Logic-based ad-hoc business process management: Concepts and Challenges

Darko Anicic
FZI Karlsruhe
Haid-und-Neu Strasse 10-14
76131 Karlsruhe
+49 721 9654958
Darko.Anicic@fzi.de

Nenad Stojanovic
FZI Karlsruhe
Haid-und-Neu Strasse 10-14
76131 Karlsruhe
+49 721 9654852
Nenad.Stojanovic@fzi.de

Ljiljana Stojanovic
FZI Karlsruhe
Haid-und-Neu Strasse 10-14
76131 Karlsruhe
+49 721 9654804
Ljiljana.Stojanovic@fzi.de

ABSTRACT

The need for unpredicted, real-time changes in the business process is increasing tremendously. However, a well-founded approach for representing these processes in an executable form is missing. In this position paper we introduce the concept of the event-driven ad-hoc business process management and present some of challenges.

Keywords

Ad-hoc BPM, Logic Programming, Complex Event Processing

1. MOTIVATION

An increasing dynamics in today business and life requires flexible infrastructures that can sense a problem or opportunity almost immediately after their occurrence (ideally: before they will appear) and react accordingly. It is especially relevant for the business processes which are underpinning complex work or life situations, like emergency management, since many unexpected events (problems or opportunities) can happen all the time and traditional approaches for coping with the diversity in business processes, like using business rules, are simple not feasible. Indeed, if we consider a forest fire, so many parameters can be changed every second (like the direction of the wind, the intensity of fire) so that any a priory coded adaptivity will fail. Additionally, even someone wants to define adaptivity in design-time, there are situations which cannot be calculated in advance. For example, if during the fighting against the fire a very strong wind starts, the correct action can be calculated only in the moment of the execution, since there are so many parameters that can influence the decision and which will be know in the real-time, like the intensity of the fire, the number of available firemen, the environment, … It means that the system must reason about these events in real time in order to calculate corresponding change in the workflow. This is what we call ad-hoc process changes and corresponding processes event-driven ad-hoc processes. Therefore, any precoding of the possible changes (alternative paths in the process execution) will decrease the flexibility of the process, i.e. the efficiency of the running process instance.

However, these requirements are quite challenging and require a different view on the process flexibility: flexibility in a process is not defined a priory (like in business rules) but it is calculated in the real time, based on the constraints which are defined (mainly) a priory, in the design time. Obviously, it requires a different formalism for representing business processes: one that enables declarative representation and reasoning with constraints.

In this position paper we present such an approach based on the logic.

2. A UNIFYING FRAMEWORK FOR EVENT-DRIVEN AD-HOC PROCESSES: MODELING AND REASONING

Figure 1 depicts the main aspects of event-driven ad-hoc processes addressed in this work. In particular, the figure shows the conceptual relationships between executing tasks, the workflow scheduler, and the dynamic change manager. Tasks that need to be executed are scheduled by the scheduler. The scheduler orders executing tasks according to the model specified by a concrete workflow. The dynamic change manager (DCM) may alter the scheduling plan, i.e., the order in which tasks are scheduled for execution. This may happen due to detection of certain events that represent unexpected situations.

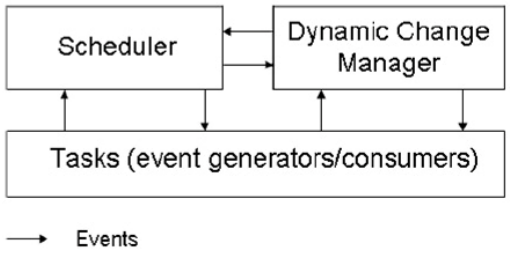

Figure 1. Conceptual architecture of event-driven ad-hoc processes

In the presented conceptual architecture, tasks are seen as external event sources. The scheduler receives a stream of events from these sources, and schedules them in time. The incoming event stream is denoted by the arc pointing from tasks to the scheduler (see Figure 1). As events may represent statuses of executing tasks (e.g., start, end etc.), scheduling an event amounts to scheduling of a task (process). Process scheduling must ensure that scheduling satisfy all constraints, specified by the workflow (possible after reordering some events in the incoming stream). Reordering is realized by sending events from the scheduler back to tasks (depicted by the arc in the reverse direction).
For example, such an event may carry information that a corresponding task is either allowed (for execution), rejected, or delayed. Task events are also gathered by the dynamic change manager, which additionally receives external events (e.g., events from various information sources or sensors etc.). The manager correlates these events into complex events (relevant with respect to a particular business domain). Hence the manager utilises event processing to detect real-time situations that require certain decisions. Decisions may deviate an ongoing workflow instance. They are made by humans, however DCM with its event processing capabilities help in discovering situations (that might require deviations). The deviation (adaptation) is typically driven by the need to take into account new emerging issues (e.g., something accidentally happen) or to optimise the execution with respect to certain events (that just happened). Finally, complex events may further be used externally for e.g., activity monitoring, various analytics etc. (see Figure 1).
In the following we describe main elements:

Control flow graphs. The focus of the following use case is to manage an emergency situation caused by a nuclear plant accident. Let us imagine that due to a critical accident in a nuclear plant, a large quantity of radioactive substance is released in atmosphere.
In such a situation, an emergency response system coordinates with a number activities including emergency responders, ambulances, fire trucks etc.

Events. Events play a few important roles in our framework. We summarise the roles in the following list:
– Control flow graph: workflow tasks are typically modeled in terms of their externally observable events (such as start, end, commit, precommit, abort etc). Such events can be directly incorporated as nodes in a control flow graph; scheduling of the control flow amounts to scheduling of events. (For brevity, our running example collapses all significant events for the same task into one event.)
– Workflow constraints: temporal and causality relationships among workflow tasks are expressed as events. Verification of workflow constraints is performed as the task of proving that a given set of events with their temporal and causality interactions is consistent.
– Complex event patterns: events are used to build more complex event patterns; that may be used for various monitoring or analytical purposes, or initiate ad-hoc changes in a workflow. In this scope, events can represent not only tasks, but external events too (e.g., events from various sensors, other workflows or services etc.).

Our framework for event-driven ad-hoc processes needs to sense for events all the time during its operation. For example, real time events depict the current radiation measurements, weather conditions, traffic information, and the present situation in the decontaminated zone.

Complex events. Complex events represent more meaningful situations of interest.
They help in assessing different situations and making real time decisions

Constraints. Control flow graphs are typically used to represent main activities and their basic dependencies in a workflow. More fine-grained dependencies are specified by constraints; they capture global temporal and causality interactions. Yet another situation when constraints are useful is when specific requirements need to be taken into account (though they may be omitted in other situations).

Ad-hoc changes. Emergency response workflows need to cope with unpredictable changes. Classical workflow management systems offer good process support as long as the processes are structured and do not require much flexibility. However emergency response workflows are expected to be flexible. In practice, it is not feasible to specify all possible cases that may emerge in an incident situation. For example, it may happen that during one emergency situation, another one happens. We can try to structure typical flow of response activities in one incident situation, but not in other ones (if they occur).We assume that ad-hoc changes in workflows are, in major cases, a subject of human's decision; CEP provides just a means to detect real-time situations that possibly require ad-hoc (unpredictable) changes.

With previously described building components (control flow graph, constraints, (complex) events, changes) we define the main problems addressed in this paper:
Complex event processing. Process multiple streams of atomic events with the goal of detecting complex events, according to meaningful event patterns.
Ad-hoc workflow scheduling. Given the fact that tasks are represented as events, decide whether the scheduling of event streams that satisfies both the workflow constraints and ad-hoc changes exists

3. CONCLUSION

Hitherto, approaches to ad-hoc and dynamic process-aware information systems acted on the assumption that decision on process changes are not strictly time sensitive. The emphasis was rather on full support to process modifications. In many practical cases (e.g., emergency management) the time to react on certain situation is limited. Further on, decisions on ad-hoc process modifications need to be carefully assessed taking into account many changing parameters. To address these requirements, we have proposed a framework for event-driven ad-hoc processes. The framework features both event processing capabilities as well as capabilities to accept on-line process changes. The framework is based on declarative rules, and as such it features greater flexibility with ad-hoc changes.