Generic Data Access in XML Based Workflow Management Systems

Christian Dreier[1], Johann Eder[2], Marek Lehmann[2], and Amirreza Tahamtan[2]

[1] Alps-Adria University of Klagenfurt
Department of Informatics-Systems
cdreier@edu.uni-klu.ac.at
[2] University of Vienna
Department of Knowledge and Business Engineering
{first_name.last_name}@univie.ac.at

Abstract. Various kinds of data are processed in workflow management systems: from case data to control data, from internal data to access to external databases or documents exchanged in inter-organizational workflows. We propose a uniform treatment of all kinds of business data in workflows. This is achieved by an abstraction mechanism which enables the transparent access to data in any source in a uniform way. We describe an implementation of our extendible generic data access plug-in which provides the workflow management system with updateable XML views of relational data.

1 Introduction

Workflow processes may involve different documents like orders, invoices etc. All these documents represent business data. Each workflow management system (WfMS) [1] must be able to handle these data, which may come from many different sources. These data are used in two different ways. First, they are required by individual activities. Second, the WfMS uses data to make automatically the control flow decisions based on data values. Clearly, workflow management would be not possible without data. It is perhaps surprising, that the data perspective in workflow management was usually left in background [4].

Activities in a workflow process frequently manipulate data stored in the local system environment. Currently, most of the integration work with environmental databases is done manually and requires comprehension of both process model and data model. Therefore, most of the activity programming is related to accessing external data sources.

We propose to separate the process logic and the data access mechanisms by introducing so called data access plug-ins. A data access plug-in is a reusable and interchangeable wrapper around environmental databases that presents to the WfMS their content as XML data and manages access to these databases. This paper is the continuation of our work on data access in WfMSs [7].

The importance of XML technology is increasing tremendously in the workflow management. Workflow management systems [13], B2B standards [9], and

Web services [2] use XML as a data format. Complex XML documents published and exchanged by business processes are usually defined with XML Schema types. Frequently, hierarchical XML data used by processes and activities has to be translated into flat relational data model used by environmental databases.

Section 2 presents the idea of generic data access plug-in, whose implementation is described in Sec. 3. We draw some conclusions in Sec. 4.

2 Generic Data Access Plug-ins

Consider the following frequent scenario: an enterprise has a large database with the customer data stored in several relations and used in many processes. In our approach the company defines a complex XML Schema type describing customer data and implements a data access plug-in which wraps this database and retrieves and stores customer data in XML format. This has several advantages:

- Business data from external systems are accessible by the WfMS. Thus, these data can be passed to activities and used to make control flow decisions.
- Activities can be parameterized with XML documents of predefined types. The logic for accessing external data sources is hidden in a data access plug-in fetching documents passed to activities at runtime. This allows activities to be truly reusable and independent of physical data location.
- Making external data access explicit with the data access plug-ins rather than hiding it in the activities improves the understandability, maintainability and auditability of process definitions.
- Both data access plug-ins and XML Schema type are reusable.
- This solution is easily evolvable. If the customer data have to be moved to a different database, it is sufficient to use another data access plug plug-in. The process definition and activities remain basically unchanged.

A proposal we described in [7] required data access plug-ins to be defined each time from scratch. In this paper we propose a generic data access plug-in (GDAP) which offers basic operations and can be extended by users to their specific data.

The task of GDAP is to translate the operations on XML documents to the underlying databases. GDAP exposes to the WFMS a simple interface which allows XML documents to be read, written or created. Moreover, GDAP allows an XPath expression to be evaluated in order to enable data based process routing. As the relational and object-relational databases are most widely used, our GDAP is intended to query and update complex XML document containing data from one or many relational tables. Thus, documents produced by GDAP can be seen as XML views of relational data. A user can use the basic functionality offered by GDAP and adapt it to the underlying relational schema. An extension requires a definition of an XML view and triggers responsible for checking the freshness of a view.

A view produced by GDAP which supports all its operations (i.e. read and write) is an updateable XML view of relational data. The view updateability

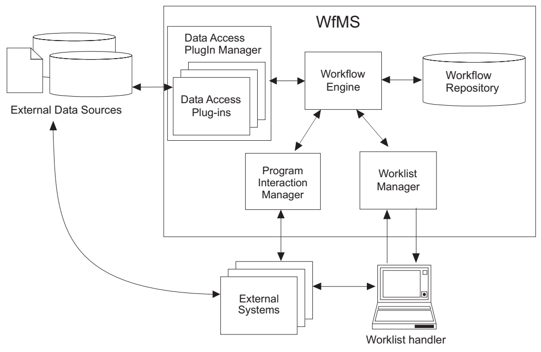

Fig. 1. Workflow management system architecture with data access plug-ins

problem is well known in relational and object- relational databases [5]. The mismatch between flat relational and hierarchical XML models is an additional challenge. This problem is addressed in [12]. However, most proposals of updateable XML views [11] and commercial RDBMS (e.g. [8]) assume that XML view updateability problem is already solved.

The processes and activities managed by the WfMS can run for a long time. At the same time the relations in the original database can be modified by other systems. This raises the problem of view freshness. GDAP offers a possibility to check the view freshness. In case of view update operations GDAP automatically checks whether the view is not stale before propagating update to the original database.

3 Implementation

To validate our approach we implemented a generic data access plug-in [6] which was integrated into our prototype WfMS [10]. A general architecture of our WfMS is presented in Fig. 1. The workflow engine provides operational functions to support the execution of processes. The workflow repository stores both workflow definition and instance data. The program interaction manager calls programs implementing automated activities. The worklist manager is responsible for worklists of the human actors and for the interaction with the worklist handlers. The data access plug-in manager is responsible for registering and managing data access plug-ins. Apart from the generic data access plug-in there may be specialized plug-ins for specific data sources (e.g. legacy systems). Our implementation included GDAP for relational databases and another one for XML files stored in a file system.

The current implementation of our GDAP for relational databases takes advantage of XML-DBMS middleware for transferring data between XML documents and relational databases [3]. XML-DBMS maps the XML document to the database according to an object-relational mapping in which element types are generally viewed as classes and attributes and XML text data as properties of those classes. An XML-based mapping language allows the user to define an XML view of relational data by specifying these mappings. The XML-DBMS supports also insert, update and delete operations. We follow in our implementation an assumption made by the XML-DBMS that the view updateability problem has been already resolved.

Our GDAP controls the freshness of generated XML views using the predefined triggers and so called view-tuple lists (VTLs). Triggers are defined on tables which were used to create a view and log in special log tables information about modifications made to data source tables. A VTL is managed and stored internally by GDAP and contains primary keys of tuples which were selected into the view. The exact method of checking view freshness is not in the scope of this paper and can be found in [6].

4 Conclusions

The concept and the architecture we propose strives for achieving true physical and logical independence of process and data. The abstraction represented in exchangeable plug-ins for data access frees workflow definitions from the accidentiality of representation formats. The implemented general data access plug-in enables flexible publication of relational data as XML documents. Besides the obvious advantages for intra- and interorganizational exchange of data and documents, maintenance and evolution of workflow systems will benefit considerably.

References

1. W. van der Aalst, K. van Hee. *Workflow Management: Models, Methods, and Systems.* MIT Press, 2002
2. T. Andrews et al. Business Process Execution Language for Web Services (BPEL4WS). Tech.Rep., BEA, IBM, Microsoft, SAP, Siebel Systems, 2003.
3. R. Bourret. XML-DBMS Middleware. *http://www.rpbourret.com/xmldbms*.
4. Ch. Bussler. Has workflow lost sight of dataflow?, HPTS Workshop, 1999.
5. C.J. Date. *An Introduction to Database Systems*. Addison Wesley, 2003.
6. Ch. Dreier. Generic Data Access in XML-Based Lighweight Workflow Management System. In German. Master's thesis, University of Klagenfurt, 2005.
7. J. Eder and M. Lehmann. Uniform access to data in workflows. EC-Web 2004, LNCS 3182, pp. 66–75, Springer, 2004.
8. Oracle Corp. *XML Database Developer's Guide - Oracle XML DB. (9.2)*, 2002.
9. M. Sayal, F. Casati, U. Dayal, and M.-Ch. Shan. Integrating workflow management systems with business-to-business interaction standards. In: ICDE'02, IEEE, 2002.
10. M.Siekierski, A.Wojnowska. XForms Workflow Engine. Univ. of Klagenfurt, 2004.
11. I.Tatarinov,Z.G. Ives,A.Y. Halevy,D.S. Weld. Updating XML. SIGMOD, 2001.
12. L. Wang and E.A. Rundensteiner. On the updatability of XML views published over relational data. ER 2004, Springer, 2004.
13. Workflow Management Coalition. Process definition interface - XML process definition language (XPDL 2.0). 2005.