MONITORING WORKFLOWS EXECUTION USING ECA RULES

Giancarlo Tretola and Eugenio Zimeo
Department of Engineering, University of Sannio, 82100, Benevento, Italy

Keywords: Service composition, Adaptation, Monitoring, Event-driven programming.

Abstract: A fundamental requirement for modern large-scale distributed applications is the need to quickly adapt to the fast variations in their operational environment. In SOA the need for adaptation influences design, enactment and execution of service compositions (workflows). To maintain a high quality standard and to support continuous improvements of workflows, the quality of intermediate steps has to be assessed during execution and, if needed, interventions have to be performed at the same time. The key concept is to avoid problems, anticipating the detection of their causes. We propose a monitoring system based on the definition of general management policies that are translated in management rules using an event-driven approach. Rules are employed to react to unexpected events, malfunctions, errors or violations of constraints with the objective of dynamically adapting the process in order to overcome the problems still maintaining a high-quality delivery.

1 INTRODUCTION

Workflow management is becoming an effective approach for distributed application development using the Internet as infrastructure and services composition as programming paradigm for exploiting distributed resources and functionalities, both in Web environment (Natis and Schulte, 1996; Peltz, 2003) and Grid computing (Czajkowski et al., 2001; Foster et al., 2002).
In these contexts, service performance, reliability, responsiveness and other quality related features might change dynamically and quickly. The ability of the process to reach the functional goal and to satisfy QoS constraints must be dynamically adjusted consequently (Buhler and Vidal, 2005).

Workflow paradigm is naturally oriented to analyze the achieved results in order to improve the process definition: Adaptive Workflows (Tretola, 2007; Buthler and Vidal, 2005; Sheth and Verma, 2005). Adaptivity needs agents able to interact with the services and the execution environment for retrieving and analyzing the information of interest in order to identify anomalies. Process anomalies can be classified in three categories: **behavioural**, caused by improper execution of process activities; **semantic**, caused by logically erroneous results after activity execution; **systemic**, caused by malfunctioning of the supporting infrastructure.

In order to intercept anomalies during process execution, monitoring plays a fundamental role. It is intended as a planned or event-driven measurement of one or more properties of a workflow instance in order to identify anomalies. Monitoring is useful to: measure the actual QoS of the provided services for comparison to the declared one; verify the compliance of the running process with constraints and requirements and to detect the deviation from the plan; enable manual and automatic corrective actions able to handle unexpected events; collect and store the information related to all the processes executed, in order to support further analysis aimed at processes improvement; monitor the operational environment in which the process is executed and to enable adaptation to changes.

We propose an approach for measuring and evaluating data, on the way, which may alert the system of incoming problems in order to avoid further complications. The actions that may be undertaken are operation related to the control flow of the process, rescheduling or re-planning. The action may be also related to the binding of a functionality to another performer. To complete the approach it is necessary to underline the necessity to have a semantic definition of the entities involved in the monitoring and the related measurement processes usable for obtaining numerical information about such entities (Giallonardo and Zimeo, 2007; LOCOSP Project).

Tretola G. and Zimeo E. (2009).
MONITORING WORKFLOWS EXECUTION USING ECA RULES.
In *Proceedings of the 4th International Conference on Software and Data Technologies*, pages 423-428
DOI: 10.5220/0002282504230428
Copyright © SciTePress

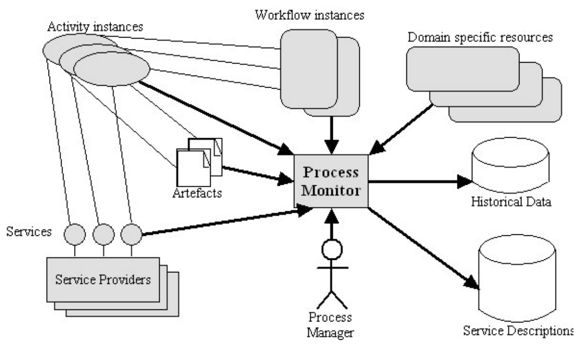

Figure 1: Monitoring system and related components.

The architectural component in charge of process monitoring is the Monitor. It is a component that analyzes the activities of the process and shows real-time information about them, such as status and advances. It is also able to interact with other kinds of sensors or generally measurement tools to retrieve information on the context that may influence the process. The process monitor is also able to access to information describing service semantics and QoS. It also stores the collected data to enable post execution evaluation and to compare the measured QoS with the declared one.

The monitor behavior is based on the definition of general management policies, defined for handling with the problems that may arise during execution. Such policies are translated in management rules, prior to execution, using an event-driven approach. Rules are employed to react to unexpected events, malfunctions, errors or violations of constraints with the objective of dynamically adapting the process in order to overcome the problems still maintaining a high-quality delivery (Tretola, 2007).

2 RELATED WORK

Considering process composed using SOA approach we may consider several works dealing with the problem of monitoring. Such works may be classified in three groups.

Language based. In this approach the monitoring is performed using the built in features offered by the process definition language. It is similar to Exception handling in programming languages. The monitoring policy is embedded in the process definition.

Example: the approach proposed in Dellarocas and Klein that acts before execution, associating to exception processes for detection and avoidance.

Weaving. The monitoring policy is defined independently from the process and its definition language. Before executing the process its monitoring policy is added to the process instance and executed by the work. It is similar to Pre-processing phase in some programming languages or to the pre-execution binding. The monitor policy is bound to the process instance.

Example: Monitoring rules are defined separately and blended with the WS-BPEL process at deployment-time (Green et al., 2000).

Dynamic. The monitoring policy is defined using a specific definition language. Then the policy is provided to specific components and services in the execution system in charge of monitoring process execution. It is similar to Event based systems. The monitor policy is associated to the enactment system.

Examples: in Zeng et al. it is presented the proposal to combine workflow management and agents. responsible for monitoring QoS parameters, introducing flexible variation in the process, using predefined alternatives, and supervising of the service process in execution by the provider. Crossflow (Green et al., 2000) is a dynamic approach to monitoring, performed by components responsible for monitoring QoS parameters, introducing flexible variation in the process, using predefined alternatives, and supervising of the service process in execution by the provider. In Baresi and Guinea the approach is based on managing exceptions through the ECA paradigm, using CHIMERA-EXC language for specifying exception handling, and ECA rules based approach to monitor exception

Event Condition Action rules (**ECA** rules) have been used in many settings, including active databases (Patton, 1999; Widom and Ceri, 1995), for triggering functionality based on data monitoring, and workflow management, specifying and implementing business processes (Bonifati, 2001). They automatically perform actions in response to events provided that stated conditions hold. ECA rules allow for the management of an event, in specified condition, by means of a predetermined action.

They are characterized by three parts: an **Event** defines the event, among the set of observable events, which the rule reacts to; a **Condition** expresses the configuration of the state needed for performing an action; an **Action** describes the activity to be performed if the condition is verified. The advantages of using ECA rules for checkpoint modelling are several.

3 DEFINING THE MONITORING METRICS

Monitoring requires a measurement process in order to retrieve information from running processes. It may retrieve information from: workflow instances, running activities and produced artefacts. The measurement is performed by dedicated measurement tools able to provide the quantitative evaluation of the monitored parameters. In order to enable the retrieving of information, a common description of such resources is needed to the process designers and to the service providers.

An ontology-based model is a possible solution for enabling interactions among system components and to grant future extensibility of the system. The ontology has to provide a common and shared model for allowing measurements of basic parameters and also for derived ones, which are obtained by aggregating different parameters. Properties may be related to: **Internal** properties of the workflow (i.e. activity state, variable value, etc.) or **External** properties (i.e. artefacts or quantity influenced or influencing the process, QoS attributes, SLAs, etc.).

Furthermore, in order to take into account context information it is necessary to perform measurements from the environment of running workflow instance. Measures may be related to two main aspects: **internal environmental conditions** influenced by or influencing the process execution (i.e. the provisions in a warehouse, the latency in a server, etc.) and **external conditions** influencing the goal related to the process ((i.e. the increase of demand or of the price of a product).

In order to describe quality parameters, and more generally non-functional parameters of the monitored resources, an adequate semantic model is needed. A domain independent ontology is used to define conceptual basis of the measurement, then a domain dependent ontology has to be defined for the specific operational environment (LOCOSP Project).

4 MONITORING AND MANAGEMENT SYSTEM

In our vision the Monitoring System is able to act at run-time, measuring and managing the process in execution. The approach proposed is based on the concept of **checkpoint**: a point in the control flow in which the Monitoring System performs some measurements, evaluates the overall state of the executing process, and may perform some actions.

A checkpoint is an additional concept added to the control flow that represents a moment of verification and validation of the process progressing towards the goal regarding the non-functional aspects of the performing activities. They may be associated to a single activity, to a set of activities or to a process instance. A checkpoint can be viewed as a sub-process composed of the sequence of two activities: **monitoring** and **management**.

The monitoring activity has the role to collect the data necessary to asses the situation of the process activity or activities to be monitored. The data retrieved are examined to decide whether a management procedure has to be started. In such case, the appropriate procedure is selected and executed.

Management procedure is related to a specific interface, we have defined, for dynamic adaptation, which allows run-time modification to the control flow of a process instance in execution in the workflow engine.

A checkpoint has an activation criterion, which represents the triggering cause that starts the checkpoint activities. It may be of different types: **synchronous**, i.e. based on a temporal event; **asynchronous, i.e.** based on the occurrence of non-temporal events. A synchronous criterion is further classified as **absolute time**, the specific date and time in which the checkpoint has to be performed (for example the 23rd of March at 15:00); **relative time**, a time period starting from an event related to the process execution (for example after 1 minute from starting of a selected activity). An asynchronous criterion is subdivided in **generic workflow events** (e.g. the start of an activity) and **specific resource events** generated during the activity execution (e.g. the percentage of task completion). Each activation criteria is bound to an event that is generated internally, by the workflow enactment system (first three types), or externally in the service provider system (the fourth type).

The conceptual behaviour of a checkpoint is based on the Event Condition Action paradigm (ECA rules). It enables the separation of the workflow definition from the monitoring policy. Workflow enactment and checkpoint execution are two parallel executing flows. The control flow of the process defines the normal flow of the execution, managed and supervised by the workflow engine. The monitoring flow handles the reaction flow that is responsible to handle the events fired during the execution of the process. Different process instances may be monitored with different policies.

Checkpoints are separated entities associated to the process for the execution; therefore it is possible to change or redefine them for each instance to be enacted. Finally, since a checkpoint is event-based, it is possible to use the same measurement tools and management actions in several situations. It is sufficient to associate them to the appropriate event.

When an event fires, the checkpoint starts the execution: the monitoring process performs the measurement of the interested parameters and decides whether a management action has to be performed. This is performed by evaluating an activation condition, composed of logical (AND and OR) and relational operators (greater than (>), less than (<) and equal (=)). The operands can be of three types: constant values known at design time; local variables defined in the checkpoint context and evaluated during the execution; workflow variables retrieved from the process instance in the workflow engine at execution time. It is worthy to note that more than one checkpoint may be associated to a single event. It is possible to have more than one ECA rule that reacts to the same event, with different conditions to be evaluated and different management actions to be performed.

The Monitoring System is based on two main interacting components: one is the observed system, the workflow engine, and the other is the observer, the Monitor, responsible to supervise the QoS parameters of the process during its execution. After a process is described, checkpoints can be associated to it. In order to allow this, the observed and the observer have to share the definition of the observable parameters, the tools able to provide a measure for them and the procedure to be used to perform corrective actions. The selected model is a shared ontology describing the QoS parameters of the specific domain, obtained by specializing the domain independent ontology. The ontology is used both during the checkpoint definition, for selecting which values have to be used for evaluating the condition, and during the checkpoint execution, for selecting the specific measurement process and for assigning a measure to the entity according to the defined metrics.

The Monitoring System is a general purpose element able to dynamically adapt to parameters to be monitored, using the ontology to act correctly. The components of the systems and their roles are shown in figure 2.

Process Definition Tools. They contribute to the process definition. A prototype implementation enables the definition of a process and related checkpoints.

Engine. It is the component responsible for the process enactment. This is used as an observed system. So it is characterised by an external interface composed of two parts: sensing and effecting. The sensing interface is used to access to the information about the process and activities in execution. The effecting interface is used to interact with the engine for altering the structure of the process during its execution.

Monitor. It represents the observer in the system. Its role is to receive events, to execute the ECA rules bound to the specified checkpoints, and to interact with the engine using the sensing-effecting interface. It is also able to interact with the measurement tools in order to perform some parameter measurements for evaluating the conditions.

Measurement Tools. They are the set of external tools that are able to measure and evaluate the QoS parameters involved in the execution.

Management Tools. They are a set of tools that are able to execute corrective actions on the executing process control flow on the external environment (i.e. Send a notification to a service provider).

Sensing-Effecting Interface. The Sensing interface allows for retrieving information from the engine in order to assess the state of the component. The Effecting interface allows for managing the component.

Rule Engine. The checkpoint description, provided using the XML language, is then translated into a rule engine implemented with Jess (JESS). The rule engine manages the ECA rules.

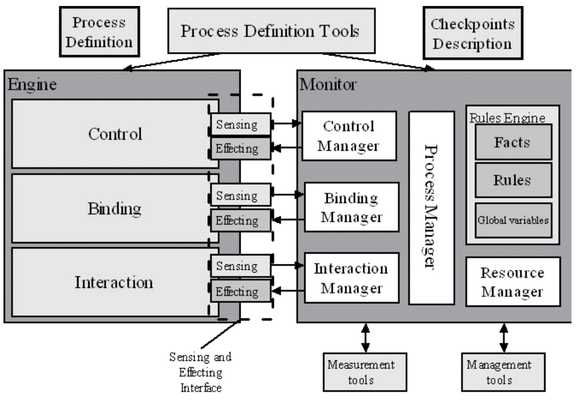

Figure 2: The overall system with the execution Engine and the Monitor.

In order to allow the management, the Workflow engine should allow access to the internal state of the process instances that is in execution. In our experimentation we have defined a set of operations that are needed for allowing such modification.

First of all we have defined and implemented operation for suspending and resuming a process enactment, i.e. the activity are considered atomic and enacted activities are completed, but no more activity is activated for the execution. We considered this policy for taking into account several problems that may arise in adapting a process that usually is based on distributed activities. What happens if a triggering event arrives in the middle of the distributed transaction? In our approach no alteration of the running activities is performed, the enactment is suspended and no other activity execution is started.

While suspended a process may be dynamically modified using other operation we have introduced. It is possible to add an activity in the control flow, updating the transitions in the process graph. It is also possible to remove an activity from the control flow also in this case updating the transitions in the control flow, removing the unnecessary ones. We have also implemented operations that allows for altering the activities start-line and deadline in order to perform a re-scheduling of the activities enactment without changing the process control flow. We are considering also other operation: for example re binding to perform a different selection of another resource to execute an activity. Other operations are under evaluation, for example re-planning of the single instance or re-planning of the original process description and so on.

The ECA rules are managed by a Knowledge Base or an Agent. In our test we used a JESS engine, in which fact and rules are asserted basing on two categories. The first one is the generic rules that are applicable to any process because are related to events that may happen in each workflow, i.e. a service not reachable. To the second categories belong the rules that are specific of the particular workflow and of its operative environment. The use of a knowledge base is interesting because it may works in two ways. One is based on reaction of the engine to variation of the knowledge base, for example assertion of a new fact, in order to check if there are active rules. This may be used to perform reaction to event received by the manager. Another way of work for the engine is a backward reasoning, in which a rule is analyzed in order to check if it is active given the current facts and rules asserted in the knowledge base.

5 EXAMPLES

In this section, we present a first example related to a context monitoring scenario. In this case the workflow process is related to management of activities in a vineyard. The parameters under measurement are environmental parameters: temperature, humidity, wind velocity, luminosity etc. The environmental information may be used to reschedule activity in the vineyard, for example repeating a treatment against parasites if "humidity and the temperature are greater than known threshold".

```
CHECKPOINT: Parasite Avoidance
ON EVENT: Temperature
Variation(Context.Temperature)
IF CONDITION: (temperature > 30) &&
(measured humidity > 0.75)
PERFORM ACTION: ADD
AntiParasiteTreatementActivity
```

Another action we are considering in our virtual experimentation is the using of heater during bad season to decrease damage from frost.

```
CHECKPOINT: Frost
ON EVENT: Temperature
Variation(Context.Temperature)
IF CONDITION: (temperature > 5) &&
(measured wind < 0.5)
PERFORM ACTION: ADD
VineyardHeatingActivity
```

This is an example of a context aware application in which there is no need to specify in the process all the possible intervention in each possible moment because the system reacts to the dangerous situation modifying the process accordingly.

6 CONCLUSIONS

Nowadays, distributed applications, implemented using workflow management, place a great challenge. The large availability of services allows for the possibility to select from a wide offer of functionalities. Such functionalities are provided by different organization and with different levels of performance. The monitoring is then a fundamental phase in the life cycle of workflow, for allowing the possibility of fine tuning in the selection of services. Collecting the measurement of the actual QoS of invoked services allows for a greater confidence in subsequent selection operations. Moreover, monitoring provides a lot of information at run-time that may be used to prevent problems. The detection of a non conformity in QoS of a service that is in use in a workflow may be dealt immediately, without waiting consequent problems. Problems in the interaction may be resolved without waiting errors blocking the systems. The management action may involve a rebinding of the activity to another service, the rescheduling of activities in the process or the planning of new activities to be executed to handle the new situation. As we have seen activity and process related values are not the only data that may be used. Context measurement is also really useful in the process management. They allow for retrieving information, external to the specific activities in execution that may alter the operational environment, demanding for adaptation of the process control flow, or even change the overall goal of the process. Dynamic monitoring enables the execution of process that otherwise may be not described if not using complex syntax and detailed description of each anomalies to be handled in the control flow.

Our experimental work is related to SAWE (SAWE) a Workflow Enactment System designed to be autonomic and adaptive, able to execute process using Web Services, Java local or remote objects, and Grid resources. In our work we have defined a conceptual checkpoint model and used an XML extension to model it.

Future activities are related to the realization of a higher level language for the definition of monitoring policy coupled with the definition of an interface for dynamically managing the process execution. The objective is to make totally automated the monitoring and the management.

ACKNOWLEDGEMENTS

The research activities of this work are framed in the research project ArtDeco, funded by MIUR.

REFERENCES

Tretola, G., 2007. Autonomic Workflow Management in e-Collaboration Environment. Ph.D. Thesis.

Paton, N., 1999. Active Rules in Database Systems. *Springer-Verlag*.

Widom, J., Ceri, S., 1995. Active Database Systems. *Morgan-Kaufmann*, San Mateo, California.

Bonifati, A., Ceri, S., Paraboschi, S., 2001. Pushing reactive services to XML repositories using active rules. In Proc. 10th World-Wide-Web Conference.

Natis Y.V., Schulte, R.W., 1996. *Service Oriented Architectures,* Part 1.

Peltz, C., 2003. Web Services Orchestration and Choreography. *Computer*, vol. 36, pp. 46-52.

Giallonardo, E., Zimeo, E., 2007. More Semantic in QoS Matching. *In IEEE International Conference on Service-Oriented Computing and Applications*, pp. 163-171.

Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C., 2001. Grid Information Services for Distributed Resource Sharing. *In: 10th IEEE International Symposium on High Performance Distributed Computing*, pp. 181-194.

Foster, I., Kesselman, C., Nick, J., Tuecke, S., 2002. The Physiology of the Grid: an Open Grid Services Architecture for Distributed Systems Integration. *Technical report, Global Grid Forum*.

Buhler, P. A., Vidal, J. M., 2005. Towards Adaptive Workflow Enactment Using Multiagent Systems. *Information Technology and Management 6*, pp. 61-87.

Sheth, A.P., Verma, K., 2005. Autonomic Web Processes. *Proceedings of the Third Conference on Service Oriented Computing, International Conference on Service Oriented Computing*.

Dellarocas, C., Klein, M., 2000. A Knowledge-based Approach for Handling Exceptions in Business Processes. *Information Technology and Management 1*, pp. 155-169.

Zeng, L., Ngu, A., Benatallah, B, 2001. An Agent Based Approach For Supporting Cross-Enterprise Workflows. *ADC,* pp. 123-130.

Green, P., Aberer, K., Ludwig, H., Hoffner, Y., 2000. Crossflow: Cross-Organizational workflow Management For Service Outsourcing In Dynamic Virtual Enterprises. *IEEE Data Eng. Bull*. 24(1), pp. 52-57.

Baresi, L., Guinea, S., 2005. Towards Dynamic Monitoring of WS-BPEL Processes. *International Conference on Service Oriented Computing*, pp. 269-282.

Casati, F., Ceri, S., Paraboschi, S., Pozzi, G., 1999. Specification and Implementation of Exceptions in Workflow Management Systems, *ACM Trans. Database Syst*. 24(3), pp. 405-451

LOCOSP Project, http://plone.rcost.unisannio.it/locosp

SAWE. http://www.gridworkflow.org/snips/gridworkflow/space/SAWE

JESS. http://herzberg.ca.sandia.gov/jess/