Collaborative Process Flexibility Using Multi-Criteria Decision Making

Ioannis Patiniotakis, Nikos Papageorgiou, Dimitris Apostolou, Yiannis Verginadis, Gregoris Mentzas

Collaborative Process Flexibility Using Multi-Criteria Decision Making

Ioannis Patiniotakis[1], Nikos Papageorgiou[1], Dimitris Apostolou[2], Yiannis Verginadis[1] and Gregoris Mentzas[1]

[1]Institute of Communications and Computer Systems,
National Technical University of Athens
9 Iroon Polytechniou Str., Athens, Greece
{ipatini@mail.ntua.gr, npapag@mail.ntua.gr, jverg@mail.ntua.gr, gmentzas@mail.ntua.gr}
[2]Informatics Department, University of Piraeus
80 Karaoli & Dimitriou Str., Piraeus, Athens, Greece
{dapost@unipi.gr}

Abstract. The ability to deal with both foreseen and unforeseen changes in a collaborative process (also known as Collaborative Process flexibility) is considered critical for the business process management systems. In this paper, we present an approach that uses a modeling framework and engine called Situation Action Network (SAN), along with multi-criteria decision making methods and techniques (MCDM). Such combination introduces event-driven flexibility in collaborative processes. We discuss and implement appropriate notions and mechanisms in order to alleviate a part of the modeler's effort during the design of hierarchical rules (i.e. SAN trees). This increases their run time flexibility and support the adaptation of collaborative business processes. We validate our approach using an illustrative scenario taken from the nuclear crisis management domain.

Keywords: Collaborative Processes, Flexibility, Adaptation, MCDM, SANs

1 Introduction

In today's agile business environment, information systems should be able to model business needs in a flexible manner in order to be effective. It must allow domain experts to model business systems with means that are intuitively comprehensible using concepts that are familiar to them, such as goals and actions for achieving them. Goal-orientation is based on separating the declarative statements and defining desired system behavior, thus hiding from business users the low-level system details [1]. This flexibility is even more imperative when business information systems should deal with and support collaborative processes. Collaborative Process flexibility can be seen as the ability to deal with both foreseen and unforeseen changes in a collaborative process, by varying or adapting promptly those parts that are affected.

Situation Action Networks (SANs) is a modeling framework that can be used for defining business systems' reactions to significant situations with the purpose of fulfilling or satisfying a goal [2]. SANs are hierarchical goal-directed models that

comprise nodes with specific semantics, presenting possible decomposition paths from goals into subgoals and primitive actions. Goals are related to situations that trigger their activation and reactions that should be performed towards achieving these goals when certain conditions are met (Fig. 1). SANs are modeled and executed by the SAN editor and SAN engine, respectively. SAN engine can enable automatic search for new goals when specific circumstances, i.e. situational and contextual settings, arise and recommend actions in order to satisfy the currently active goals.

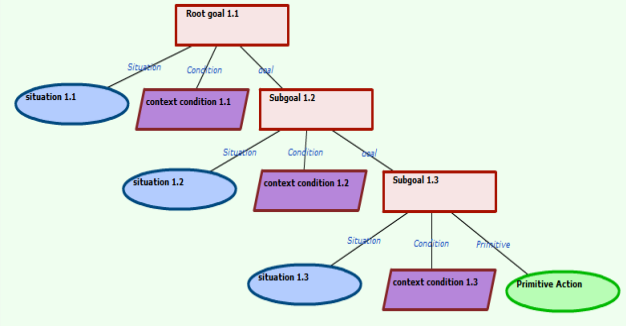

Fig. 1. Basic Modeling Primitives of SANs

The SAN framework has been coupled with an aspect-oriented extension of BPMN2.0 in order to enable the execution of BPMN2.0 processes and support business process adaptation [3]. In that work, SANs are used to monitor the process execution environment and describe meaningful reactions to problems. Based on these monitoring capabilities, SANs are able to detect process execution problems (e.g. violation on quality "threshold") and trigger lookup for suitable adaptation actions by using a reasoning mechanism. The implementation of the recommended adaptation is based on aspect-oriented programming (AOP) techniques [4]. We note that the actual SAN modeling is not considered trivial and dictates for an editor that is domain expert and is capable for conceptually validating the outcome.

Despite the intuitiveness of goal-orientation for business users, modelling a goal-oriented business process to its full detail can be tedious and time-consuming. Furthermore, not all process parameters may be known a priori, or large numbers of alternative goal decompositions are possible, thus making the development of a detailed model impractical or even impossible. In order to remedy these issues, process modeling flexibility with SANs can be improved with techniques that enable them to dynamically "choose" the required or desired detailed goal decompositions.

In this paper, our research objective is to extend the SAN modeling framework with the appropriate constructs and corresponding methods in order to alleviate a part of the modeler's effort during the design of BPMN2.0 processes and increase their run time flexibility.

Different kinds of flexibility are needed during the life cycle of a business process. There have been numerous studies of flexibility in business processes oriented systems, both in terms of the factors which motivate it and the ways in which it can be achieved. The following papers cover extensively the literature on both the notion of process flexibility as well as design and evaluation approaches of flexible process-

aware systems [5], [6], [7], [8], [9], [10]. Our approach focuses on relieving the modeler from having to a priori map situations to specific actions that should be enacted when the situations occur. Instead, a pool of primitive actions or even linked SANs can be mapped at design time with situations. These pools may be dynamically updated at run time. The framework extension involves the capability to automatically select appropriate primitive actions based on real time evaluated criteria using Multi-Criteria Decision Making (MCDM) methods. We formally present this extension, we discuss the necessary SAN engine extensions and we conclude with an illustrative scenario that exhibits the flexibility improvements.

2 Flexibility Approach

We introduce two new node types in the SAN formalism, named Abstract Action and Action Pool that enable dynamic (at runtime) "choice" over the required or desired goal decompositions. This dynamic behaviour of SANs along with the use of AO4BPMN2.0 (aspect-oriented extension of BPMN2.0, presented in [3]), induces flexibility in collaborative business processes.

Abstract actions are leaf nodes in SANs that when visited (during SAN traversal), they determine on-the-fly the tasks they should be performed, selecting them from a repository of alternative actions, called Action Pool. For that purpose search and selection multi-criteria decision making methods (MCDM) are specified beforehand. In our implementation we have used the Linguistic Ordered Weighted Average (LOWA) [11] method that acts on linguistic variables, such as words or sentences in a natural language, and produces (linguistic) ranking. The tasks can range from primitive actions up to whole SANs. Abstract action nodes replace themselves with the task(s) that are automatically selected from an associated action pool, based on the search and selection method and resolution policy used. Abstract Action nodes must be annotated with metadata providing the necessary configuration for the search and selection methods. These metadata specify the criteria that will be used along with their weights. Additional metadata control the ordering, the size of results list and filtering ("LOWA Criteria", "Ascending Order", "Results Count", Allowed Values", "Results Filter", "Mapping"). All the annotations needed both in abstract actions nodes and in action pools are inserted through the SAN editor.

An Action Pool contains several possible alternative decompositions for an abstract action. Normally the contained items should all provide decompositions for the same purpose or goal. An action pool can be used by more than one abstract action, possibly in different SANs. It's worth noting that Action Pools can either be predefined sets of decompositions, or they can be retrieved / generated at runtime from an online service, for instance databases or web services. Action Pools must be annotated firstly with criteria parameters. Any metadata defined for annotating an Action Pool actually denote mandatory metadata that should also appear in the action pool items. Some or all of the action pool metadata can be used as search and selection criteria in LOWA method or any other MCDM methods that might be defined in the future. All action pool Items must be annotated with values, pertaining

to the criteria defined in action pool. These values are used to calculate the average scores of action pool items using LOWA. The values can either be linguistic terms, numeric or text values. In case of linguistic terms a mapping parameter is used for specifying how the linguistic terms map to actual criteria values.

A Search and Selection Method is responsible to scan all action pool items, examine their properties and figure out which of them fit the specific purpose of the abstract action node. When more than one action pool items fit the purpose, it is again the responsibility of search and selection method to pick the best one. In some cases, however, it is acceptable (or even desired) to select the top N best items or all matching items. In such cases an ordering is required. We used the multi-criteria decision making method LOWA for performing this ranking of the action pools items.

A Resolution Policy is needed when the search and selection method returns more than one items. It is responsible to make the final selection of the item that will be used to replace the abstract action node. It is allowed to select more than one item, in which case it must define the way the items should be combined. For the time being five resolution policies have been defined: i) use the item at the 1st place (best ranked item) in the results list, ii) select an item from the results list randomly, iii) combine all matching items into a "Parallel Any" action, iv) combine all matching items into a "Parallel All" action, v) combine all matching items into a "Parallel Timeout" action.

2.1 Algorithm and Implementation

LOWA has been used as MCDM method to select and rank the action pool items, using linguistic terms. This method accepts the following input arguments from SAN engine core: i) reference to the "owner" abstract action, ii) reference to the specified action pool, iii) reference to the local context. These are references to various SAN model entities that contain configuration data that the method needs in order to work appropriately. Configuration data are stored as metadata annotations on the arguments passed to SAN model entities. Using the above input, the following algorithm (Table 1) is implemented by our system in order to properly use the abstract action capabilities of SAN formalism.

Table 1. SAN Search & Selection

Algorithm SAN Search & Selection

```
Require: self reference to abstract action node
items ← get_action_pool_items( get_action_pool(self) )
criteria ← get_criteria(self)
allowed_values ← get_allowed_values(self)
mapping ← get_mapping(self)
weights ← get_criteria_weights(self, criteria)
for each item I in items do
   values ← empty list
   for each criterion C in criteria do
         item_value ← get_item_value(I, C)
         if mapping ≠ Ø then
                  item_value ← translate_value(mapping of C, item_value)
```

```
            end if
            add item_value to values
        end for
        lowa_value ← calculate_lowa(values,weights,#criteria,allowed_values)
        IT.lowa_score ← lowa_value
    end for
    remove elements of list 'items' with 'lowa_score ≠ filter'
    order elements of list 'items' based on 'lowa_scores' with direction 'order'
    remove elements of list 'items' after position 'count'
```

The dynamic decomposition functionality described has been incorporated in the reference SAN engine by extending its base API and implementing extensions at the traversal engine and SAN repository (Fig. 2). All components along with their extensions were coded using Java programming language. The Adaptation Manager component handles the coupling of SAN engine with the aspect-oriented extension of BPMN2.0 (presented in [3]) in order to enable the execution of BPMN2.0 processes and support business process adaptation. Its primary role is to closely monitor the execution of process instances and activate the corresponding advices based on SAN traversal (further details can be found in [3]).

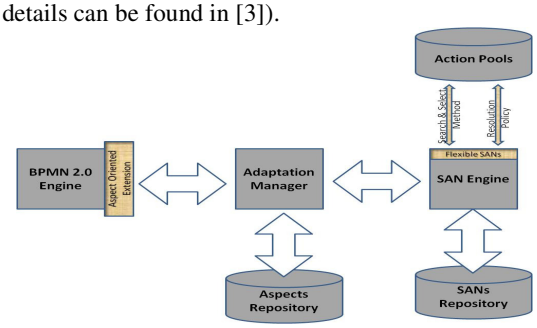

Fig. 2. Conceptual Architecture

3 Illustrative Scenario

We use the nuclear crisis management scenario[3], where a large quantity of radioactive substance is accidentally released in the atmosphere. A number of mitigating actions take place based on a predefined collaborative process that involves several different actors, authorities and services (e.g. police, military, fire brigade, national institute for radioprotection and nuclear safety, representatives of national authority etc.). In this scenario instead of pre-modeling (in one SAN) all possible adaptation actions, we use the notion of abstract action and action pool in order to be able to select adaptation advices at run time based on the current context.

The SAN (Fig. 3) designed, takes advantage of an action pool and instead of having a pre-designed recommendation process of generating advices, when "Study Advice" gets delayed, the abstract action "Recommend of weaving a Wf Advice" is used. This abstract action is associated with a number of alternative advices that could

be suggested, based on the current context information. In this scenario Radiation Level and Weather Conditions are used as selection criteria for the search and selection method. The contextual values of these two criteria are derived from the field and meteorological event streams, and they are expressed as linguistic terms taking one of the values: Normal, Notify, Alert, Dangerous and Critical. The SAN Designer can set the weights for these criteria (e.g. 70% Radiation Level, 30% Weather Conditions). We use an Action pool that includes items that generate adaptation advices (i.e. recommendations):

1. *Advice 2 without interrupt* (Radiation Level: ALERT, Weather Conditions: ALERT). Since a dangerous or critical situation has not been detected yet, the involved actors can afford to wait for this study to be finished before the Representative of the National Authority asking for additional clarifications. When both the "study advice" and "study clarifications" tasks finish, the normal workflow execution can proceed. The Advice 2 involves the weaving after the "study advice" task the following sequential ones: "Ask for clarifications", "Wait and receive clarifications", "Study clarifications".

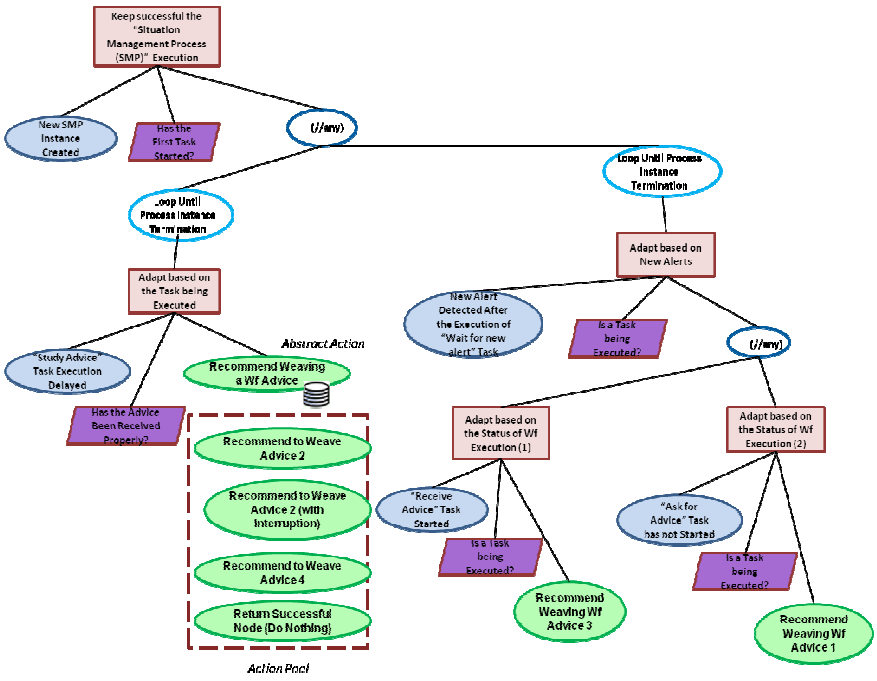

Fig. 3. SAN deployed for Collaborative Process Flexibility

2. Advice 2 with task interruption (Radiation Level: CRITICAL, Weather Conditions: DANGEROUS). In this case the reactions should be accelerated, so the task "Study Advice" is interrupted and the advice 2 is weaved.

3. Advice 4 (Radiation Level: DANGEROUS, Weather Conditions: NORMAL). Although, a dangerous situation has been detected, there is some time (since weather conditions are normal) to consider carefully the proper reactions. The important thing in such situation is to take the best possible decision that is why advice 4 is weaved after the "Study Advice" task has been completed with a delay (no interruption this time). The Advice 4 involves the weaving after the "study advice" task the following sequential ones: "Ask for activity report", "Wait and receive activity report", "study activity report".

4. Do Nothing - normal workflow execution (Radiation Level: NOTIFY, Weather Conditions: NORMAL). Although SAN has detected a significant delay in the "Study Advice" task, it is not meaningful to weave any adaptation action since the weather conditions are normal, thus the radiation cloud will not move fast. This means the there is the necessary time for the involved actors to take the correct decisions.

The following figure presents recommendations generated by our system, and published as events, using the aforementioned setup.

Fig. 4. Recommendation to weave Advice 2

4 Conclusions

In this paper, we presented an extension of the Situation Action Network (SAN) modelling framework in order to introduce run-time flexibility in collaborative processes. The extension enables dynamic decomposition of SANs allowing for abstract definition of SANs at design time. Dynamic decomposition enables more

flexible and modular modelling, since the decomposition details can be modelled separately from the main problem. This separation facilitates SANs' reuse and improves the usefulness and applicability of each main model. On the other hand, the decomposed models (contained in Action Pools) can be treated as runtime libraries and also be re-used. Further, our approach implements an abstraction layer over the main modelling process by removing the decomposition search, selection and execution method details. Finally, decompositions for various purposes, in the form of action pools, can be developed and provided by third parties, or collaboratively be built from a group of designers. Continuing this work on flexible SANs, we plan for an extensive evaluation of our approach where any lag or limitations in terms of recommendations speed, will be measured.

Acknowledgments. This work has been partially funded by the European Commission under project PLAY (Grant FP7-258659). The authors would like to thank the project partners for their advices and comments regarding this work.

References

1. Mylopoulos, J., Chung, L., Yu, E.S.K.: From Object-Oriented to Goal-Oriented Requirements Analysis. Commun. ACM 42(1): 31-37 (1999)
2. Patiniotiakis, I., Papageorgiou, N., Verginadis, Y., Apostolou, D., Mentzas, G.: Dynamic Event Subscriptions in Distributed Event Based Architectures. International Journal: Expert Systems with Applications, Elsevier, Volume 40, Issue 6, pp. 1935–1946, (2013)
3. Patiniotiakis, I., Papageorgiou, N., Verginadis, Y., Apostolou, D., Mentzas, G.: An Aspect Oriented Approach for Implementing Situational Driven Adaptation of BPMN2.0 Workflows. In 6th International Workshop on Event-Driven Business Process Management collocated with BPM 2012, Tallinn, Estonia, (2012)
4. Charfi, A., Mezini, M.: AO4BPEL: An Aspect-Oriented Extension to BPEL. World Wide Web Journal: Recent Advances on Web Services, special issue (2007)
5. Snowdon, R.A., Warboys, B.C., Greenwood, R.M., Holland, C.P., Kawalek, P.J., Shaw, D.R.: On the Architecture and Form of Flexible Process Support. Software Process Improvement and Practice, 12:21–34, (2007)
6. Soffer, P.: On the Notion of Flexibility in Business Processes. In Workshop on Business Process Modeling, Design and Support (BPMDS05), pp. 35–42, (2005)
7. Regev, G., Soffer, P., Schmidt, R.: Taxonomy of Flexibility in Business Processes. In Proceedings of the 7th Workshop on Business Process Modelling, Development and Support(BPMDS'06), (2006)
8. Carlsen, S., Krogstie, J., Sølvberg, A., Lindland, O.I.: Evaluating Flexible Workflow Systems. In Proceedings of the Thirtieth Hawaii International Conference on System Sciences (HICSS-30), IEEE Computer Society Press, Maui, Hawaii (1997)
9. Heinl, P., Horn, S., Jablonski, S., Neeb, J., Stein, K., Teschke, M.: A Comprehensive Approach to Flexibility in Workflow Management Systems. In WACC '99: Proceedings of the international joint conference on Work activities coordination and collaboration, ACM pp. 79–88, New York, NY, USA, (1999)
10. Kumar, K. and Narasipuram, M.M.: Defining Requirements for Business Process Flexibility. In Workshop on Business Process Modeling, Design and Support (BP-MDS06), Proceedings of CAiSE06 Workshops, pages 137–148, (2006)
11. Herrera F., Herrera-Viedma E., Verdegay J. L.: Direct approach processes in group decision making using linguistic OWA operators. Fuzzy Sets and Systems, 79, pp. 175–190, (1996)