

Uniform and Flexible Data Management in Workflow Management Systems

Johann Eder, Marek Lehmann

University of Vienna, Austria

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. Moreover, we ensure simplicity by binding the human user interface layer of the workflow system with XML-based forms. The concept contributes to transparency of data location, and logical and physical independence of data, business logic and presentation in workflow systems. It facilitates the reuse of predefined activities and forms on different data sets and eases the interaction of a workflow with its environment by abstracting from the actual representation of data.

1 Introduction

There is already a quite long history of endeavours to model business processes and support their execution. A particular role always played the two intertwined, but separate aspects: the data or structural aspect and the dynamic, behavioural or process aspect [23,24]. Workflow management systems are a successful product of these research and development efforts. As we will argue below, workflow management typically focuses on the process aspect of business processes. In this paper we argue that workflow management systems should improve the management and handling of data and present an approach for uniform and flexible data management in Workflow Management Systems.

Workflow processes may involve different kinds of 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 the background[7]. Only quite recently interest in data management aspects of workflow systems increased somewhat with the analysis of workflow data patterns[18] which in particular showed how complex the handling of data in workflow management system actually is.

Workflow Management systems (WfMSs) are not intended to provide general data management systems capabilities, although they have to be able to work with large amounts of data coming from different sources. Business data, describing persistent business information necessary to run an enterprise, maybe controlled either by a WfMS or be managed in external systems (e.g. corporate database). The WfMS needs a direct data access to make control flow decisions based upon data values. An important drawback is that WfMS-external data can only be used indirectly for this purpose, e.g. be queried for control decisions. Therefore, most of the activity programming is related to accessing external databases[3], a great impediment for flexibility, understandability and changeability of workflows[8]. On the other hand, data used within a workflow system may be in different types and formats. Basically each product uses different, proprietary solutions varying from the minimal set of built-in primitive types(number, string, date) to user defined types. This sometimes causes inconsistency with the XML data format used in inter organizational workflows and by web services.

In a typical workflow, performing of a manual task is often associated with filling in some form (e.g. purchase order form). A form has a certain format and is composed of a number of fields with some kind of structure. These fields are mapped to the data used within the workflow. Existing WfMSs support form based manual tasks in two ways: they offer their own proprietary form format or allow web based solutions to be used. The drawback of the former case is, that the client software must understand the form format. The client software must be installed on a machine of each human workflow participant which increases installation and maintenance costs. In the latter case, the workflow management system may offer web based form processing like HTML forms (e.g. PantaRhei[11]). Well accepted and understood web standards enable the use of many design tools and do not require specific client software, except a standard web browser. On the other hand, HTML-based solutions are very limited.

Forms described in HTML are flat, more sophisticated user interface and client behaviour requires lot of programming in scripting languages(e.g. JavaScript) or form processing on the server side. Another important drawback is the necessity to provide the mapping between HTML form fields, represented as a set of attribute-value pairs, and data used internally in a workflow.

We propose to solve all three presented problems. The problematic issue of data location is sorted out by the introduction of data access plug-ins which manage the distributed data sources, so that access to business data is transparent. Limitations of the internal data formats are avoided by basing our system upon XML which provides a flexible data representation. The third problem solved concerns the complexity of the interface layer in most workflow management systems. In order to provide fast and easy communication with the human actors, we use XForms technology to present XML data in a web browser. We validated the functionality of the proposed approach in a prototype WfMS.

Section 2 presents our mechanism for accessing transparently business data stored in many different data sources. An XML based user interface is described in Sec. 3. Both these mechanisms are incorporated into our workflow metamodel in Sec. 4 and the prototype architecture in Sec. 5. We discuss related work in Sec.6 and draw some conclusions in Sec.7.

2 Uniform XML Based Data Access

We propose to provide the workflow management system with a uniform and transparent access method to all business data stored in any data source. The workflow management system should be able to use data coming from external and independent systems to determine a state transition or to pass it between activities as parameters. This is achieved by an abstraction layer called data access plug-ins [13]. Data access plug-ins are reusable and interchangeable wrappers around external data sources which present to the workflow management system the content of underlying data sources and manage the access to it. The functionality of external data sources is abstracted in these plug-ins.

On the other hand, we propose to use XML as the main business data format at every stage of workflow processing. The workflow management system should be able to test conditions on XML data to determine the state transitions, regardless of where these data are stored and maintained. Data passing between activities should also rely on XML-standards, independent of whether these activities are internal to a workflow or external.

Both goals aim at a seamless integration of intra-and inter-organizational workflows and on location transparency of data.

2.1 Data Access Plug-ins

A data access plug-in is a wrapper presenting to the workflow management system the content of external data sources as XML documents. Each data access plug-in provides documents in one or several predefined XML Schema types. Both a data access plug-in and XML Schema types served by this plug-in are registered to the workflow management system. Once registered, a data access plug-in can be reused in many workflow definitions to access external data as XML documents of a given type. A workflow designer specifies in a workflow definition which document should be accessed by which data access plug-in.

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-in. The process definition and activities remain basically unchanged.

The task of a data access plug-in is to translate the operations on XML documents to the underlying data sources. A data access plug-in exposes to the workflow management system using a simple interface which allows XML documents to be read, written or created in a collection of many

documents of the same XML Schema type. Each document in the collection is identified by a unique identifier. The plug-in must be able to identify the document in the collection given only this identifier.

Each data access plug-in allows an XPath expression to be evaluated on a selected XML document. The XML documents used within a workflow can be used by the workflow engine to control the processing flow. This is done in conditional split nodes by evaluating the XPath conditions on documents. If a given document is stored in an external data source and accessed by a data access plug-in, then the XPath condition has to be evaluated by this plug-in. XPath is also used to access data values in XML documents.

Data access plug-ins can be used in workflow definitions described in our workflow definition language WDL-X. A sample WDL-X fragment of an order processing workflow is presented in Fig. 1. There is a declaration of two documents typed with XML Schema. An order document is of type `orderType` (line 3) and is accessed by a data access plug-in `orderDbPlugin`, i.e. it is stored outside of the workflow repository. A document describing an invoice is stored in the repository (line 4).

```

1: <process name="processOrder" owner="marek" version="1.0">
2:   <documents>
3:     <document name="order" type="orderType" accessPlugin="orderDbPlugin"/>
4:     <document name="invoice" type="invoiceType"/>
5:   </documents>

```

Fig. 1. Sample WDL-X script fragment with a declaration of a document accessed by a plug-in

A data access plug-in can be wrapped around any data source, if it provides the described basic functionality. In particular a data access plug-in can be also wrapped around a web service. In this case the functionality of a data access plug-in is limited only to read mechanisms (i.e. open and retrieve document, test XPath expression) and update operations are not allowed. This limitation is imposed by two facts: the data access plug-ins are intended only to serve data, but the web services offer interfaces to provide services and have active properties. A call to a web service can start an external workflow. Such a behaviour initiated by a data access plug-in would make the workflow definition obscure and very difficult to analyze and maintain. Therefore, we argue that an implementation of a data access plug-in based on a web service should be done very carefully to avoid unwelcome side effects during the workflow execution. On the other hand, the web services are very good sources of data. A recent empirical study

shows that majority (up to 84%) of existing public web services are simply data sources, and most of them offer just one operation[14]. Such web services can provide data which are completely managed outside the context of an active workflow instance, e.g. credit card validity, tax rates, metal and oil prices, currency exchange rates.

2.2 Generic Data Access Plug-in

A proposal we described in [13] required data access plug-ins to be defined each time from scratch. On the other hand, most business data remain stored in relational databases. Therefore, a generic and expandable solution for relational data sources was needed. A generic data access plug-in (GDAP) offers basic operations and can be extended by users to their specific data sources. GDAP is responsible for mapping of the hierarchical XML documents used by workflows and activities into flat relational data model used by external databases. Thus, documents produced by GDAP can be seen as XML views of relational data.

The workflows and activities managed by the WfMS can run for a long time. In a loosely coupled WfMS scenario it is neither reasonable nor possible to lock data in the original database for processing time in a workflow. At the same time these data can be modified by other systems or workflows[7]. In order to provide optimistic concurrency control, some form of view invalidation is required[20]. Therefore, GDAP provides a view freshness control and view invalidation method. In case of view update operations GDAP automatically checks whether the view is not stale before propagating update to the original database.

3 XML Based User Interface

We propose an XForms based user interface to the manual tasks. XForms is a new standard[10] for describing forms. It is both XML based and web enabled. These features are very important. First, we use XML as the data format internally in the workflow management system and to communicate with the external systems. Because XForms are designed to work with XML, there are no inconsistencies and no need for special data mappings or transformations. On the other hand, by giving access to manual tasks from a simple web browser, we provide access to workflow applications for a large number of users at very low installation and maintenance cost. Moreover, this enables external users(e.g. customers) to initiate and take

part in a workflow, e.g. by submitting an order request from their own web browser. Thus, enterprise-wide and inter-organizational workflows can be easily created with standardized tools [3].

Traditionally such functionality was provided by HTML forms, but for large and complex systems it is obvious, that HTML forms have many disadvantages. Traditional forms remember the data entered by the user in a flat form, as a set of attribute-value pairs. Such data, when received by the server, have to be additionally processed and validated. Another problem was caused by the necessity to provide a mapping between flat HTML forms and data format used internally in the workflow, e.g. hierarchical XML is usually mapped into HTML with XSLT[2]. On the contrary, the XForms standard allows data to be validated on the client side. Moreover, XForms provide a more advanced graphical interface together with data derivation and calculation mechanisms. XForms make clear distinction between the data content and the graphical presentation. What is most important, XForms are designed to work with XML. All these features make XForms an ideal technology for workflow management systems.

An XForm has a set of input and output parameters which are defined in its code. Each XForm parameter has its unique string identifier and is an XML document which may be typed by an XML Schema type. A form can accept a set of XML Schema typed XML documents as the input presented to a user and used for internal calculations. It can also produce XML documents as output.

A manual task has also a set of input and output parameters which are XML documents described by XML Schema types. The user interface to such manual tasks is provided by XForms. When a user chooses a manual task from a worklist, all the required documents are retrieved(possibly with data access plug-ins) and passed to a XForm as input parameters. The parameterized XForm is presented to the user. After the user finishes the interaction with the XForm, the results are sent as one or more XML documents to the workflow management system which saves them in appropriate locations (possibly using data access plug-ins).

A manual task can have associated a default form in its definition as presented in Fig. 2. The important issue is to provide a mapping between the parameters of a manual task and an XForm. Each formal parameter of the task is mapped to a formal parameter of the form. The number of the parameters must be the same, mapped parameters must have the same XML Schema type and same input mode (*IN*, *INOUT*, *OUT*). The actual parameters of a task instance are used to parameterize a form template at runtime, and a complete and parameterized XForm is send to the client.

```
1: <manualTaskDefinition name="produceInvoice" owner="marek"
    defaultForm="invoiceForm">
2:   <formalParameters>
3:     <formalParameter name="order" inputMode="IN"
        dataType="orderType"
        mapToFormParam="order"/>
4:     <formalParameter name="invoice"
        inputMode="OUT"
        dataType="invoiceType"
        mapToFormParam="invoice"/>
5:   </formalParameters>
6: </manualTaskDefinition>
```

Fig. 2. WDL-X definition of a manual task with an associated default form

Forms are reusable. The same XForm can be used by different manual tasks and can take different documents as input. A workflow designer specifies a default form for each manual task registered in the workflow management system. When the task is used as a single step in a workflow definition, the system uses its default form to provide the user interface. The workflow designer may override this default form in a particular workflow definition and specify that, a manual task used in a particular step should use a different form. This makes the task independent from the user interface representation. For example the same manual task may have different interface to different users, e.g. different to a secretary and different to a manager.

4 Workflow Metamodel

We propose a new workflow metamodel which captures both data access plug-ins and forms together with reusable workflows and activities and associated data (Fig. 3). The metamodel reflects the nested structure of complex activities and supports the graph representation of workflows. The metamodel is tailored to the purpose of this paper and, therefore, does not contain all components required in a workflow metamodel, e.g. we do not consider the organizational structure.

A *workflow* uses *activities* and can have a set of declared *documents*. Documents can be passed to activities as parameters. An activity is either a task, a complex activity or a (sub-) workflow. An activity can be used to compose complex activities and workflows. An activity occurrence in such a composition is represented by a *step*. One activity can be represented by

several steps in one or several other complex activities or workflows. In other words, steps are placeholders for reusable activities. Between the subsequent steps there is additionally a *transition* from a predecessor to a successor. The control structure of a complex activity is described by its *type* (*seq* for sequence, *par* for parallel and *cond* for conditional). We limit our metamodel only to the description of full blocked workflows [26].

In the presented metamodel, workflow graphs can be viewed on different levels of detail. A workflow or a complex activity can be viewed as a composed whole, with all its relations to the subactivities. It can also be viewed decomposed into a full flattened graph. This is achieved by *activity steps* and *control steps*. An attribute *type* of a step can have either a value *activity* (for activity steps) or one of the following for the control steps: *par-split*, *par-join*, *cond-split* or *cond-join*. An activity step offers a compact view of a complex activity. In a graph representation it would be a complex activity represented as a single box. An activity step has a method *flatten* which eliminates a level of composition. A control step represents a control element such as a split or a join. As we permit only full blocked workflows, each join has a corresponding split and vice versa. This is represented by a recursive relation *is_counterpart*. Corresponding control steps are the boundaries of a complex activity. The control steps offer a more detailed view on a workflow graph. They have a method *unflatten* which is inverse to the method *flatten*. Both methods are described in [12].

Each *document* used in a workflow definition is typed with an XML Schema type (*DocType*) which has a unique name. The XML documents may be used as variables declared in a workflow definition, or used as formal parameters to activities. An XML document declared in a workflow definition can be accessed by a *data access plug-in*. Each data access plug-in can serve documents of predefined XML Schema types. The XML Schematype of a document accessed by the data access plug-in must be among types supported by this plug-in.

Each workflow can have many workflow instances. Within a *workflow instance* are instantiated documents and steps of a corresponding workflow definition. Step instances represent actual activity instances. Which activity is instantiated by a given step instance is described by the relation between a step instance and a corresponding step and the relation between a step and an activity. A *type* of a step instance is the same as a type of a corresponding step. The step instances form a workflow instance graph. Each step can have predecessors and successors and the transitions between them. The hierarchical structure of activity instances is reflected by the recursive relation *parent*.

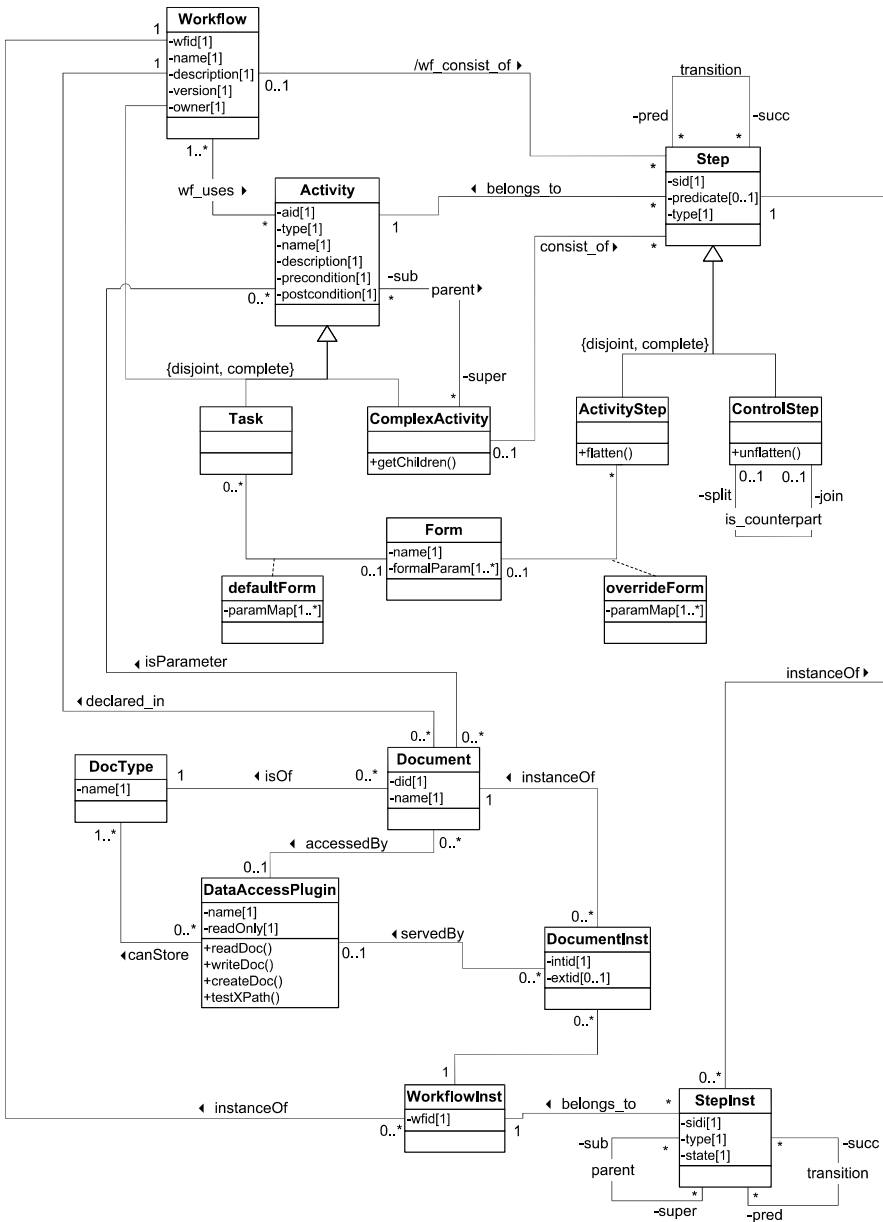


Fig. 3. Workflow metamodel

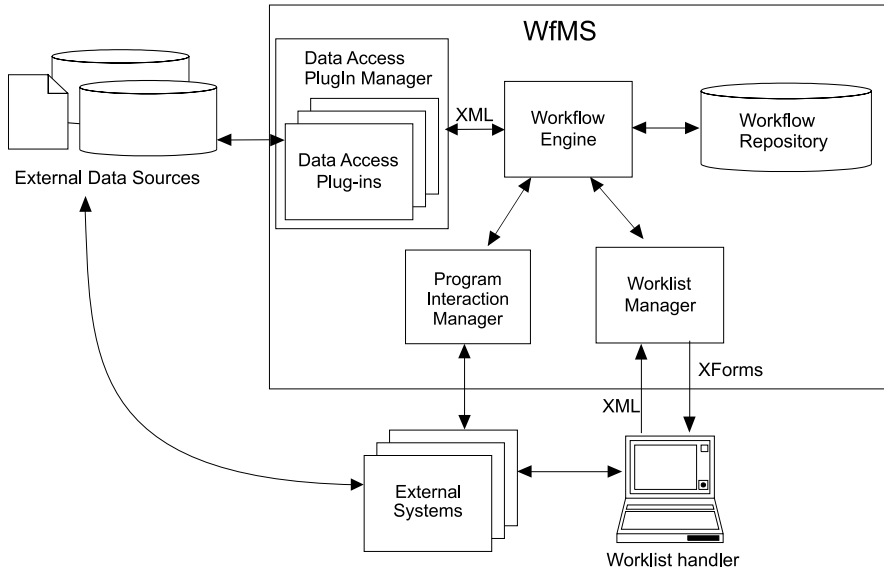


Fig. 4. Proposed WfMS architecture with data access plug-ins

Instances of documents, declared in the workflow definition as accessed with a data access plug-in, are served at runtime by this plug-in. Each document instance has a unique internal identifier, used by the workflow management system. The instances of documents accessed by a data access plug-in have additionally an external identifier, used by the plug-in to identify the document instance. This is a reference to the actual document content in an external data source.

A manual task registered in the workflow management system, may have a default XForm. Each form has a unique name and a set of formal parameters. The parameters of the manual task are mapped into parameters of the form. A workflow designer may override this default form in a workflow definition, and decide to use a different form for a particular step corresponding to this manual task. In this case the parameter mapping must be also provided.

5 Proposed Architecture and Prototype

We propose a new architecture of a workflow management system which supports the usage of XML documents at every stage of workflow processing. This architecture allows the workflow management system to transparently access many sources of business data via data access plug-ins and

to provide user interface to manual task with XForms. The architecture is presented in Fig. 4.

The *workflow engine* provides operational functions to support the execution of workflow instances, based on the workflow definitions. The *workflow repository* stores both workflow definitions and workflow instances (control data). It can also contain business data local to the workflow management system, i.e. local XML documents. 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 client software (*worklist handlers*). Human actors execute the manual tasks with a user interface provided by XForms. Therefore, the worklist handler must be capable of handling XForms. The worklist manager parameterizes XForm templates with XML documents passed as actual parameters to manual tasks and sends these XForms to the worklist handler. The worklist handler may send back the output XML documents.

The access to *external data sources* is provided with *data access plug-ins*. The *data access plug-in* manager is responsible for registering and managing data access plug-ins. The WfMS is extensible, because new data access plug-ins can be registered to the manager and used in the workflow definitions. This architecture is very flexible, because existing data access plug-ins may be replaced by new ones without any integration into existing workflow definitions.

The presented architecture was prototypically implemented [22]. A lightweight workflow engine was implemented as a Java servlet, which produced parameterized XForms instead of standard HTML to communicate with clients. We used Apache Tomcat as a servlet container and the DENG browser¹ to present XForms. The prototype represented internally all business data as XML documents accessed by data access plug-ins. Our implementation included GDAP for relational databases and another one for XML files stored in a file system.

The current implementation of the GDAP for relational databases [9] takes advantage of the XML-DBMS middleware for transferring data between XML documents and relational databases [6]. 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.

¹ See <http://sourceforge.net/projects/dengmx>

6 Related Work

In most existing workflow management systems, data used to control the flow of the workflow instances (i.e. workflow relevant data) are controlled by the workflow management system itself and stored in the workflow repository. If these data originate in external data sources, then external data are usually copied into the workflow repository. There is no universal standard for accessing external data in workflow management systems. Basically each product uses different solutions [19]. A chain of so called materialization and dematerialization programs was proposed in [16]. Such chains can be attached to activities. On the contrary, we proposed to associate data access plug-ins to documents used in a workflow and not to activities. This has two main advantages. First, it allows a business logic of activities to be separated from a data access logic of data access plug-ins. Second, both the activity and the data access plug-in are independent and can be reused in many workflow definitions and easily maintained and replaced.

The importance of XML technology is increasing tremendously in the workflow management. Workflow management systems [26], B2B standards [21], and Web services [4] use XML as a data format. Methods for integrating workflow management systems with standards for web services are becoming more important [17, 21]. Web services are sometimes treated as data sources and composed using data integration techniques [25]. Another approach for processing XML documents in workflow management systems is presented in [5]. The authors proposed to partition a single XML document into several meaningful segments, i.e. units of work that can be performed by an activity in a workflow process.

Forms used for manual tasks can have proprietary format or use HTML like in PantaRhei [11]. PantaRhei used even a form-flow metaphor to provide access to workflow specific data. The authors of [2] proposed to use process aware XSLT style sheets to provide an active user interface to XML data used in workflows. There are also proposals to use XSLT to produce GUI for web services [15]. But mixing XSLT with the business logic can make a workflow definition very obscure. The XForms which provide only presentation and are reusable and data and business logic independent are a great step forward.

7 Conclusions

It was our ambition to show that integrated consideration of the data aspects and the dynamic aspects of workflow systems is possible and that this integrated view can lead to rather lean and flexible systems which are comparatively easy to comprehend and use due to uniform general architectural principles. We did so by designing and implementing a small workflow management system which shows how data management can be both flexible and uniform.

The main contributions of the presented approach for uniform access to data in workflows are:

- Separation of the business logic (activities), data access mechanism (data access plug-ins) and user interface (XForms).
- All data in workflows (application data, workflow relevant data, data in external sources, etc.) are described, represented and processed uniformly.
- We offer a simple and transparent mechanism for accessing data stored in many different data sources (workflow repository, external systems).
- Seamless integration with external systems can be achieved by exchange of process and application data in XML format.
- XML datatypes and data access plug-ins can be reused in many workflow definitions.
- Reusability of activities is made easier and is no longer prohibited by differences in data representation.

Thanks to XForms and the way they are parameterized in our system, the user interface to manual tasks is more flexible and modular. Moreover, the interface is not fixed, as in many typical applications, but the XForms templates can be easily redesigned and improved, without the need to change the implementation. And last but not least, the use of browser ensures portability which is nowadays a very important feature.

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 accidentality of representation formats. Besides the obvious advantages for intra- and interorganisational exchange of data and documents, maintenance and evolution of workflow systems will benefit considerably.

Acknowledgments We would like to thank our students: Christian Dreier, Maciej Siekierski and Aleksandra Wojnowska, who implemented the prototype WfMS and GDAP.

References

- [1] van der Aalst, W., van Hee, K.: Workflow Management: Models, Methods, and Systems. MIT press, Cambridge, MA, 2002.
- [2] Aberer, K., Datta, A., Despotovic, Z.: Separating business process from user interaction utilizing process-aware xslt style-sheets. In WECWIS'02: Proceedings of the Fourth IEEE International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS'02), page 69. IEEE Computer Society, 2002.
- [3] Ader, M.: Workflow and business process management comparative study. Volume 2. Technical report, Workflow & Groupware Strategies, June 2003.
- [4] Andrews, S., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S.: Business process execution language for web services (bpel4ws). Technical Report 1.1, BEA, IBM, Microsoft, SAP, Siebel Systems, 5 May 2003.
- [5] Bae, H., Kim, H.: A document-process association model for workflow management. *Comput. Ind.*, 47(2):139–154, 2002.
- [6] Bourret, R.: Xml-dbms middleware. Viewed: May 2005, <http://www.rpbourret.com/xmldbms/index.htm>.
- [7] Bussler, C.: Has workflow lost sight of dataflow?, 1999. High Performance Transaction System Workshop 1999.
- [8] Carlsen, S., Krogstie, J., Sølberg, A., Lindland, O.I.: Evaluating flexible workflow systems. In Hawaii International Conference on System Sciences (HICSS-30), 1997.
- [9] Dreier, C.: Generischer datenzugriff in xml-gestützten lightweight workflow management system. Master's thesis, University of Klagenfurt, 2005.
- [10] Dubinko, M., Klotz Jr. L.L., Merrick, R., Raman, T.V.: Xforms 1.0. W3c recommendation, World Wide Web Consortium (W3C), 14 October 2003.
- [11] Eder, J., Groiss, H., Liebhart, W.: The workflow management system pantarhei. In A. Dogac, L. Kalinichenko, T. Özsu, and A. Sheth, editors, *Workflow Management Systems and Interoperability*. Springer-Verlag, 1998.
- [12] Eder, J., Gruber, W.: A meta model for structured workflowssupporting workflow transformations. In Proceedings of the 6th East European Conference on Advances in Databases and Information Systems (ADBIS 2002), volume 2435 of Lecture Notes in Computer Science, pages 326–339. Springer-Verlag, 2002.
- [13] Eder, J., Lehmann, M.: Uniform access to data in workflows. In Kurt Bauknecht, Martin Bichler, and Birgit Pröll, editors, *Proceedings of the 5th International Conference on E-Commerce and Web Technologies, EC-Web 2004*,

- volume 3182 of LNCS, pages 66–75, Zaragoza, Spain, August/September 2004. Springer-Verlag.
- [14] Fan, J., Kambhampati, S.: A snapshot of public web services. *SIGMOD Record*, 34(1):24–32, March 2005.
 - [15] Kassoﬀ, M., Kato, D., Mohsin, W.: Creating guis for web services *IEEE Internet Computing*, 7(5):66–73, 2003.
 - [16] Leymann, F., Roller, D.: *Production Workflow. Concepts and Techniques*. Prentice-Hall PTR, Upper Saddle River, New Jersey, USA, 1999.
 - [17] Lienhard, H.: Web services and workflow - a unified approach. In *Workflow-Handbook 2003*, pages 49–60. Workflow Management Coalition, 2003.
 - [18] Russell, N., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Workflow data patterns. *Proc. of 24th Int. Conf. on Conceptual Modeling (ER05)*, 3716:353–368.
 - [19] Russell, N., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Workflow data patterns. Technical Report FIT-TR-2004-01, Queensland University of Technology, Brisbane, Australia, April 2004.
 - [20] Rys, M.: Bringing the internet to your database: Using sqlserver 2000 and xml to build loosely-coupled systems. In *Proceedings of the 17th International Conference on Data Engineering ICDE*, April 2-6, 2001, Heidelberg, Germany, pages 465–472. IEEE Computer Society, 2001.
 - [21] Sayal, M., Casati, F., Dayal, U., Shan, M-S.: Integrating workflow management systems with business-to-business interaction standards. In *Proceedings of the 18th International Conference on Data Engineering (ICDE’02)*, page 287. IEEE Computer Society, 2002.
 - [22] Siekierski, M., Wojnowska, A.: Xforms workflow engine. Technical report, University of Klagenfurt, 2004.
 - [23] Sølvberg, A., Kung, C.H.: Activity modelling and behaviour modelling. In *Information Systems Design Methodologies: Improving the Practice*, 1986.
 - [24] Sølvberg, A., Kung, C.H.: On structural and behaviour modelling of reality. In *Database Semantics*, 1986.
 - [25] Thakkar, S., Knoblock, C.A., Ambite, J-L.: A view integration approach to dynamic composition of web services. In *Proceedings of 2003 ICAPS Workshop on Planning Web Services*, 2003.
 - [26] WfMC. Workflow process definition interface - xml process definition language (xpdl). Technical Report WFMC-TC-1025, Workflow Management Coalition, 2002.

Conceptual Modelling in Information Systems
Engineering

Krogstie, J.; Opdahl, A.L.; Brinkkemper, S. (Eds.)

2007, XIV, 346 p., Hardcover

ISBN: 978-3-540-72676-0