Enabling Poka-Yoke Workflows with the AristaFlow BPM Suite

Manfred Reichert[1], Peter Dadam[1], Stefanie Rinderle-Ma[1], Andreas Lanz[1], Rüdiger Pryss[1], Michael Predeschly[1], Jens Kolb[1], Linh Thao Ly, Martin Jurisch[2], Ulrich Kreher[2], and Kevin Goeser[2]

[1]Institute of Databases and Information Systems, University of Ulm, Germany
[2]AristaFlow GmbH, Ulm, Germany

Abstract. This tool presentation gives insights into the AristaFlow BPM Suite – a next generation process management technology that has originated from our research activities in the ADEPT project. We show how AristaFlow ensures ease of use for process implementers, application developers, and end users. Our overall vision is to realize robust and flexible process support without encountering bad surprises during runtime. We denote such error-safe process executions as *Poka-Yoke Workflows*.

1 Introduction

During the last decade we have developed the ADEPT process management technology [1]. Due to the high interest of companies in ADEPT, in 2008, we founded a spin-off to transfer ADEPT into an industrial-strength product version called *AristaFlow BPM Suite* [2]. One of our basic goals is to enable robust and flexible process-aware information systems (PAIS). In particular, we want to ensure error-safe process execution even at the presence of exceptions or dynamic process changes. We denote such error-safe processes as *Poka-Yoke Workflows*.

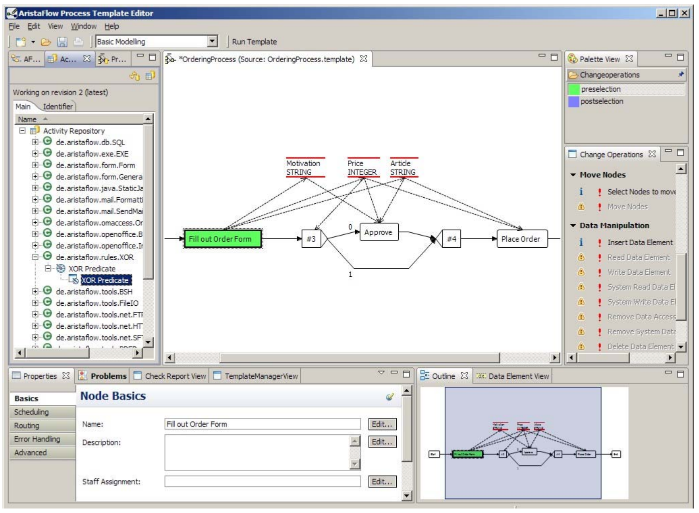
In our tool demo we focus on a fundamental pillar of Poka-Yoke Workflows: *ease of use*. Ease of use does not only have to be ensured for end users (i.e., process participants), but should be provided to process implementers and application developers as well. In order to enable Poka-Yoke Workflows we had to develop a technology which supports "correctness by construction" during process composition and which guarantees correctness in the context of dynamic process changes. This was probably the most influential challenge for our research activities. It also had significant impact on the development of the AristaFlow BPM Suite. In particular, we hide the inherent complexity of process-orientation (especially in conjunction with flexibility) as far as possible from system administrators and application programmers; i.e, we perform all complex things "beneath the surface" in the process management system.

2 Ease of Use Aspects

In the following we describe ease of use aspects in relation to three user groups: process implementers, application developers and end users. Our tool demonstration gives detailed insights into these different aspects.

2.1 Ease of use for process implementers

For implementing Poka-Yoke Workflows, we pursue the idea of process composition in a *"plug & play"* style supported by correctness checks. The latter contribute to exclude errors during process execution. As prerequisite, for example, implicit data flow dependencies among application services, which are relevant for their execution order, have to be made known to the process engine. AristaFlow provides an intuitive graphical editor and composition tool to process implementers (cf. Fig. 1), and it applies a *correctness by construction* principle by providing at any time only those operations to the user which allow to transform a structurally sound process schema into another one; i.e., change operations are enabled or disabled according to which region in the process graph is marked for applying an operation. Deficiencies not prohibited by this approach (e.g., concerning data flow) are checked on-the-fly and are reported continuously in the problem window of the *Process Template Editor*.



Working on revision 2 (latest)

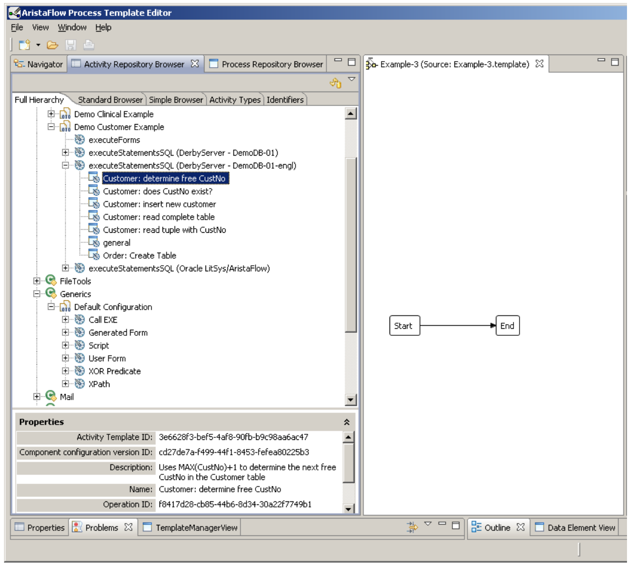
Name: Fill out Order Form

Description:

Staff Assignment:

Fig. 1. AristaFlow Process Template Editor

Generally, we should not require from process implementers that they have detailed knowledge about the internals of the application functions they can assign to process activities. However, this should not be achieved by undermining the *correctness by construction* principle of Poka-Yoke Workflows. In AristaFlow, all kinds of executables, that may be associated with process activities, are first registered in the *Activity Repository* as activity templates. An activity template provides all information to the *Process Template Editor*; e.g., about mandatory and optional input/output parameters or data dependencies to other activity templates. The process implementer just drags and drops an activity template from the *Activity Repository Browser* window of the *Process Template Editor* (cf. Fig. 2) onto the desired location in the process graph. Depending on the intended purpose of usage, an activity template can be rather specific or generic. When using a specific template everything can be fixed; e.g., input/output parameters and all settings. In this case, the only remaining task for the process implementer is to check whether the proposed mapping of parameters to process data elements is correct. When using a specific database activity template, for example, input/output parameters, details of the used database, connection parameters, and the fully specified SQL statement are fixed. A more generic Activity Template, in turn, may leave open the SQL statement or the number and types of input/output parameters.


AristaFlow Process Template Editor
File View Window Help
Navigator
Activity Repository Browser
Process Repository Browser
Example-3 (Source: Example-3.template)
Full Hierarchy
Standard Browser
Simple Browser
Activity Types
Identifiers
Demo Clinical Example
Demo Customer Example
executeForms
executeStatementsSQL (DerbyServer - DemoDB-01)
executeStatementsSQL (DerbyServer - DemoDB-01-engl)
Customer: determine free CustNo
Customer: does CustNo exist?
Customer: insert new customer
Customer: read complete table
Customer: read tuple with CustNo
general
Order: Create Table
executeStatementsSQL (Oracle LitSys/AristaFlow)
FileTools
Generics
Default Configuration
Call EXE
Generated Form
Script
User Form
XOR Predicate
XPath
Mail
Start
End
Properties
Problems
TemplateManagerView
Outline
Data Element View

Properties

Activity Template ID:	3e6628f3-bef5-4af8-90fb-b9c98aa6ac47
Component configuration version ID:	cd27de7a-f499-44f1-8453-fefea80225b3
Description:	Uses MAX(CustNo)+1 to determine the next free CustNo in the Customer table
Name:	Customer: determine free CustNo
Operation ID:	f8417d28-cb85-44b6-8d34-30a22f7749b1

Fig. 2. Activity Repository Browser window in the Process Template Editor

2.2 Ease of use for application developers

A developer who wants to provide a new application function must implement a corresponding activity template and add it to the *Activity Repository*. It then becomes available in the *AristaFlow Process Template Editor* for process modeling (cf. Fig. 2). To simplify implementation of activity templates, we support several levels of abstraction. At the lowest one we provide an *Execution Environment* for each kind of supported basic operation (e.g., SQL statements, web services, EXE files, BeanShell scripts, basic file operations, and system-generated forms). However, implementation of an execution environment requires some knowledge about internals and, therefore, is typically not the task of an ordinary application developer, but is performed by system implementers.

An execution environment defines the set of methods needed to interact with the runtime system as well as to implement the operations and facilities that shall be provided by the activity template. An activity template for database access, for example, may allow the user to specify connection details. In general, AristaFlow needs information about the runtime behavior of the activities; e.g., whether they may be aborted, suspended, or undone. The developer of an activity template has to implement interface methods that inform the AristaFlow runtime environment which of these facilities are supported. For the latter case he must also provide the implementation of this functionality. The task of implementing a new activity template is simple if it can be based on a generic activity template. In this case, implementation can be reduced to putting the appropriate entries into the set of forms representing the activity template.

2.3 Ease of use for end users

Enabling ease of use for end users is mainly the task of application developers. They decide how "manual" process activities interact with the end user. They also decide whether the standard workflow client is used or whether a dedicated one shall be provided. An important prerequisite for realizing adapted user interfaces is to provide the appropriate methods to the application developer; e.g., to enable him to realize end user interactions in the context of ad-hoc changes. To implement clients with such capabilities, the application developer can make use of system functions available at the AristaFlow API.

Fig. 3 and Fig. 4 illustrate how an ad hoc change may foster Poka-Yoke Workflows. Assume that a process instance wants to issue a request for a book quote using Amazon's web service facilities, but then fails in doing so. The user detects that his process is in trouble and calls the system administrator. The latter then invokes the *AristaFlow Process Monitor* to take a look at this process instance (cf. Fig. 3). Analyzing the execution log of the failed activity he detects that its execution failed because the connection to Amazon could not be established. Let us assume that he considers this as temporary problem and offers the user to reset the activity so that it can be repeated once again. Being a friendly guy, he takes a short look at the process instance and its data dependencies, and sees that the result of this and the subsequent activity is only needed when executing the "Choose offer" activity. Therefore, he offers the user to move these two activities after activity "CheckSpecialOffers"; i.e., the user can continue to work on this process instance before the PAIS tries to reconnect to Amazon. To accomplish this change he would switch to the *Instance Change Perspective* of the *Process Monitor* which provides the same set of change operations as the *Process Template Editor*. In fact, it $\underline{\text{is}}$ the *Process Template Editor*, but it is aware that a process instance has been loaded and, therefore, all instance-related state information is taken additionally into account when enabling/disabling change operations and applying correctness checks. The system administrator would now mark the nodes "Get Amazon offer" and "Get Amazon price" as source area and the nodes "CheckSpecial Offer" and "Choose offer" as target area, and then perform operation `Move nodes`. The resulting process is depicted in Fig. 4. Another option would be to move node "RetrieveSnailOffer" (where we are waiting for an E-Mail response) after "CheckSpecialOffer" as well. Then "CheckSpecialOffer" would become immediately executable.

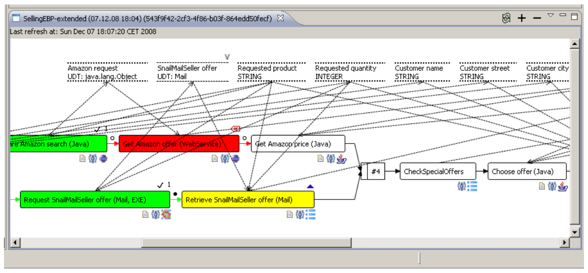
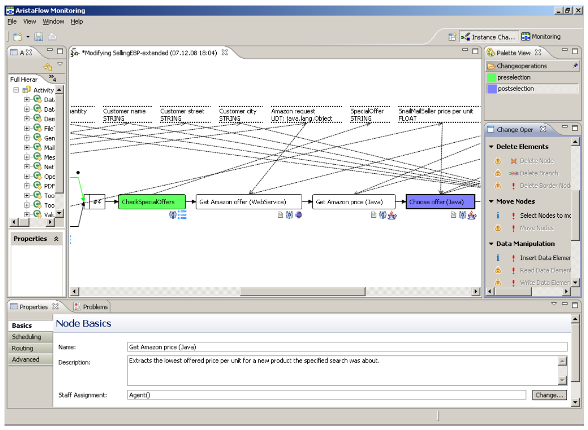

SellingEBP-extended (07.12.08 18:04) (543f9f42-2cf3-4f86-b03f-864edd50fecf)
Last refresh at: Sun Dec 07 18:07:20 CET 2008
Amazon request UDT: java.lang.Object
SnailMailSeller offer UDT: Mail
Requested product STRING
Requested quantity INTEGER
Customer name STRING
Customer street STRING
Customer city STRING
Amazon search (Java)
Get Amazon offer (WebService)
Get Amazon price (Java)
#4
CheckSpecialOffers
Choose offer (Java)
Request SnailMailSeller offer (Mail, EXE)
Retrieve SnailMailSeller offer (Mail)

Fig. 3. Process Monitor: Monitoring Perspective



Node Basics

Name: Get Amazon price (Java)

Description: Extracts the lowest offered price per unit for a new product the specified search was about.

Staff Assignment: Agent()

Fig. 4. Process Monitor: Instance Change Perspective

3 Conclusions

Due to its "correctness by construction" principle and its ability to quickly compose robustly executable and adaptable processes, AristaFlow is suited to enable Poka-Yoke (= error-safe) Workflows. Currently, we apply AristaFlow in different industrial context with very positive feedback on its components and features.

References

1. Reichert, M., Rinderle-Ma, S., Dadam, P.: Flexibility in process-aware information systems. LNCS Transactions on Petri Nets and Other Models of Concurrency (ToPNoC) **2** (2009) 115–135
2. Dadam, P., Reichert, M.: The ADEPT project: A decade of research and development for robust and flexible process support - challenges and achievements. Computer Science - Research and Development **22** (2009) 81–97