. Ferrate (2009)). This had several reasons. Firstly, it is in process of being open sourced as "Apache Wave" (see Apache (2011)). It is based on open standards and other software can inter-operate with it. Secondly, we wanted to leverage an existing platform to demonstrate how our concepts unfold in the context of different tools already used in disaster management, such as maps, communication tools, pictures or videos. Thirdly, it provides an infrastructure for extending our framework to the distributed inter-organizational level (cf. Franke (2011)). Fourthly, we used the prototype to experiment it with students (see Franke (2011)). Of course, the concepts could also be implemented in other software, such as social networking services.

5.1. *Preliminaries*

The collaboration service "Google Wave" is an infrastructure that supports "real-time" collaborative editing of distributed documents, called "Waves". The Open Wave Federation Protocol allows different Wave servers to communicate with each other. This is mainly used to replicate "Waves" over different servers. This means each organization involved in a disaster response can host its own server and decides what they share with whom. Each "Wave" has participants of the same or different servers. New participants can be added to a "Wave". "Google Wave" can be extended in two different ways: "Gadget" and "Robot".

A "Gadget" can be inserted into a "Wave" and supports further collaboration functionality. For instance, a map can be inserted into a "Wave" and different participants can add landmarks in real-time. A "Gadget" stores its own data (e.g. landmarks) in the "Wave" where it has been inserted. "Gadgets" are implemented in the Hypertext Markup Language (HTML) and Javascript.

A "robot" can be compared to an automated participant of a "Wave". It can be hosted on any server and implemented in any programming language. It can react on events, such as changes, in a "Wave" and react to them. It is used to connect a "Wave" with the outer world or other "Waves". For example, it can insert live stock market quotes in a "Wave".

5.2. *Our Extension*

Our concepts are implemented using both extension mechanisms of "Google Wave". Firstly, we provide a "Gadget" that supports collaborative modeling of activities and their temporal dependencies. Furthermore, it allows users to change the state of activities. At the moment, activities and dependencies can be visualized in two different ways. On the one hand, they can be visualized in a graphical notation. On the other hand, they can be visualized as a table. The first possibility provides a better overview what has been done, what is currently going on and what are the next steps. The second possibility seems to be more suitable if a user needs to get a quick overview over all activities or wants to enter a lot of new activities rapidly.

Secondly, our "Robot" reacts on changes entered by the user in the "Gadget". For instance, every time a dependency is added or removed, it performs verification of the model described in the "Gadget" and returns the result to the "Gadget", which displays

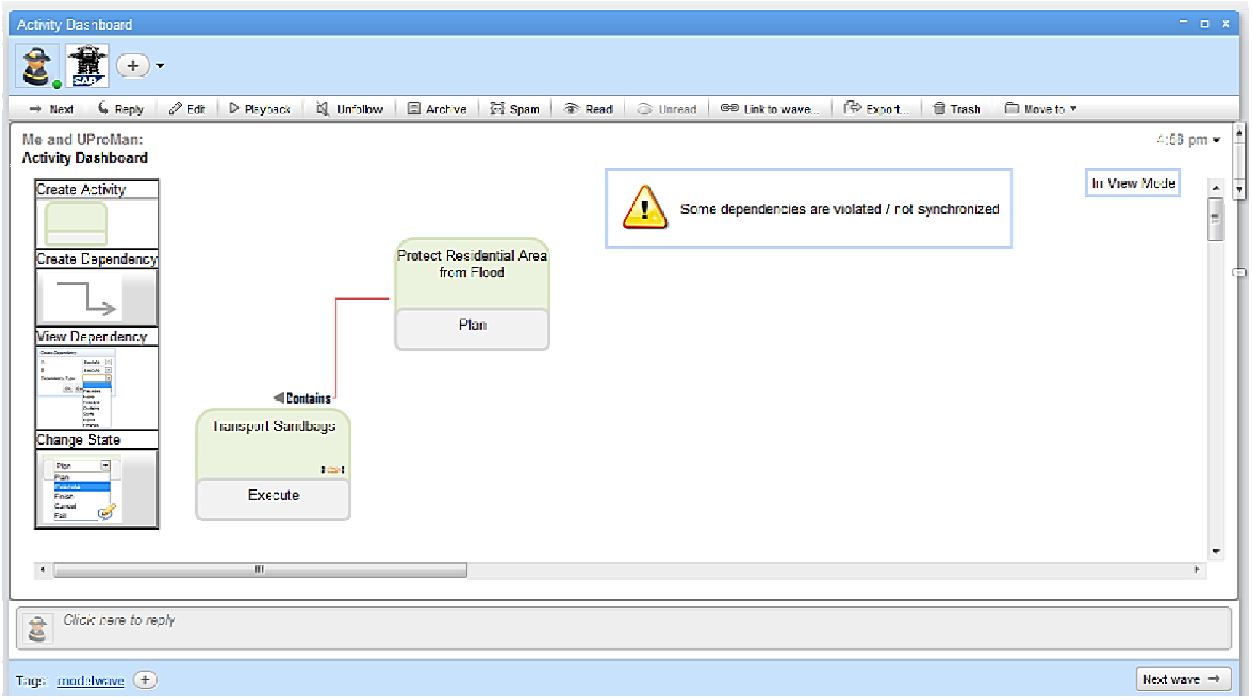

Figure 8. Screenshot of our prototype

it to the user. If a state change violates dependencies or leads to unsynchronized dependencies then the robot communicates this to the "Gadget" as well. These dependencies are then highlighted to the user. We provide further extensions out of the scope of this paper, such as replicating selected activities in different models in different "Waves" of different organizations (cf. Franke (2011).

In Figure 8, we illustrate a screenshot of a "Wave" containing our "Gadget". It currently displays a model of activities and temporal dependencies using the graphical notation. It is the view of the fire chief, who modeled two activities: "Protect Residential Area from Flood" and "Transport Sandbags". He established a dependency "contains" between them. The activity "Protect Residential Area from Flood" is currently in state "Plan". The activity "Transport Sandbags" has been set to the state "Execute". This violated the dependency between these two activities. This is highlighted to the user and the dependency is shown in red.

6. Comments

The implementation showed that our concepts are technically realizable. However, in order to assess future perspectives with respect to disaster management and more generally coordination, we need to evaluate them empirically. As a first step, we presented the concepts to four experienced disaster managers. Three of them were fire fighter commanders in the US and Europe. Another one was commander of a large humanitarian aid organization. We now describe selected comments by them.

The fire fighter commander from Southern California confirms the problem of unstructured coordination that we have identified in section two: *"[..] the pile of messages in the inbox, [which contains] the reality as a situation [..] being able to put them in context and update them and coordinate them to create a common picture is the difficulty."*

Furthermore, he highlights another problem with current tools: *"We have an incident action plan, which each entity utilizes and keeps track of their activities. [This happens] more or less on a manual basis and [people enter] who is command and what actions are taken. [..] We update it every twelve hours and put [..] future actions and intended actions for the next twelve hours operational period. [..] A lot of this in larger incidents has gone to a web-based electronic data exchange. [This is a] software program, that does not alert you to inter-connection failures [and] that does not tie pieces of information together [..]. My experience with it [is] that it is not more than a collection of emails back and force. [They are] not in chronological order and [the software] does not necessarily go back and correct. [For example a message says] it is action A [..] and in 30 minutes you change your mind now its B, because there is another email [saying] this is B. [..] Later on you may say it [is] C, and A is still in the system. [..] If somebody looks at [the messages] as a sequence they going to think it is A, when it is really C. [..] This [should be] more of a system that can change [..] that has trigger points in it [..] and be more of an open actual working plan in real-time. [It should not be] a collection of information that is gone back and forth."*

The fire fighter commander from Washington confirms the problem of shifting goals (end states): *"You are reacting based on an end state, an ultimate objective, an ultimate goal or an ultimate concept of what this incident will look like when it is over. [..] The whole [response] will be tailored to mediating this end state, as opposed just to continuous analysis/reaction/analysis/reaction."*

The fire fighter commander from Southern California maps our approach to existing concepts: *"We use time lines as markers for future action [and] we have what we call trigger points. [This means] when the incident advances to a certain point, it triggers other things. [..] That would fit into your model as well, using time lines, connecting inter-dependencies, and [defining] future actions."*

The fire fighter commander from Washington highlights the ability of the system to measure progress, but also coordinate the activities towards a goal and that this goal (end state) can shift:

"there is a couple things [about your approach], [..] it is a good way to measure progress, and the second thing is that you recognize that the end state will change, because of the dynamic situation of the incident you are involved in [..] the end state may need to be modified or you might have intermediate type of objectives."

Franke (2011) describes how experiments with students can be used to gain more generic insights on coordination. These experiments are complementary to evaluations, for example, in disaster exercises, where a lot of complex factors play a role and it is more difficult to assess if our concepts can make a difference with respect to coordination. Nevertheless, we think that our concepts are important steps towards a more complete solution for coordination of activities in dynamic situations.

7. Conclusion

We described in this paper a framework for coordination of activities in dynamic situations. The underlying idea of the framework is that users can model evolving processes by describing activities and temporal dependencies between them in a structured manner. Contrary to existing approaches, we do not require full specification of all dependencies. We support a richer set of temporal dependencies than process-based approaches explained in section three. Contrary to rule-based approaches, there is a natural visual-

ization of these temporal dependencies. Thus, the user can easier understand what has been done, what is currently going on and what can be done next. Furthermore, the framework is able to verify the model for correctness with acceptable performance. This means that the user can always ensure that the model is correct. Finally, the framework can highlight deviations from the model and how activities have been executed. These deviations are caused by shifting goals leading to a reassessment of activities and their relations. Contrary to existing approaches, we do not enforce these dependencies, because this would require that the user can always adapt the model in time, which cannot be assumed in dynamic situations.

Although dynamic situations are ad-hoc and unstructured, we argued that a structured approach for coordination is more beneficial than an unstructured one. This has also been confirmed by the state of the art as well as comments by experienced disaster managers about our approach. As future work, it would be interesting to investigate the relevance of other types of dependencies and how they can be used for coordination in dynamic situations in a structured way. For instance, resource dependencies or spatial dependencies can be of importance for evolving processes. At the moment, our implementation only allows describing them in an unstructured manner in a collaborative document by mentioning, for example, the amounts of sandbags needed or by inserting a map into this collaborative document. Furthermore, another important research direction is how our framework can be extended to the distributed inter-organizational level. At this level, selected activities are shared between different organizations and replicated in the activity workspaces of different organizations. Temporal dependencies can be established between internal and replicated activities. This can lead to diverging views on activities and temporal dependencies. Mechanisms need to be introduced to ensure a converging view. We already discussed further directions for empirical evaluation of our concepts. We believe that our concepts are highly relevant for the proposed case management process modeling standard by the Object Management Group (OMG) (see OMG (2009)), because dynamically evolving processes are a core part of it.

Acknowledgements

References

Adams, M., 2007. Facilitating Dynamic Flexibility and Exception Handling for Workflows. Thesis (PhD). Queensland University of Technology.

Allen, J.F., 1983. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26 (11), 832–843.

Apache, 2011. Incubator - Apache Wave, `http://incubator.apache.org/wave/`. [online] [18.04.2011].

Becker, Jörg Kugeler, M. and Rosemann, M., eds. , 2010. *Process Management: A Guide for the Design of Business Processes*. Springer.

Bhattacharya, K., *et al.*, 2007a. Artifact-centered Operational Modeling: Lessons from Customer Engagements. *IBM Systems Journal*, 46 (4), 703–721.

Bhattacharya, K., *et al.*, 2007b. Towards Formal Analysis of Artifact-Centric Business Process Models. *In*: *5th International Conference on Business Process Management*, Brisbane, Australia.

Bowers, J., Button, G., and Sharrock, W., 1995. Workflow from Within and Without: Technology and Cooperative Work on the Print Industry Shopfloor. *In*: *Fourth European Conference on Computer-Supported Cooperative Work*, Stockholm, Sweden.

Cugola, G., 1998. Tolerating Deviations in Process Support Systems via Flexible Enactment of Process Models. *IEEE Transactions on Software Engineering*, 24 (11), 982–1001.

Dadam, P. and Reichert, M., 2009. The ADEPT Project: A Decade of Research and Development for Robust and Flexible Process Support. *Computer Science - Research and Development*, 23 (2), 81–97.

Dayal, U., Hsu, M., and Ladin, R., 1990. Organizing Long-Running Activities with Triggers and Transactions. *In*: *ACM SIGMOD Conference on Management of Data*, Atlantic City, New Jersey, United States.

de Leoni, M., Mecella, M., and De Giacomo, G., 2007. Highly Dynamic Adaptation in Process Management Systems Through Execution Monitoring. *In*: *5th International Conference on Business Process Management (BPM'2007)*, Brisbane, Australia.

de Man, H., 2009. Case Management: Cordys Approach. *BPTrends*, February, 1–13.

Dourish, P., *et al.*, 1996. Freeflow: Mediating between Representation and Action in Workflow Systems. *In*: *ACM Conference on Computer Supported Cooperative Work (CSCW)*, Boston, Massachusetts, USA.

Döweling, S., *et al.*, 2009. Soknos - An Interactive Visual Emergency Management Framework. *In*: *Workshop on Geographical Information Processing and Visual Analytics for Environmental Security*, Trento, Italy.

Dumas, M., van der Aalst, W.M.P., and ter Hofstede, A.H., 2005. Introduction. *In*: *Process-Aware Information Systems.*, 3–21 Wiley.

Ellis, C., Keddara, K., and Rozenberg, G., 1995. Dynamic Change Within Workflow Systems. *In*: *Conference on Organizational Computing Systems*, Milpitas, California, USA.

Fahland, D. and Woith, H., 2009. Towards Process Models for Disaster Response. *In*: *Process Management for Highly Dynamic and Pervasive Scenarios*, Milano, Italy.

Fahland, D., *et al.*, 2009. Instantaneous Soundness Checking of Industrial Business Process Models. *In*: *7th International Conference on Business Process Management*, Ulm, Germany.

Ferrate, A., 2009. *Getting Started with Google Wave*. O'Reilly Media, Inc.

Franke, J., 2011. Coordination of Distributed Activities in Dynamic Situations - The Case of Inter-organizational Crisis Management. Thesis (PhD). École Doctorale IAEM Lorraine, Université Henri Poincaré/Université de Lorraine, Nancy, France.

Franke, J., Charoy, F., and El Khoury, P., 2010. Collaborative Coordination of Activities with Temporal Dependencies. *In*: *18th International Conference on Cooperative Information Systems (COOPIS'2010) / OnTheMove (OTM) Conferences*, Heraklion, Crete, Greece.

Franke, J., *et al.*, 2011. Reference Process Models and Systems for Inter-Organizational Ad-Hoc Coordination - Supply Chain Management in Humanitarian Operations. *In*: *8th International Conference on Information Systems for Crisis Response and Management (ISCRAM'2011)*, Lisbon, Portugal.

Gaaloul, K., *et al.*, 2010. Dynamic Authorisation Policies for Event-based Task Delegation. *In*: *22nd International Conference on Advanced Information Systems Engineering*, Hammamet, Tunisia.

Georgakopoulos, D., *et al.*, 2000. Managing Escalation of Collaboration Processes in Crisis Mitigation Situations. *In*: *16th International Conference on Data Engineering*, San Diego, California, USA.

Gerede, G.E. and Su, J., 2007. Specification and Verification of Artifact Behaviors in Business Process Models. *In*: *5th international Conference on Service-Oriented Com-*

puting (ICSOC'2007), Vienna, Austria.

Grigori, D., Charoy, F., and Godart, C., 2001. Anticipation to Enhance Flexibility of Worflow Execution. *In*: *Database and Expert Systems Applications (DEXA'2001)*, Munich, Germany.

Grinter, R.E., 2000. Workflow Systems: Occasions for Success and Failure. *Computer Supported Cooperative Work*, 9 (2), 189–214.

Huth, C., Erdmann, I., and Nastansky, L., 2001. 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*, Maui, Hawaii, USA.

Klein, G., 1999. *Sources of Power: How People Make Decisions*. MIT Press.

Krokhin, A., Jeavons, P., and Jonsson, P., 2003. Reasoning about temporal relations: the maximal tractable subalgebras of Allen's interval algebra. *Journal of the ACM*, 50 (5), 591–640.

Kumar, V., 1992. Algorithms for Constraint Satisfaction Problems: A Survey. *AI Magazine*, 13 (1), 32–44.

Landgren, J. and Nulden, U., 2007. A Study of Emergency Response Work: Patterns of Mobile Phone Interaction. *In*: *SIGCHI Conference on Human Factors in Computing Systems*, San Jose, CA, USA.

Lutz, C., Wolter, F., and Zakharyashev, M., 2008. Temporal Description Logics: A Survey. *In*: *15th International Symposium on Temporal Representation and Reasoning*, Montreal, Quebec, Canada.

Mak, H.Y., *et al.*, 1999. Building Online Crisis Management Support Using Workflow Systems. *Decision Support Systems*, 25 (3), 209–224.

Medina-Mora, R., *et al.*, 1992. The Action Workflow Approach to Workflow Management Technology. *In*: *ACM Conference on Computer-supported Cooperative Work (CSCW)*, Toronto, Canada Toronto, Canada Toronto, Canada.

Müller, D., Reichert, M., and Herbst, J., 2007. Data-driven Modeling and Coordination of Large Process Structures. *In*: *15th International Conference on Cooperative Information Systems (CoopIS)*, Vilamoura, Algarve, Portugal.

Müller, R., Greiner, U., and Rahm, E., 2004. AgentWork: A Workflow System Supporting Rule-Based Workflow Adaptation. *Data & Knowledge Engineering*, 51, 223–256.

Nebel, B. and Bürckert, H.J., 1995. Reasoning about Temporal Relations: A Maximal Tractable Subclass of Allen's Interval Algebra. *Journal of the ACM*, 42 (1), 43–66.

Olding, E. and Rozwell, C., Expand Your BPM Horizons by Exploring Unstructured Processes. , 2009. , Technical report G00172387, Gartner.

OMG, 2009. Case Management Process Modeling (CMPM) - Request for Proposal, OMG Document: Bmi/2009-09-23, `http://www.omg.org/cgi-bin/doc?bmi/2009-09-23`. [online] [29.09.2011].

Pesic, M., 2008. Constraint-Based Workflow Management Systems: Shifting Control to Users. Thesis (PhD). Technische Universiteit Eindhoven.

Rahaman, M.A., Roudier, Y., and Schaad, A., 2009. Document-based Dynamic Workflows: Towards flexible and Stateful Services. *In*: *IEEE International Conference on Services Computing*, Bangalore, India.

Raposo, A.B., Magalhaes, L.P., and Ricarte, I.L., 2000. Petri Nets Based Coordination Mechanisms for Multi-Workflow Environments. *International Journal of Computer Systems Science & Engineering*, 15 (5), 315–326.

Reijers, H.A., *et al.*, 2007. Workflow Management Systems + Swarm Intelligence = Dynamic Task Assignment for Emergency Management Applications. *In*: *5th Interna-*

tional Conference on Business Process Management, Brisbane, Australia.

Rüppel, U. and Wagenknecht, A., 2007. Improving Emergency Management by Formal Dynamic Process-modelling. *In*: *24th Conference on Information Technology in Construction*, Maribor, Slovenia.

Sadiq, S., Sadiq, W., and Orlowska, M., 2001. Pockets of Flexibility in Workflow Specification. *In*: *20th International Conference on Conceptual Modeling*, Yokohama, Japan.

Skaf, H., Charoy, F., and Godart, C., 1996. Maintaining Consistency of Cooperative Software Development Activities. *In*: *6th International Workshop on Foundations of Models and Languages for Data and Objects*, Schloss Dagstuhl, Germany.

Tarjan, R.E., 1972. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1 (2), 146–160.

Unknown, The Federal Response to Hurricane Katrina - Lessons Learned. , 2006. , Technical report, The White House.

van der Aalst, W., ter Hofstede, A., and Weske, M., 2003. Business Process Management: A Survey. *In*: *1st International Conference on Business Process Management*, Eindhoven, The Netherlands.

van der Aalst, W., Weske, M., and Grünbauer, D., 2005. Case Handling: A new Paradigm for Business Process Support. *Data & Knowledge Engineering*, 53 (2), 129–162.

Vanderfeesten, I., Reijers, H.A., and van der Aalst, W., 2011. Product-Based Workflow Support. *Information Systems*, 36 (2), 517–535.

Wang, J. and Akhil, K., 2005. A Framework for Document-Driven Workflow Systems. *In*: *3rd International Conference on Business Process Management*, Nancy, France.

Weick, K., 2000. *Making Sense of the Organization*. Blackwell.

Zahoor, E., Perrin, O., and Godart, C., 2010. DISC-SeT: Handling Temporal and Security Aspects in the Web Services Composition. *In*: *8th IEEE European Conference on Web Services*, Ayia Napa, Cyprus.