PRIMA: A Model-Based Method for Analyzing Place-Related Information in Disaster Response Processes

Completed Research Paper

Stefan Sackmann
Chair of Information Management
Martin Luther University Halle-Wittenberg
stefan.sackmann@wiwi.uni-halle.de

Marlen Hofmann
Chair of Information Management
Martin Luther University Halle-Wittenberg
marlen.hofmann@wiwi.uni-halle.de

Hans Betke
Chair of Information Management
Martin Luther University Halle-Wittenberg
hans.betke@wiwi.uni-halle.de

ABSTRACT

Processes in disaster response management (DRM) and business processes are similar due to their general structure and goals. Thus, applying workflow management systems (WfMS) is discussed as a promising approach to manage disaster response processes (DRP). However, one main obstacle for realizing the potentials of WfMS in DRM is the lack of methods and tools addressing disaster-specific aspects that exceed the “classical” business context. A particular challenge is posed by the analysis of interdependencies resulting from stationary and mobile activities and resources. Therefore, in this contribution, a novel model-based method for analyzing place-related information is proposed and discussed. The PRIMA method aims at the identification of non-operable activities (and possible remedies) before the execution of an actual DRP stalls and is improvised. Applying the method promises a sound basis for both effective and efficient planning of DRP as well as their successful management by future disaster response WfMS.

Keywords

Disaster Response Management, Disaster Response Process, Place-Related Information, Place-Related Interdependencies, Analysis Method

MANAGING DISASTER RESPONSE PROCESSES WITH WORKFLOW MANAGEMENT SYSTEMS

Methods and tools from the domain of business process management (BPM) and workflow management (WfM) are considered as promising approaches to facilitate an improved disaster response management (DRM) and execution of disaster response processes (DRP) (e.g. Fahland and Woith, 2009; Georgakopoulos, Schuster, Baker und Cichocki, 2000; Hofmann, Sackmann and Betke, 2013b; Rueppel and Wagenknecht, 2007; Sell and Braun, 2009). The application of adaptive workflow management systems (WfMS) (e.g. Dadam, Reichert, Rinderle, Jurisch, Acker, Göser, Kreher and Lauer, 2007) has been proposed especially for the tactical echelon comprising coordination of various decentralized and parallel operated DRP (e.g., Chen, Sharman, Rao and Upadhyaya, 2008): so-called disaster response workflow management systems (DRWfMS) will facilitate the overall and systematic management of DRP by providing methods and tools for information management, communication, and, in particular, provision of process transparency (; Hofmann, Sackmann and Betke, 2013b, 2013a; Jansen, Lijnse and Plasmeijer, 2010; Sell and Braun, 2009, Ziebermayr, Huber, Kollarits and Ortner, 2011).

However, to the best of our knowledge, such systems have not yet been realized in practice. This is attributed to the differing context of BPM and DRM. Thus, the application of methods and tools cannot be transferred directly but needs a domain-specific adaptation. One main difference is apparently the unpredictability of disasters and also of constraints which might determine the operability of any pre-planned DRP. Thus, ongoing DRP are usually subject to a continuous adaptation during runtime that is not known to the same extent in BPM. Moreover, in non-trivial disaster response situations, process adaptation becomes a highly complex task that is usually not feasible to be managed manually “with the naked eye”. In our view, a future success of DRWfMS will be reliant on appropriate methods and tools providing an automated analysis of ongoing DRP and changing context data as well as an automated reasoning and calculation of necessary process adaptations.

The unpredictability of disaster sites and possibly threatened assets has been identified as a major problem for disaster response (e.g. Bassett, 2008; Fleischhauer, 2008) and flexible adaptation to place-related information is crucial to DRM. Since many place-related aspects (e.g. the actual resource situation on-site) can only be considered during the runtime of DRP and the actual place of process execution is usually unknown before the occurrence of a disaster, identifying and analyzing place-related inconsistencies and impossibilities has enormous potential to improve process transparency and the effectiveness of process execution. In this contribution, a novel method called PRIMA (**P**lace-**R**elated **I**nformation in Workflow **M**odels **A**nalysis) for a model-based analysis of DRP with regard to place-related information and resulting constraints is presented. This contribution is structured as follows: the next section briefly discusses the integration of place-related information into process modeling languages. The proposed model extensions are used in the third section as starting point for developing an algorithm analyzing place-related constraints and identifying inconsistencies in modeled DRP. The section concludes with an interpretation of the achieved results and the limitations of the PRIMA method. The contribution ends with a short conclusion, first ideas for further improvement, and an outlook on arising research desiderata.

INTEGRATING PLACE-RELATED INFORMATION INTO PROCESS MODELS

As discussed in the previous section, place-related context information is of major importance for DRM. In order to ensure an appropriate adaptation of ongoing DRP, such information has to be continuously analyzed and interpreted. For instance, spatial characteristics might necessitate a replacement of response activities specified in a predefined response plan (e.g. if access roads to an accident location are destroyed, flying ambulances will have to be sent instead of road ambulances). Furthermore, the disaster site could require a process adaptation with regard to additional logistic activities, e.g. to provide supplies on-site. It is also conceivable that further unforeseeable prevention measures have to be initialized if surrounding assets are in danger. Many more examples exist where “place” might also affect resource allocation, duration, priorities, and even the operability of disaster response activities.

Taking such situations into consideration and managing them effectively by future DRWfMS presupposes a methodical or rather model-based identification, analysis, and solving. This, again, requires a structured representation of place in DRP models. Since such elements are not yet known to current modeling languages in the field of BPM, an extension to allow the integration of place-related information in process models is primarily necessary. (Sackmann, Hofmann and Betke, 2013) discuss locally bound (stationary) and place-independent (mobile) resources/activities as basic place-related characteristics that include the precise location in the form of formal characteristics of the place element, e.g. as geographic coordinates by longitude and latitude. They also propose modeling elements that can be graphically represented as shown in Table 1. The integration of such elements into existing BPM languages is still the subject of further research.

Type	Description	Graphical representation
Stationary resource	stationary resources which can only be used at their site e.g. water hydrant	 Resource Longitude, Latitude
Mobile resources	mobile resources which can be used independent of site e.g. fire engine	 Resource Longitude, Latitude
Stationary activity	stationary activities which can only be executed at a certain place e.g. extinguish fire	 Activity Longitude, Latitude
Mobile activity	mobile activities which can only be executed at a certain place e.g. construct mobile hospital	 Activity Longitude, Latitude

Table 1. Modeling place-related information in DRP (adapted from Sackmann et al., 2013)

Based on this structured representation of place, a first categorization for possible issues that might result from the combination of place-dependent activities and resources becomes available (Sackmann et al., 2013). In this regard, three basic dependencies can be distinguished and combined: dependencies between …

(1) activities (activity-to-activity),
(2) an activity and a resource (activity-to-resource), and
(3) resources (resource-to-resource).

Combining these dependencies with the local characteristics (stationary or mobile) leads to 12 cases, which provide a basis for further analysis of potential conflicts that have to be analyzed in order to evaluate the operability of a DRP.

ANALYZING PLACE-RELATED INFORMATION AND DEPENDENCIES – A GRAPH THEORETICAL APPROACH

The general aim of the PRIMA approach is to identify conceivable conflicts between place-dependent activities and resources in DRP models for all the identified cases mentioned above. This requires methods and tools capable to identify all those activities within a given DRP model that cannot (or might not) be operable in the intended way due to incompatible places of activities and their required resources. Taking such conflicts into consideration and managing them effectively by future DRWfMS presupposes a methodical or rather model-based identification, analysis, and solving. However, since the cases are very different in their characteristics, problems, and solutions (Sackmann et al., 2013), the development of case-specific sub-methods appears reasonable. Thus, in this contribution, we only focus on the operability of stationary response activities and mobile transportation activities relying on mobile resources. The other types of dependencies (activity-to-activity and resource-to-resource) are not yet addressed in this contribution and mark further research desiderata.

The approach of PRIMA is to identify place-related conflicts by comparing the execution place of an activity with the place of the required resources. Since, e.g., transportation activities could take a certain resource from its place of origin to somewhere else, it must be assumed that the place of mobile resources can change during the runtime of a DRP and, hence, subsequent response activities might become inoperable. Therefore, a pure comparison between places of activity and resource is not sufficient and the sequence of activities has also to be taken into consideration. To achieve such a process-based view, PRIMA is divided into four parts that are discussed in the following in more detail:

1) Transformation of a DRP model into a formal graph representation
2) Provision of a resource-based view, i.e. preparing the graph to be analyzed
3) Identification of place-related conflicts for a given DRP
4) Interpretation of detected conflicts and suggestion for countermeasures.

To explain the working of the PRIMA approach, we describe its mode of operation by a simplified DRP which is not complex, possibly far from a “realistic” DRP but well suited for didactical reasons. We assume a process with 10 activities, several parallel branches, and only one mobile resource used by several activities (see Figure 1).

Part I: Transformation of a DRP model into a formal graph representation

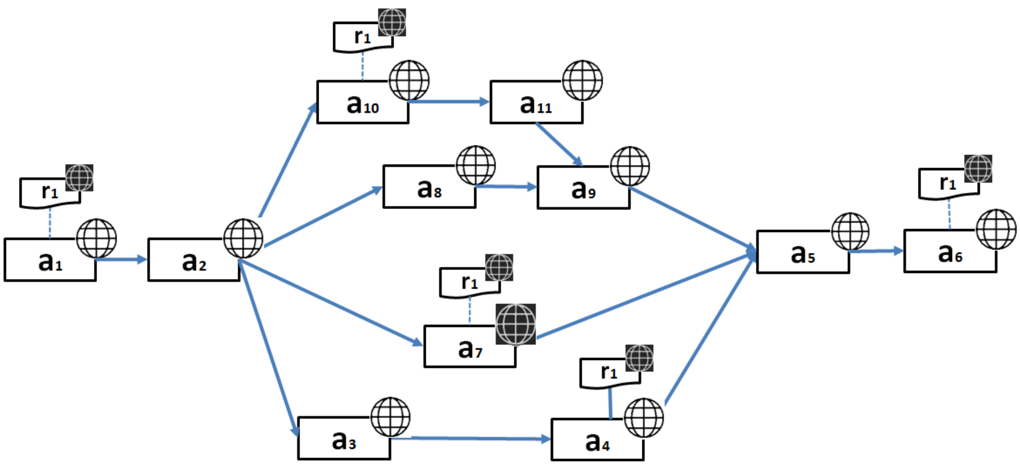

Figure 1: Exemplary DRP model with 10 activities and one resource

Since there are many (formal) modeling languages for processes available, the transformation of their respective structure to a formal graph representation is not discussed in more detail. Rather, it is assumed that a method or wrapper tool already exists that realizes this transformation and provides a sound graph representation. The specification of the required elements is as follows:

Definition 1 [Disaster response process]: We assume that a *disaster response process DRP* is a directed graph $DRP = (A, E)$ which comprises a set of *response activities* $A = \{a_1, \ldots, a_n\}$ as vertices and a set of directed edges E which comprises ordered pairs $e = (a_i, a_n)$ as directions from $a_{i,}$ to a_n. Furthermore, for allowing a complete analysis, we assume that each DRP has exactly one starting and one ending activity (that also could be a dummy).

Definition 2 [activity]: An *activity* a_i denotes a certain unit of work to counteract a disaster event. In this contribution, activities are semantically distinguished into response activities (such as firefighting or rescue work) and transportation (or communication) activities that bring activity-related mobile resources on-site. On a formal level, both are described in a similar manner and defined as a tuple as follows:

$$a_i := (id, name, dependency, kind, origin, delivery, directPredecessor, resources)$$

- *id* as a unique identifier of a certain activity
- *name* which describes the activity
- *dependency* ∊ *DependencyType* = {stationary, mobile}
- *kind* ∊ *ActivityKind* = {transport, response}
- Place-related information (e.g. specified by coordinates)
  - *origin* as place of execution of a response activity or origin of a transportation activity
  - *delivery* as place of delivery of a transportation activity (undefined for response activities)
- *directPredecessor* = $\{\emptyset, a_n, \ldots, a_m\}$ as a set of direct predecessor activities of a_i. The set of direct predecessors can be generated by analyzing the set of direct edges *E* accordingly.
- *resources* = $\{\emptyset, r_n, \ldots, r_m\}$ as a set of required resources ∊ *R* (see Definition 3).

As already mentioned, response activities are assumed as stationary while transport activities are assumed as mobile in this contribution. Taking mobile response activities into consideration is seen as part of a future extension of our basic PRIMA method; however, it is not expected to change the general approach and basic algorithms presented here.

Definition 3 [resources]: A *resource* r_j denotes a non-sharable resource that is necessary to perform a response activity (e.g. actors, information, material, supplies, machines, etc.). Resources are defined as a tuple as follows:

$r_j := (id, name, dependency, origin)$

- *id* as a unique identifier of a certain resource
- *name* which describes the resource
- *dependency* ∊ *DependencyType* = {stationary, mobile}
- *origin* as place where the resource is located (e.g. specified by longitude and latitude coordinates)

Since all resources are assumed as mobile resources, *dependency* is not really required by our basic method. However, since it will be required for future analysis, we have already integrated this characteristic into our model . All resources together define the formal *set of resources* $R = \{r_1, \ldots, r_m\}$.

As already mentioned, we make some assumptions with regard to the characteristics of resources. Firstly, resources are seen as non-sharable and, therefore, they are exclusive in their use and cannot be used by different activities at the same time. Secondly, resources are seen as not consumed and independent of each other, thus, they can be analyzed separately. Last but not least, resources are seen as mobile and can be transported by transport activities. As a matter of course, these assumptions together might be somewhat unrealistic for many DRP. Integrating sharable, consumable, and stationary resources into the basic PRIMA approach would mean a considerable extension, e.g. by a specific quantity structure and its analysis. Again, this is not expected to change the general approach presented in this contribution.

Part II: Provision of a Resource-Based View

The method `Search_predecessor()`(see Figure 2) generates a resource-based view of the DRP graph for each single resource by identifying all activities relying on it and determining the predecessor relationship between them. In the following, the algorithm is explained step by step and by means of the introduced example DRP (Figure 1). Since this DRP is simplified and only contains one resource, the algorithm has to be executed only once. However, if several resources are used, it has to be executed for each single resource $r_j \in R$ separately.

Step (*1) specifies for each resource $r_j \in R$ the set of its using activities $RA_j \subseteq A = \{a_1, \ldots, a_n\}$. Since there is only one resource used in our example, the set RA_1 comprises the following elements: $\{a_1, a_4, a_6, a_7, a_{10}\}$.

Step (*2) picks one activity $a_i \in RA_j$ to identify its predecessor relationship in regard to the remaining activities in RA_j.

Step (*3) checks if a_i is the start node of the graph and has no predecessor. In this case, a_i is inserted in an internal `predecessorList` which, in the end, contains every element from RA_j and its resource-related predecessors as additional element(s). For marking root elements, a self-reference of a_i is used. In our example DRP, activity a_1 is such a root activity and, thus, `predecessorList` would be extended by the element $(\{a_1, \{a_1\})$.

Step (*4) analyzes all a_i which are no root activities. To start the subsequent breath-first-search, a `searchList` is initialized with the set of a_i`.directPredecessor` containing all activities that have still to be analyzed in regard to their predecessor-relationship to a_i (*4a). As long as this `searchList` is not empty (*4b), the first list entry is taken as current `searchNode` (*4c) to examine if an activity $a_n \in$ `searchNode.directPredecessor` (*4d) uses the resource r_j, i.e. if $a_n \in RA_j$ (*4e). For not analyzing an activity a number of times, a further list `consideredNodes` is managed including all activities already examined (*4f).

```
Method Search_predecessor(rj)

RAj = []                                                            / (*1)
For each ai ∈ A                                                     / (*1)
     If (rj ∈ai.resource)                                           / (*1)
          RAj = RAj ∪ {ai}                                          / (*1)

predecessorList = []
pre_list_r = []
searchList = []
searchNode
consideredNodes = []

for each ai ∈ RAj                                                   / (*2)

     if (ai.directPredecessor is empty)                             / (*3)
          predecessorList = predecessorList∪ {ai,{ai}}              / (*3)
     else
          searchList = ai.directPredecessor                         / (*4a)
          consideredNodes = []

          while (searchList not empty)                              / (*4b)
               searchNode = first element of searchList             / (*4c)

               for each an∈ searchNode.directPredecessor            / (*4d)
                    if (an ∉consideredNodes)                        / (*4f)
                         consideredNodes = consideredNodes ∪{an}    / (*4f)
                         if (an ∈ RAj)                              / (*4e)
                              prelist_r = pre_list_r ∪ {an}         / (*5a)
                         else
                              if (an.directPredecessor is empty)    / (*5b)
                                   prelist_r = pre_list_r ∪ {ai}    / (*5b)
                              else
                                   searchList = searchList ∪ {an}   / (*5c)
               remove searchNode from searchList                    / (*5d)

     predecessorList = predecessorList∪ {(ai, pre_list_r)}          / {*6}
     clear pre_list_r                                               / (*6)

Return predecessorList                                              / (*7)
```

Figure 2: Method Search_predecessor (pseudo code)

For demonstrating the steps we use activity a_6 from the example DRP where searchList is initiated as $\{a_5\}$. After a first iteration of the algorithm, the status for a_6 looks as follows:

$a_i \in RA_j$	searchList	searchNode	searchNode. directPredecessor	a_n	consideredNodes	pre_list_r
a_6	$\{a_5\}$	a_5	$\{a_4, a_7, a_9\}$	a_5	{}	{}

Step (*5) checks each predecessor a_n of searchNode for three different cases. In case one, a_n relies on r_j and, thus, the nearest predecessor of a_i using the resource has been found that is added to pre_list_r (*5a). This additional list is collecting all direct and indirect predecessors of a_i which also rely on the considered r_j. In the second case (*5b), a_n is the start node of the graph, which means that a_i could be the first activity using r_j. Thus, self-reference of a_i is added to pre_list_r. In the third case a_n does not rely on r_j so that the algorithm has to continue search on the path the predecessor a_n lies on. Therefore, a_n is added to the searchList (*5c). In regard to the example DRP, a_5.directPredecessor contains $\{a_4, a_7, a_9\}$. Since a_4 is the first element in list, this activity is analyzed first and put to consideredNodes, followed by a_7 and a_9. It is determined that a_4 and a_7 are using resource r_1 and, thus, both are put on pre_list_r. In contrast, a_9 is put to the searchList. As last step, the current searchNode is removed from searchList so that the loop can terminate (*5d). After this, the status of our algorithm looks as follows:

$a_i \in RA_j$	searchList	searchNode	searchNode. directPredecessor	a_n	consideredNodes	pre_list_r
a_6	$\{a_5\}$	a_5	$\{a_4, a_7, a_9\}$	a_4	$\{a_4\}$	$\{a_4\}$
a_6	$\{a_5\}$	a_5	$\{a_4, a_7, a_9\}$	a_7	$\{a_4, a_7\}$	$\{a_4, a_7\}$
a_6	$\{a_9\}$	a_5	$\{a_4, a_7, a_9\}$	a_9	$\{a_4, a_7, a_9\}$	$\{a_4, a_7\}$

In the next loop, a_9 is the first element of the searchList and analyzed correspondingly. The status of the algorithm evolves as depicted in the following table and is repeated until searchList is empty.

$a_i \in RA_j$	searchList	searchNode	searchNode. directPredecessor	a_n	consideredNodes	pre_list_r
a_6	$\{a_9, a_8\}$	a_9	$\{a_8, a_{11}\}$	a_8	$\{a_4, a_7, a_9, a_8\}$	$\{a_4, a_7\}$
a_6	$\{a_8, a_{11}\}$	a_9	$\{a_8, a_{11}\}$	a_{11}	$\{a_4, a_7, a_9, a_8, a_{11}\}$	$\{a_4, a_7\}$
a_6	$\{a_{11}, a_2\}$	a_8	$\{a_2\}$	a_2	$\{a_4, a_7, a_9, a_8, a_{11}, a_2\}$	$\{a_4, a_7\}$
a_6	$\{a_2\}$	a_{11}	$\{a_{10}\}$	a_{10}	$\{a_4, a_7, a_9, a_8, a_{11}, a_2, a_{10}\}$	$\{a_4, a_7, a_{10}\}$
a_6		a_2	$\{a_1\}$	a_1	$\{a_4, a_7, a_9, a_8, a_{11}, a_2, a_{10}, a_1\}$	$\{a_4, a_7, a_{10}, a_1\}$

Step (*6) and (*7): when searchList is empty, the search for predecessors of a_i using the considered resource r_j is terminated. As a result, the pre_list_r of dedicated a_i is added as new element to the predecessorList. In our example DRP, the element $(a_6, \{a_4, a_7, a_{10}, a_1\})$ would be added. Thereafter, pre_list_r is cleared.

Finally, when all $a_i \in RA_j$ have been examined, predecessorList contains for each a_i its set of associated direct and indirect predecessors (*7) which also rely on a considered r_j. In our example DRP, the result of the algorithm would be $\{(a_1, \{a_1\}), (a_4, \{a_1\}), (a_6, \{a_4, a_7, a_{10}, a_1\}), (a_7, \{a_1\}), (a_{10}, \{a_1\})\}$. The result now provides a resource-based view on the graph. For each activity a_i that uses the considered resource r_j, it contains a set of ordered pairs specifying only those activities that are, in regard to the sequential order of the graph, direct predecessor(s) that also use r_j.

Part III: Identification of Place-Related Conflicts

The third part of PRIMA uses the results from Search_predecessor() to identify possible place-related conflicts for a given DRP. This is achieved by validating whether the place of an examined activity is consistent with the place of the required resource(s). Therefore, the method Validation_place() has been developed (Figure 3) that is presented in the following. The proposed method operates with predecessorList (*1) passed by the method Search_predecessor() and compares the places between an a_i relying on a certain r_j and each of its direct predecessors also using the resource (*2). The list comprises elements in the form of ordered pairs $(a_i, \{a_x, \ldots, a_n\})$ whereby the first element depicts the activity to be examined and the latter comprises a list of its direct predecessors. During this comparison, the following cases are analyzed with regard to the activity examined (a_i):

- there is a path where a_i has no predecessor (*3) and the ordered pair is referring to itself. In our exemplary DRP this is true for a_1 with its `predecessorList = ({a1, {a1})`. Therefore, it is checked whether the place of origin of the resource is the same as the one of the activity to be examined (*3a). If this is not the case, the activity will not be operable (*3b). For interpretation afterwards, this is added to the list of errors with code “11”.
- the chosen predecessor a_x is a transport activity (*4): the algorithm checks whether its destination a_x`.delivery` is different from the location of the following activity a_i (*4a). If both places do not match, the activity will not be operable and, hence, an error-entry is generated with error code “12” (*4b). In our exemplary DRP this would be the case, e.g., if a_7 is a transport activity and does not end at the place of origin of a_6.
- the chosen predecessor a_x is a response activity (*5): the algorithm checks if both activities are assigned to the same place (*5a). If the places do not match, again, an error-entry with error code “13” is generated (*5b). In our exemplary DRP this would be the case, e.g., if a_4 is a response activity and does not take place at the place of origin of a_6.

After each predecessor a_x is analyzed, the place-related errors for a_i are concluded. If all predecessors produce an inconsistency (*6a) a_i will not be operable in any case, since all its predecessors end at a different location than a_i. This is logged to the list of errors with error code “02”. If no inconsistency has shown up (*6b), activity a_i will be operable, since all predecessors to a_i end at its place of origin. This is logged to the list of errors with error code “00”. If there is more than one predecessor to a_i and the results of the validation are not consistent (*6c), it is not possible to decide whether the activity will be operable or not by a pure graph analysis. This is logged to the list of errors with error code “01”. In our exemplary DRP, this would be the case, e.g., if a_4 does not take place at the origin of a_6 while a_7 does. To decide the final operability of a_6, further analysis is required as sketched in the following Part IV of the PRIMA method.

```
Method Validation_place (Predecessor_list, rj)
Error_count = 0
Error_list = []

for each (ai,Pre_listi) ∈ Predecessor_list                       / (*1)
     for each ax ∈ Pre_listi                                      / (*2)

          if ax = ai                                              / (*3)
               if (ai.origin != rj.origin))                       / (*3a)
                    Error_list = Error_list ∪ {code11, ai, ax, rj} / (*3b)
                    Error_count = Error_count + 1
          else
               if (ax.kind == 'transport')                        / (*4)
                    if (ax.delivery != ai.origin)                 / (*4a)
                         Error_list = Error_list ∪ {code12, ai, ax, rj} / (*4b)
                         Error_count = Error_count + 1
               else                                               / (*5)
                    if (ax.origin != ai.origin)                   / (*5a)
                         Error_list = Error_list ∪ {code13, ai, ax, rj} / (*5b)
                         Error_count = Error_count + 1

     if (Error_count == Pre_listi.size)                           / (*6a)
          Error_list = Error_list ∪ {code02, ai, {}, rj}
     else
          If (Error_count == 0)                                   / (*6b)
               Error_list = Error_list ∪ {code00, ai, {}, rj}
          else                                                    / (*6c)
               Error_list = Error_list ∪ {code01, ai, {}, rj}

Return Error_list
```

Figure 3: Method Validation_place (pseudo code)

The result now provides a complete list of inconsistencies as well as a final statement about the operability of all activities with respect to the resource r_j.

Part IV: Interpretation of the Results

The result of Part III is a list of identified errors. Since the validation of place-related consistency between two activities using the same resource does not cover all situations exhaustively, further manually interpretation is necessary at the current level of development. To support interpretation, the results could be illustrated in a simplified way, e.g. by a graphical presentation of the errors. In a first step, this could be realized by symbols such as simple traffic lights, e.g., showing activities that are executable as green (code00), depicting activities that are not executable as red (code02), and non-decidable activities as yellow (code01). Furthermore, the identified errors could be presented in conjunction with concrete proposals for process adaptation to make an activity operable, e.g. by providing a user interface which allows the correcting of place characteristics of resources or activities as well as by proposing additional transport activities for making the graph valid. This provides a sound basis for identifying possible conflicts with regard to the operability of a given DRP that requires further intervention by the tactical echelon of DRM (e.g. by process adaptation).

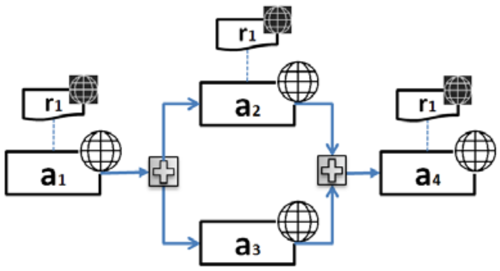

Figure 4: Example for Branch Analysis with AND split

Moreover, further improvement of the results and their interpretation could be achieved by a deeper analysis of parallel branches within the DRP. Since the implication to predecessor relationship is different depending on whether branch types are following an AND or an XOR split, further research to extend the PRIMA approach seems very promising. However, as a basic, a short description of a simple AND branch is suitable for demonstrating the general approach and potential: since an AND join means that all paths have to be executed before the next activity can take place, the whole construct can be considered to be a sub-graph. Thus, it can be interpreted as an abstract independent node from the view of a superior graph and some errors (code01) could become easily decidable. A simple example is depicted in Figure 4. Assuming that a_1`.origin` is at place A, a_2 represents a transport activity from place A to place B, and activity a_4 takes place at place B, it is easy to see that this would be a perfect match. However, one result of the `Error_list` would be (`code01`, a_4, a_1, r_1), which means that the activity a_4 might not be executable. Therefore, taking the AND connectors into consideration by firstly analyzing the sub-graph could provide more accurate results. Furthermore, the analysis could be improved by taking temporal behavior of the DRP into consideration. Assuming that activities are characterized by, e.g., execution time, common techniques for process analysis (e.g. path analytics, constraint analysis, scenarios, simulations, etc. (Long, 2012)), means a further possibility for enhancing and improving the place-related analysis of our PRIMA approach.

CONCLUSION AND OUTLOOK

The graph-based method presented in this contribution is designed to identify activities within a given DRP that might not be operable due to incompatible places of activities and their required resources. The method contains four parts that build upon each other and cover the transformation of a DRP into a graph theoretical representation, the provision of a resource-based view, the identification of place-related inconsistencies and resulting non-operability of activities, and a first interpretation of the results. On its current level of development, our PRIMA method informs disaster managers, e.g. at the tactical echelon of DRM, about place-related inconsistencies in the process flow that can be expected to result in a non-operable DRP and, therefore, should be mitigated with high priority. The implementation of a prototype and testing with more realistic process models has already been realized and gives first evidence for the applicability of the method. The calculating performance of the algorithm obviously depends on the number of resources, the number of activities using a resource, and the size as well as the complexity (number of branches) of the process model. Although it has not yet been formally proven, concluding from similar graph analytical methods used in BPM, performance should be acceptable at least for usual models. However, this is also a topic for further research. Up to now, the algorithm just provides information about possible place-related conflicts in the process model and it remains part of the human disaster manager to interpret the results and to decide appropriate action. Therefore, a next promising step would be the integration of the method into a DRWfMS providing automated analysis and adaptation of ongoing processes and, thus, valuable decision support. Further open issues result from our limitations and assumptions regarding the considered resources and activities. As yet, resources are always assumed as mobile and transportable to the execution place of a response activity while response activities are assumed as stationary. Furthermore, we do not differ between different resource types (e.g. supplies, machines, information and data or actors) but consider them to be similar to each other. This is also not very realistic compared to real resources in DRP. Thus, further research should also focus on different types of resources, e.g. with regard to their intangibility, reusability, and shareability. In addition, we define places as precise geographical coordinates and assume that each place-related element contains that kind of information. However, in DRM, geographical coordinates are usually not available when designing DRP and are only revealed in the immediate aftermath of a disaster occurrence. The comparing of precise geographical coordinates might lead to misinterpretation too (for instance, in a case where a stationary activity is only fixed to a certain area and an associated resource is fixed to a precise location). Thus, places need a more sophisticated concept than the one used in this contribution, e.g. by a “soft” interpretation component or relative locations.

Although the presented methods provide first satisfying results, numerous open issues still remain in order to facilitate a comprehensive method capable of solving place-related problems in given process models as a whole. Therefore, main research desiderata are discussed with regard to extending current process modeling languages by elements for place-related information and extending the basic identification algorithm by taking process logic (branch types) and further process properties (e.g. temporal characteristics) into consideration. Integrating such a comprehensive analysis into future DRWfMS would provide a plethora of opportunities– abandoning them would mean leaving enormous potential wasted.

REFERENCES

1. Bassett, D. (2008) Place, Disasters, and Disability in Law and Recovery from Disaster: Hurricane Katrina, *Law and Recovery from disaster: Hurricane Katrina*, University of Alabama Public Law Research, Paper No. 1287372.
2. Chen, R., Sharman, R., Rao, H.R. and Upadhyaya, S. J. (2008) Coordination in emergency response management, *Communications of the ACM*, 51, 66–73, New York, ACM Press.
3. Dadam, P., Reichert, M., Rinderle, S.B., Jurisch, M., Acker, H., Göser, K., Kreher, U. and Lauer, M. (2007) ADEPT2 - Next Generation Process Management Technology, *Proceedings Fourth Heidelberg Innovation Forum*, Heidelberg, Germany, D.punkt Verlag.
4. Fahland, D. and Woith, H. (2009) Towards process models for disaster response, *Business Process Management Workshops*, 254–265. Berlin/ Heidelberg, Springer.
5. Fleischhauer, M. (2008) The role of spatial planning in strengthening urban resilience, *Resilience of Cities to Terrorist and other Threats* (pp. 273-298). Springer Netherlands.
6. Georgakopoulos, D., Schuster, H., Baker, D. and Andrzej Cichocki (2000) Managing Escalation of Collaboration Processes in Crisis Mitigation Situations, *Proceedings of the 16th Int. Conference on Data Engineering*, San Diego.
7. Hofmann, M., Sackmann, S., and Betke, H. (2013a). A Novel Architecture for Disaster Response Workflow Management Systems, *Proceedings of the 10th Int. ISCRAM Conference*, Baden- Baden, Germany.
8. Hofmann, M., Sackmann, S. and Betke, H. (2013b) Using Workflow Management Systems to Improve Disaster Response Processes, *5th IWDENS in conjunction with 27th IEEE AINA-2013*, Barcelona.
9. Jansen, J.M., Lijnse, B. and Plasmeijer, R. (2010) Towards Dynamic Workflow Support for Crisis Management, *Proceedings of the 7th Int. ISCRAM Conf*erence, 1–5. Seattle, USA.
10. Long, K. A. (2012) Overview of Common Process Analysis Techniques, *Business Rules Journal*, 13, 12.
11. Rueppel, U. and Wagenknecht, A. (2007) Improving emergency management by formal dynamic process-modelling, *24th Conf. on Information Technology in Construction*, 559–564. Maribor, Slovenian.
12. Sackmann, S., Hofmann, M. and Betke, H. (2013) Towards a Model-Based Analysis of Place-Related Information in Disaster Response Workflows, *Proceedings of the 10th Int. ISCRAM Conference*, Baden- Baden, Germany.
13. Sell, C. and Braun, I. (2009) Using a workflow management system to manage emergency plans, *Proceedings of the 6th Int. ISCRAM Conference*, Gothenburg, Sweden.
14. Ziebermayr, T., Huber, J., Kollarits, S. and Ortner, M. (2011) A Proposal for the Application of Dynamic Workflows in Disaster Management: A Process Model Language Customized for Disaster Management, *22nd International Workshop on DEXA*, Toulouse, France.