TERRA-ACQUA, adaptable definition and execution of workflows

Genoveva Vargas-Solar, Khalid Belhajjame
IMAG-LSR, University of Grenoble
BP 72 38402 Saint-Martin d'Hères, France.
{Genoveva.Vargas, Khalid.Belhajjame}@imag.fr

Esaú E. Castillo Contreras, Karla Joana Peredo Márquez
Centro de Investigación en Tecnologías de Información, UDLAP
Ex Hacienda Sta. Catarina Mártir s/n, San Andrés Cholula, Puebla, Mexico
{is106341, is106967}@mail.udlap.mx

Abstract

This paper presents TERRA *and* ACQUA [1] *two mechanisms that support adaptable definition and execution of workflows.* TERRA *generates and stores structures (objects, events, reactions) necessary for executing a workflow.* AQUA *executes workflows according to a parametric behaviour model that represents workflow execution policies in terms of dimensions associated to values.* TERRA *and* ACQUA *provide mechanisms for modifying the structure of a workflow and its associated behaviour.*

1 Introduction

Today most applications are cooperative. Workflow technology provides useful solutions for specifying and implementing applications that enable automatic execution of business processes such as virtual enterprises, medical applications, control and e-commerce. A business process consists of activities with a common objective. They are executed by agents (human or software components). For example, consider an e-commerce application. To choose items, clients first access a catalogue (activity AccessCatalogue) that describes products (e.g., characteristics, properties, and price). Then, an order (ProcessOrder)is specified. If chosen products are available, in parallel, the payment process is started (Authorization and Billing), a package is prepared (PreparePackage) and the stock is updated (StockUpdate). Finally, the package is delivered to the client (Deliver).

A workflow is the computerized representation of a business process. Not only it describes its activities but also their execution order (i.e., sequential, in parallel) and data types that they exchange. It specifies the flow of data between activities, for instance the account number, the catalogue and chosen products are data that flow among activities, and the collaborating agents responsible for the activities execution.

Recently, flexibility and adaptability requirements for defining and executing business processes have increased. For example, add the activity ProposeBargains to be executed while the activity Payment is executed; or change input data of the activity ProcessOrder. These requirements underline the need of mechanisms that support adaptable and flexible definition and execution of workflows.

Workflow technology has evolved and provides models, systems [10, 3, 11, 9, 13, 6, 1] and techniques that enable workflow evolution and reuse [16, 12, 15, 14, 4]. From our point of view, the first step needed to introduce adaptability to workflows is the orthogonal specification of definition and execution aspects. Our work contributes with mechanisms that enable adaptable, flexible and distributed execution of business processes using workflows.

This paper presents the implementation experience of TERRA and ACQUA, two mechanisms that respectively support adaptable definition and execution of workflows. Accordingly, the remainder of the paper is organized as follows. Sections 2 and 3 respectively describe the architecture and main functions of TERRA and ACQUA and discuss implementation issues. Section 4 illustrates the use of both systems through a validation experience. Finally, Section 5 concludes the paper discussing our main contributions, the originality of our work and research perspectives.

[1] The Results obtained in this work belong to the Database group NODS of Laboratory LSR-IMAG, France. The implementation experience is conducted under the direction of the NODS group and with partial resources of the Database Technology Group, CENTIA-FUDLAP, which only has use rights.

2 TERRA

General architecture

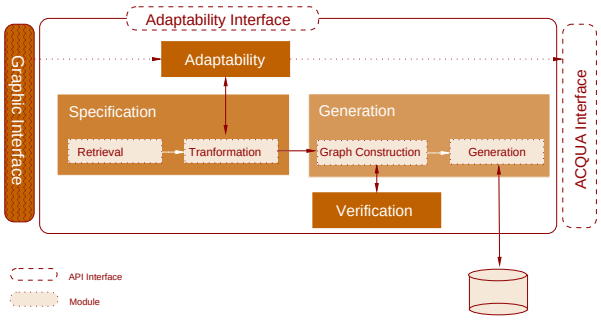

Figure 1. Architecture of TERRA

Figure 1 illustrates the architecture of TERRA. It is composed of Three main modules implementing workflow specification, verification and generation. The Specification module provides an interface with formularies for specifying workflow elements according to the knowledge model [2] implemented by the system. Based on a specification, the module creates a graph that represents the specified workflow structure. The Verification module verifies workflow structure correctness and termination. The Generation module generates executable workflows where activities are objects that implement method calls corresponding to agent interfaces; activities dependencies and communication with agents are materialized by events. The module creates a workflow schema (in the database sense) and stores it in a OODBMS (FastObjects).

Main functions

Workflow definition in TERRA is done in four steps: specification, verification, generation and adaptability. The first three steps are sequential, while the last one is orthogonal and causes the execution of the verification and generation steps.

Specification In this step a workflow is specified as a collection of ordered and synchronized activities. Workflow specification is based on knowledge model [2] that provides concepts for representing workflows. According to the model, an activity can represent a simple task such as the access to a data base; or a composite task such as determining whether a credit can be authorized. The order associated to activities is expressed with ordering operators (e.g., seq, par, and(or)-join, and(or)-split); conditions under which an activity can be executed are expressed by pre and post conditions and invariants; activities synchronization are expressed as temporal constraints. The model provides also structures for representing data flow among activities.

Verification It is possible to introduce inconsistencies while specifying a workflow. Verification process aims at detecting such inconsistencies before putting the process into execution. In particular, we consider two types of inconsistencies, deadlock and multiple terminations. The verification process adopted is inspired on [17] and it mainly consists in determining whether operators and-split (resp. or-split) and and-join (resp. or-join), are balanced. Intuitively, structural verification of a workflow is done on a graph that represents it. A recursive analysis is applied on such a graph for verifying whether every and-split (resp. or-split) structure is followed by an and-join (resp. or-join). If this condition does not hold, an exception is raised.

Generation The objective of this step is to generate and store structures needed for executing a workflow including JAVA objects that implement activity execution and events that implement communication between a workflow execution engine and agents.

Structural adaptability This step is independent of the previous and its objective is to process structural changes of workflows. For example, add or delete an activity, modify the order and/or synchronization among activities, modify data flow and the association of agent types to activities. Given an adaptability operation, the systems triggers the verification and generation processes. Finally, a modification results in the specification or the reconfiguration of a new workflow.

3 ACQUA

General architecture

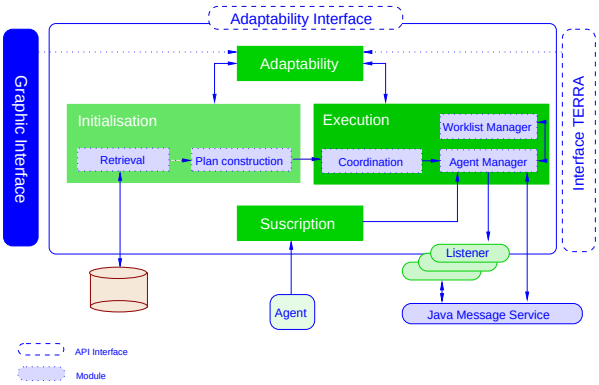

Figure 2. Architecture of Acqua

The architecture of ACQUA is depicted in Figure 2. It is composed of four main modules described as follows. The system provides a user interface for retrieving workflows, storing modified workflows and observe their execution. The initialization module generates an execution plan (Transformation sub module) based on a workflow specification retrieved by the Retrieval sub module. Execution plans are also stored in a database managed by the OODBMS (FastObjects). The subscription module receives agent connection requests and manages associated information such as their types and location. The Execution module coordinates the execution plan, manages subscribed agents (i.e., subscribe, choose) and their associated worklists (worklist manager sub module). The Adaptability module processes modification requests on policies associated to workflow behaviour.

Main functions

The execution process consists of three phases: execution plan generation, agent subscription and execution.

Execution plan construction First, the workflow specification is retrieved from FastObjects: objects implementing the workflow activities, events implementing the flow (order and synchronization operators) and policies associated to each activity and to the whole workflow. The execution plan is implemented by a graph whose root is the workflow name and the nodes can be activity nodes (terminal nodes) and operator nodes (non terminal nodes) that have also information on execution policies.

Agent subscription ACQUA assumes that agents are autonomous and that they can be connected or disconnected dynamically. For each subscribed agent, the system generates a Listener that acts as a proxy between an agent and the Agent manager of the execution module. Each agent and its associated Listener are subscribed respectively as event producers and consumers in the event service.

Execution The execution starts as a result of an explicit request from the user. First the workflow definition (execution plan) is retrieved from the specification database. Then the coordinator instantiates and configures the necessary operators. Figure 3 illustrates the configuration that has been built to handle the execution of the purchase application presented in Section 1. Operators communicate among them using asynchronous event and with agents using proxies. They have execution parameters that are configured according to the behavior model.

When an activity is triggered, the operator contacts the agent manager that chooses a subscribed agent according to *assignment policies* associated to the activity. If an agent is available the Agent Manager assigns the activity and the Worklist Manager inserts it in the agent worklist. Otherwise it notifies the coresponding operator node that decides what to do according to *scheduling policies* associated to the activity. When an agent terminates the execution of an activity it sends a notification to the associated operator through its proxy. The operator then evaluates the activity postcondition, activates the triggered activities, and passes the control to the next ordering operator.

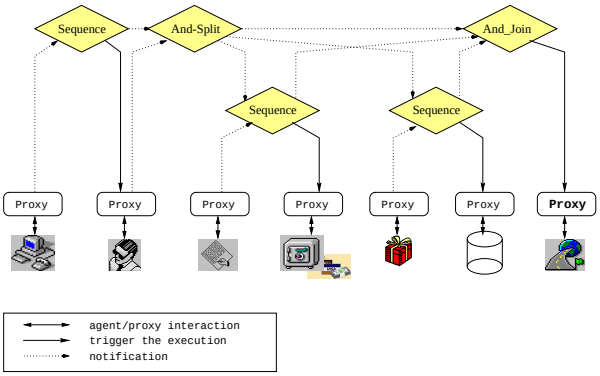

Figure 3. Workflow execution

Behavioral adaptability This phase is independent of the previous ones and its objective is to enable changes on execution policies associated to workflows behaviour. A modification request can be done by TERRA or by a programmer. Given an adaptability request related to a workflow, ACQUA analyzes its "current" execution state to determine whether such a change can be immediate or deferred. According to the modified policy, immediate changes are done on the execution plan (policies associated with activities and the whole workflow) or on policies implemented by the Agent Manager.

4 Experimentation

We are currently conducting an experimentation with TERRA and ACQUA for defining e-commerce processes. In the following we present examples related with the e-commerce application introduced in Section 1.

Specification

Each activity in a workflow is defined by its name, precondition, post condition and invariant, and by its associated data. TERRA provides an interface for specifying activities and their associated behaviour by choosing predefined values for a given set of parameters. Order and synchronization modes among activities are expressed with order (sequence, or/and-join, or/and-split) and temporal restrictions.

Agents organization associated to a workflow is specified by defining, for each agent, its name, position in an organization and role. For example, we can define an agent named 148.149.5.9 with position Bank Server and with roles Authorization and Billing.

Verification

The generation of the workflow ElectronicPurchase results in a graph representing its structure (WorkflowGraph). It is an oriented graph whose nodes are either sub-processes or ordering operators, or activities. The root node represents the workflow process. Such a graph is used for verifying the workflow structure using the algorithm proposed in [7, 8].

Generation

For each non terminal node (operator node) in the WorkflowGraph TERRA generates event types and JAVA objects that implement terminal nodes (activity nodes). Using ordering and synchronization information, TERRA generates events of the form:

$\varepsilon\tau_1$ = ⟨ type name: string, {⟨ variable name:{real, integer, string}⟩}⟩

For example, the event E_1 is produced at the end of the execution of the activity ProcessOrder, and has as a context the object representing the activity ProcessOrder and the instant of its termination.

E_1= EndActivity, {[activity1: Activity, instant: real]}
where activity.name = 'ProcessOrder'

5 Conclusions and future work

This paper presented an approach for defining and executing workflow in an adaptable way. In order to provide adaptability, similarly to DBMS of the third generation, we adopt a separation principle of the definition and execution aspects. In this way, a workflow can have different agent models and can be executed with different execution policies. In order to validate our hypothesis we specified and implemented TERRA and ACQUA two independent infrastructures that cooperate for managing workflows. The distinctive characteristic of both systems is that they support dynamic structural and behavioural adaptability.

Research perspectives concerning TERRA and ACQUA is related to services integration. We are currently conducting an experience for integrating e-commerce and data base services as part of the NODS [5] project.

References

[1] G. Alonso, G. Mohan, C. Guenthoer, R. Agrawal, A. E. Abbadi, and A. Kamath. Exotica/FMQM: A Persistent Message-Based Architecture for Distributed Workflow Management. In *Working Conference on Information Systems for Decentralized Organizations, Trondheim*, 1995.

[2] K. Belhajjame, G. Vargas-Solar, and C. Collet. A flexible workflow model for process-oriented applications. In *The 2nd International conference on Web Information Systems Engineering, WISE'2001*, Kyoto-Japan, December 2001. IEEE.

[3] F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Semantic Workflow Interoperability. In *Advances in Database Technology - EDBT'96, 5th International Conference on Extending Database Technology, Avignon, France, Proceedings*, volume 1057 of *Lecture Notes in Computer Science*. Springer, 1996.

[4] F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Workflow Evolution. *Data & Knowledge Engineering*, 24(3), 1998.

[5] C. Collet. The NODS Project: Networked Open Database Services. In *Proc. of the 14th European Conference on Object-Oriented Programming (ECOOP 2000)- Symposium on Objects and Databases*, Cannes, France, June 2000. http://www-lsr.imag.fr/Les.Groupes/storm/NODS/index.html.

[6] S. Dami, J. Estublier, and M. Amiour. APEL: a Graphical Yet Executable Formalism for process Modeling. *Automated Software Enginnering (ASE)*, D(2.9), 1997.

[7] W. V. der Aalst. Verification of workflow nets. In *Application and Theory of Petri Nets*, Berlin, 1997. Lecture Notes in Computer Science-Springer Verlag.

[8] W. V. der Aalst. Workflow verification: Finding control-flow errors using petri-net-based techniques. *Business Process Management*, 2000.

[9] A. Geppert, M. Kradolfer, and D. Tombros. EVE, Event Engine for Workflow Enactment in Distributed Environments. Technical Report 96.05, University of Zurich - Switzerland, Department of Computer Science, 1996.

[10] D. Hollingsworth. The Workflow Reference Model. http://www.aiim.org/wfmc/, 1995.

[11] G. Kappel, S. Rausch-Schott, and W. Resschitzegger. Workflow Management Frameworks. *Wiley & Sons*, 1998.

[12] M. Kradolfer. *A Workflow Metamodel Supporting Dynamic, Reuse-based Model Evolution* . PhD thesis, Department of Information Technology, University of Zurich, Switzerland, mai 2000.

[13] O. Mylopoulos, A. Gal, K. Kontogiannis, and M. Stanley. A Generic Integration Architecture for Cooperative Information Systems. In *Proceedings of the First IFCIS International Conference on Cooperative Information Systems (CoopIS'96)*, 1996.

[14] M. Reichert and P. Dadam. ADEPT flex -supporting dynamic changes of workflows without losing control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.

[15] G. Sánchez-Gutiérrez. The WIDE Workflow Model and Language. Technical report, WIDE, 1997. 2^{eme} version.

[16] D. Tombros. *An Event- and Repository-Based Component Framework for Workflow System Architecture*. PhD thesis, Department of Information Technology, University of Zurich, Switzerland, november 1999.

[17] W. M. P. van der Aalst. Workflow verification: Finding control-flow errors using petri-net-based techniques. *Lecture Notes in Computer Science*, 1806, 2000.