Repository for Business Processes and Arbitrary Associated Metadata

Jussi Vanhatalo[1 2], Jana Koehler[1], and Frank Leymann[2]

[1] IBM Research GmbH, Zurich Research Laboratory,
Säumerstrasse 4, 8803 Rüschlikon, Switzerland
{juv, koe}@zurich.ibm.com

[2] Institute of Architecture of Application Systems, Universität Stuttgart,
Universitätsstraße 38, 70569 Stuttgart, Germany
frank.leymann@informatik.uni-stuttgart.de

Abstract. We have published a repository for storing business processes and associated metadata. The BPEL Repository is an Eclipse plug-in originally built for BPEL business processes and other related XML data. It provides a framework for storing, finding and using these documents. Other research prototypes can reuse these features and build on top of it. The repository can easily be extended with new types of XML documents. It provides a Java API for manipulating the XML files as Java objects hiding the serialization and de-serialization from a user. This has the advantage that the user can manipulate the data as more convenient Java objects, although the data is stored as XML files compliant with the standard XML schemas. The data can be queried as Java objects using an object-oriented query language, namely the Object Constraint Language (OCL). Moreover, the flexible design allows the OCL query engine to be replaced with another engine based on other query language.

1 Introduction

Interoperability based on several XML standards is one of the corner stones of Web services. The Business Process Execution Language for Web Services (BPEL) [2] is the defacto industry standard for representing business processes. It is tightly related to other XML standards, such as the Web Service Definition Language (WSDL) and XML Schema. In addition, arbitrary metadata represented in XML format may be associated to business processes depending on the context and applications that use the data.

XML data is commonly used by different applications. However, currently managing the documents and searching information from their contents is laborious and inefficient. It is beneficial to store the data in a repository that takes care of data access and executes queries. Although it is important for interoperability to exchange data in XML format across organizations and systems, it is often more convenient for a developer to manipulate the data as Java objects, instead of XML. Our goal was to build a business process repository that stores data as documents compliant to the XML standards, but allows applications to be implemented directly on the Java representation of the data model.

We have implemented the BPEL Repository, which is an Eclipse plug-in built to store business processes together with other XML data. It provides a framework for storing, finding and using these documents. Other research prototypes can reuse these features and build on top of it. The repository can easily be extended with additional XML schemas because of its flexible architecture. By default it supports the common Web service standards, such as BPEL, WSDL and XML schema, and it can easily be extended to support other XML schemas for business processes and other data.

The object-oriented approach frees developers from the burden of the underlying XML data model and allows them to concentrate on the object model of their application, which they usually know well. The Eclipse Modeling Framework (EMF) [10] is used to hide data serialization and de-serialization from the user. The framework takes care of representing the XML data as EMF objects that are Java objects. As a novel feature, it is possible to query the XML files as EMF objects using an object-oriented query language, namely the Object Constraint Language (OCL) [7] that is part of the UML specification. Native XML databases support typically XQuery as their query language. A major advantage of OCL over an XQuery is its ability to navigate through the data model and follow all the associations of an object model. In contrast, XQuery forces the user to formulate the queries based on the tree structure of the underlying XML schema.

In contrast to our file system based solution, there are other business process repositories [12] [14] that are built on top of a database system. However, their database schemas are created manually. Flexibility is an advantage of the BPEL Repository, because EMF is used to automatically generate support for new and modified XML schemas. In research projects, data structures are often modified and new ones are introduced. The automatization makes adapting these changes easier. Nevertheless, repositories based on a database have typically better performance and scalability than our solution.

The BPEL Repository was recently published in IBM alphaWorks [16] with special licensing terms for academic use. The software has been integrated with a change management system called CHAMPS [3], [13].

2 Solution

The architecture of the BPEL Repository is presented in Figure 1. All components are plug-ins on the Eclipse platform. The core component of the solution is the Repository API, which provides an application programming interface for external software to build on.

The Repository User Interface (UI) is an example implementation that uses the Repository API. However, it is also a useful graphical user interface to manage the contents of the repository. The user interface is integrated in the Eclipse workbench and built on the Standard Widget Toolkit (SWT) and JFace libraries.

The Data Handler is a sub-component that takes care of the data access on a file system. It abstracts the choice of the storage medium from the other

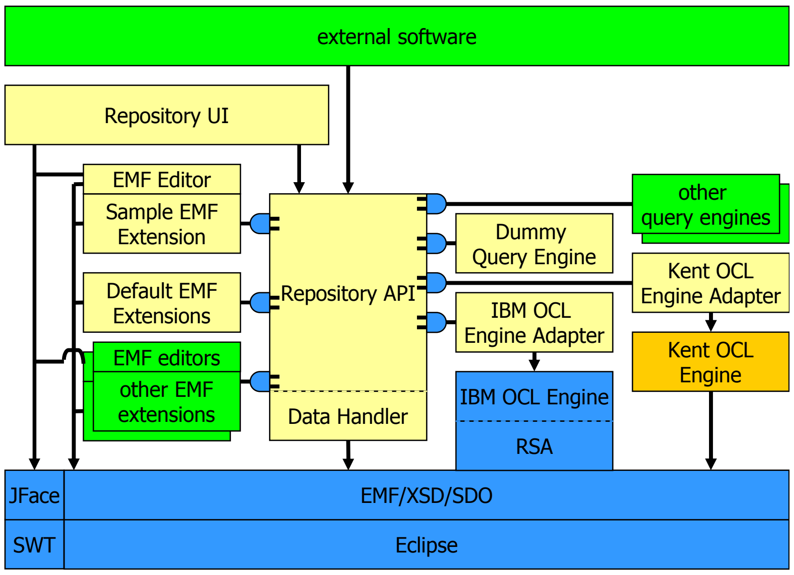

Fig. 1. Repository architecture showing components and technologies used.

components. The data access component could be replaced with another one storing data in a database by using a technology such as the Service Data Objects [10]. The Data Handler uses the Eclipse Modeling Framework to serialize EMF objects into XML files and de-serialize the files back to EMF objects. Thus, all repository components manipulate data as EMF objects rather than of XML.

2.1 Flexibility of Manipulating Data as EMF Objects

In the repository, data is represented as EMF objects. Therefore, all data must have an EMF model. However, the EMF model can be automatically generated from an XML schema, a UML class diagram or Java classes [4], [10]. As in the business process management context data is often stored as XML conforming standardized XML schemas, it is trivial to obtain EMF models for XML files. The repository can be extended to support a new data type by plugging in the EMF model of this new data type.

The components providing EMF models for the repository are shown on the left-hand side of the Repository API in Figure 1. The Default EMF Extensions plug-in contains EMF models for BPEL, WSDL and XML schema standards. Thus, the repository supports the respective file types by default. This component can be replaced by another component supporting a different version of these standards or completely different file types. Because the Eclipse plug-in mechanism is used, this does not require any modifications in the other parts of the repository.

Similarly, other EMF extensions can be plugged into the repository. In the evolving research community, extensibility is an asset. For instance, in the context of combining business process management with semantic Web, the repository can easily be extended to support a new document type containing metadata related to a business process.

The Sample EMF Extension contains an EMF model that is used in the user guide [16] to illustrate, step by step, how to create an EMF model and plug it into the repository. It is also explained how EMF can be used to automatically generate a graphical editor for the instances of an EMF model.

First, the data structure can be modeled as a UML class diagram, which is usually much faster than describing the same structure as an XML schema. Next, the UML class diagram is transformed to an EMF model. An editor can be automatically generated for the EMF model. An XML schema can be generated from the EMF model, if desired. In any case, instances of an EMF model can be serialized to interoperable XML files. Instances of the model can be generated for testing purposes with the editor, which takes care of proper syntax. Finally, the EMF model is plugged into the repository, which persists the data and provides capabilities for querying data. A chief advantage is that queries can be formulated using the same object-oriented model as was used to create the data structure in the first place. Thus, the XML representation is used only for interoperability with other systems, and the developers need not bother with the concrete XML syntax.

2.2 Query Engines

We used existing query engines with the repository. The repository handles the iteration over the queried objects, but each sub-query is executed in the query engine plugged into to the repository. It is possible to change the query engine to another pre-registered one between queries. The available query engines are shown on the right-hand side of the Repository API in Figure 1. The repository has been tested with two OCL query engines, that query Java objects with an object-oriented query language, namely OCL.

If the repository is installed on top of the IBM Rational Software Architect (RSA) product, the OCL engine of the latter can be used. However, as we did not want to limit the repository to a single commercial query engine or a specific query language, the query engine interface has been built generic. Therefore, the IBM OCL engine is plugged into the repository using an adapter. The IBM OCL Engine Adapter is delivered together with the repository.

Another OCL engine was built at the University of Kent [1]. It is an open source tool that can be plugged into the repository using the Kent OCL Engine Adapter. The Eclipse Modeling Framework Technology project [11] is building another open source OCL engine that could also be adapted to the repository. We have not yet implemented the corresponding adapter because this OCL engine is still under development.

It is straightforward to plug a new query engine into the repository or adapt an existing query engine for it. An example of how to adapt an OCL engine is included in the user guide [16]. The Dummy Query Engine is an example implementation to show how a new query engine can directly be integrated into the repository. Thus, also query engines based on another query languages can be used.

The repository is not aware of the query language that is used. The repository merely passes the query and other parameters from the user interface or external software to the query engine selected together with the EMF object that is to be queried. Thus, any query engine that can execute queries on EMF objects can be plugged into the repository.

One limitation of the query mechanism is that the performance is only linear compared with the number of documents that are queried. Indexing data or other ways to improve the query performance are not used. However, this performance has been sufficient for research prototypes. For example, querying 100 BPEL files took 3 seconds on a laptop in our performance tests [15]. One way to improve OCL queries would be to map them to a query language, such as XQuery, that is natively supported by a database system. In that case, the repository would also be built directly on top of the database system. Some work on mapping OCL to XQuery already exists [5], [6].

2.3 Data Structure

The data is organized in a tree of organizations. The organizations are mapped to directories in a file system. Each organization may contain a business process and associated metadata grouping the related files together. In addition to these data documents, a descriptor document is stored in each organization. It contains the file type and the role of each data document in the organization. This information is used to make the conversion between EMF objects and XML files. In addition, the role describes how the data document is related to the other documents in the organization. For instance, a WSDL file stored with the repository may contain the public interface or the partner links of the BPEL business process.

Queries can be applied to files with a specified role in an organization, a list of organizations, or a list of sub-trees in the organization hierarchy. Related files can be searched based on their roles.

Data can be accessed from the file system as XML files and through the repository as EMF objects. Any directory in a file system can act as the root organization of the repository contents. The data in the repository can be moved to another location or a computer as simply as copying the directories.

2.4 Usage Scenario

The repository has been deployed with a change management system called CHAMPS. As part of the solution, planning algorithms are used to facilitate the automatization of the change and configuration management. The plans are stored as BPEL files into the BPEL Repository. In addition, the plans are analyzed and the results are stored as metadata associated to the plans. The metadata includes information about the plan such as its number of activities, degree of concurrency, execution duration and correctness [13].

Storing the analysis results as metadata enables reuse of the data, and unnecessary recomputing of the results can be avoided. The suitable plans are found from the repository by querying the content of the plans and their associated metadata. For example, the plans that are structurally correct can be found by querying the metadata. Among these plans the ones that reach a specified goal can be found with a subsequent query.

During the development of the system, trying out different alternatives of the metadata schema was uncomplicated, since the data structure was designed as a UML class diagram and the corresponding EMF classes were used as the basis of the implementation and the OCL queries. The developers were able to avoid completely working with the XML representation of the data, because it was automatically generated and used only as the serialization format behind the scenes.

3 Conclusion

The repository has already been proved useful for IBM internal research prototypes [13]. By publishing the repository, we wanted to make it widely available to the research community as we are interested in more user experience with it. We would also be interested in finding out how convenient users find OCL as a query language, because currently OCL is more common as a language to express constraints rather than queries.

As next steps, we plan to contribute our experiences gained while building the repository to the IP-SUPER project [8], [9] funded by European Union. The project merges business process management with semantic Web services. As part of the project, we plan to build a business process library, most likely on a database system rather than a file system in order to improve the query performance for more extensive querying purposes. This would also be beneficial, when the repository is used for searches based on an ontology or as a component of a business process execution engine.

References

1. Dave Akehurst and Octavian Patrascoiu. Object constraint language library. Web site, June 2004. http://www.cs.kent.ac.uk/projects/ocl/.
2. Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic, and Sanjiva Weerawarana. *Business Process Execution Language for Web Services.* OASIS Org., 2003. http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/.
3. Aaron B. Brown, Alexander Keller, and Joseph L. Hellerstein. A model of configuration complexity and its application to a change management system. In *Proceedings of the 9th International IFIP/IEEE Symposium on Integrated Management (IM 2005)*, May, 2005.

4. Frank Budinsky, David Steinberg, Ed Merks, Raymond Ellersick, and Timothy J. Grose. *Eclipse Modeling Framework*. The Eclipse Series. Addison-Wesley Professional, 2003.
5. Ahmed Gaafar and Sherif Sakr. Proposed framework for integrating XML/XQuery and UML/OCL. In *Proceedings of the 7th Conference in the UML series (UML2004)*, pages 241–259, Lisbon, Portugal, 2004.
6. Ahmed Gaafar and Sherif Sakr. Towards a framework for mapping between UML/OCL and XML/XQuery. In *Proceedings of the IADIS e-Society 2004 Conference (ES2004)*, pages 241–259, 2004.
7. Object Management Group. *OCL 2.0 Specification*. OMG, 2005. http://www.omg.org/docs/ptc/05-06-06.pdf.
8. Martin Hepp, Frank Leymann, John Domingue, Alexander Wahler, and Dieter Fensel. Semantic business process management: A vision towards using semantic web services for business process management. In *Proceedings of the IEEE ICEBE 2005*, pages 535–540, Beijing, China, October 2005.
9. IP-SUPER. Semantics utilised for process management within and between enterprises. Web site, April 2006. http://www.ip-super.org/.
10. Eclipse Org. Eclipse modeling framework. Web site. http://www.eclipse.org/emf/.
11. Eclipse Org. Eclipse modeling framework technology. Web site, 2006. http://www.eclipse.org/emft/projects/ocl/.
12. Minrong Song, John Miller, and Ismailcem Arpinar. RepoX: XML repository for workflow designs and specifications. Technical Report #UGA-CS-LSDIS-TR-01-011, University of Georgia, August 2001.
13. Biplav Srivastava, Jussi Vanhatalo, and Jana Koehler. Managing the life cycle of plans. In *Proceedings of the 17th Innovative Applications of Artificial Intelligence Conference*, pages 1569–1575, Pittsburgh, Pennsylvania, USA, 2005.
14. Tammo van Lessen. Konzipierung und Entwicklung eines Repository für Geschäftsprozesse. Master's thesis, Institute of Architecture of Application Systems, University of Stuttgart, March 2006.
15. Jussi Vanhatalo. Building and querying a repository of BPEL process specifications. Master's thesis, Helsinki University of Technology, Institute Eurecom and University of Nice – Sophia Antipolis, September 2004.
16. Jussi Vanhatalo. BPEL Repository. IBM alphaWorks, April 2006. http://www.alphaworks.ibm.com/tech/bpelrepository.