REPOX: AN XML REPOSITORY FOR WORKFLOW DESIGNS AND SPECIFICATIONS*

`

MINRONG SONG

JOHN A. MILLER[1]

ISMAILCEM B. ARPINAR

August 26, 2001

Abstract

More and more workflow systems are taking XML as their basic data format for workflow process definition, data type definition, and control information definition. For XML-based workflow systems, an XML repository is used to manage XML object resources in a safe and efficient way. RepoX, an XML repository, has been developed for the METEOR workflow system. It maps XML documents to a relational-object database and also provides extraction/retrieval, version control, check in/check out, and searching and query functions. In addition, it has support for adaptive workflows, which may need workflow definition information from the repository at runtime, in a dynamically changing environment.

Keywords: XML, Repository, Version Control, Configuration Management, Workflow Systems, Adaptive Workflow, RepoX

---

* *We would like to thank the LSDIS Lab members and its director, Dr. Amit Sheth, for their help with this project*

1 *Contact person:*
John A. Miller
Department of Computer Science, The University of Georgia, 415 Boyd Graduate Studies Research Center, Athens, Georgia 30602, U.S.A.
Tel: (706) 542-3440 Fax: (706) 542-2966 Email: jam@cs.uga.edu

1. Introduction

Many business transactions and processes are tedious and time-consuming. Workflow applications provide automatic and manageable solutions to save time and effort for human staff by automating, reengineering, and optimizing business and scientific processes. To ensure scalability, a typical workflow system may need to exchange and distribute information, or even incorporate other workflow systems to facilitate interoperability. So a standard data format is highly needed for information interchange. The Extensible Markup Language (XML) is becoming a new universal format for exchanging data on the World Wide Web. Many groups are beginning to make workflow definitions in XML format. All workflow-related information including workflow process definitions, data type definitions, and transition information can be defined in XML document format. The Workflow Management Coalition (WfMC) announced an initiative to provide XML-based workflow standards in 1999 [1].

For workflow systems using XML documents for saving workflow process definitions and especially for XML-based workflow systems, a repository is needed to provide control for the XML documents. The repository allows uniform access to shared data and facilitates integration among tools in the workflow system [2]. An XML repository gives the best solution to maintain, exchange, and modify the workflow process definition metadata, which is in the form of XML documents. The XML repository, where users can look up the workflow definition objects, serves as the metadata foundation of the workflow system.

This paper will focus on some key issues for XML repositories such as storage, extraction, retrieval, version control, and configuration control. RepoX, which is an XML repository implementation for workflow systems, is also introduced. RepoX's support for developing workflow applications, especially in workflow designs or specifications, is addressed in detail. The ultimate design goal for this repository is to support adaptive workflows. A secondary goal for this repository is that it should be easy to retarget it to

other applications (*e.g.,* saving specifications of simulation models). The RepoX repository is used in the fungal genome project sponsored by the Genetics Department of the University of Georgia to support development of workflow applications.

The rest of this paper is organized as follows. Section 2 explains the concept of repository and XML repository. Section 3 addresses the role of XML repositories in workflow systems. Sections 4 and 5 deal with storage, retrieval, and control management issues for XML repositories together with solutions and examples from RepoX. Section 6 illustrates the repository's supports for developing workflow applications. Section 7 concludes the paper and discusses future work.

2. Repository and XML Repository

2.1 Basic Concept of Repository

"A repository is a place to define, store, access, and manage all the information about an enterprise including its data, and its software systems" [3]. It is not just a passive data dictionary or database. More than information storage, the repository, which is an integrated holding area [4], should also keep the information up to date by providing processing methods and make it available to a user as needed. A repository, which maintains valuable information about all of the information system assets of an organization and the relationships between them [5], acts as a central manager of all of the information resources in an enterprise. A repository should provide services such as change notification, modification tracking, version management, configuration management, and user authorization [6].

Several standards have been developed for the repository marketplace. The Information Resource Dictionary System (IRDS) is a standard that describes the requirements and architecture of a repository [7]. The Case Data Interchange Format CDIF) was initiated by the major CASE (Computer-Aided Software Engineering) vendors and developed as a formal way of transferring data between CASE tools [8]. The

Portable Common Tool Environment (PCTE) is a standard for a public tool interface for an open repository [9].

2.2 XML—The New Standard for Exchanging Data Electronically

XML—the Extensible Markup Language—is a new standard adopted by the World Wide Web Consortium (W3C) to complement HTML for data exchange on the web [10]. The Internet, which has no built-in semantics rather than those required for presentation of the data, is overloaded with the data explosion in electronic information systems. XML is a solution that can associate semantics with data. So that data and semantics can be transmitted together over the Internet. XML is gaining widespread acceptance in the industry for its simple, open, extensible, and self-explaining nature.

XML was designed specifically for describing the contents of data, rather than the presentation. It is different from HTML in three major respects.

1. Users can define their own tags as they wish.
2. Any structure can be nested to an arbitrary depth.
3. An operational grammar description can be associated with an XML document.

A Document Type Definition (DTD) serves as a kind of grammar for the corresponding XML document and is part of the XML language. A DTD can also be a schema for the data represented in the XML document to some extent. XML schema complements DTDs by introducing types and allowing user-defined data types.

2.3 XML Repository

The information technology constitutes logically coherent concepts and is always distributed over several different data sources, which may be from different organizations and are heterogeneous in nature. The widespread availability of XML-capable clients and their flexibility in structuring information make it possible for XML to become the

universal data format. Without the help of a repository, it will be difficult to control XML objects in a manageable way and make them available when needed.

"An XML repository is a special purpose repository that can manage XML objects in a native format allowing the developer to focus entirely on business logic, instead of database design and programming" [11]. It should provide several basic functions such as importing/exporting XML data from original text files, user check in/check out, version control, as well as searching and querying on XML elements. In the electronic commerce world, XML repositories are the online source for obtaining the appropriate tag, document-type definition, data element, database schema, software code or routines. As a result, companies, especially small enterprises, can speed up processing and expand their ability to conduct electronic commerce. One major problem is the integration of heterogeneous biological databases, and XML provides an exchange mechanism.

2.4 New Standards for XML Repositories

XML/EDI (Electronic Data Interchange) developed by the XML/EDI Group represents a new framework for e-business data exchange. It is one of the most important applications of XML, which combines the semantics of EDI with XML. Documents in the XML format can be exchanged across different organizations according to those standards easily, so business and scientific processes can be synchronized from application to application. The XML/EDI group also proposed a standard on public repository for XML message definition [12].

Other examples where XML dialects are used in repositories include AIAG in the automotive domain, and HL7 and HIBCC in health care.

3. XML Repositories in Workflow Systems

Many enterprise business and scientific processes as well as research procedures require both human staff and software applications to process data in a pre-defined way.

Workflow Management System (WfMSs) is a system to provide complete support for process definition, workflow enactment, and administration and monitoring of workflow processes [13]. Each node in the whole processing chain or network works on a specific task and sends the results and other control information to the next node if it exists. Figure 1 shows the components of a typical workflow system.

Typically, workflow systems need to integrate other software such as client software and application software into one whole system. So the translation between the workflow API and the other software's native API may increase the cost of software development and maintenance. Moreover, if the whole processing procedures become more and more complicated, they may begin to include other processes from different organizations. If different organizations use different workflow systems, it can be a problem to communicate with each other. If an XML-based workflow standard becomes adopted, different workflow systems can exchange their processing descriptions much more easily. Different workflow engines can communicate and cooperate with each other by the means of interoperability [14, 15]. Figure 2 shows the Workflow Reference Model that describes the Five Interfaces. Process Definition Interchange (Interface 1) is the interface between the bulid-time and run-time environments to make the process definition generated by one modeling tool usable by workflow runtime systems. It has recently been rewritten to use Wf-XML [16, 17]. Some specifications on the Wf-XML language definition are addressed in [1].

3.1 METEOR Workflow Management Systems

METEOR (Managing End-To-End OpeRations) workflow management systems are developed by the workflow research group at the Large Scale Distributed Information Systems Lab in the Computer Science Department at the University of Georgia. The METEOR WfMSs [18, 19, 20] include both the design/build-time and runtime enactment components that support large-scale multi-system workflow applications in

heterogeneous and distributed operating environments.

ORBWork and WebWork are such two WfMS implementations. ORBWork is a fully distributed CORBA-based workflow enactment service [21, 22], and WebWork is a fully distributed workflow enactment service relying solely on Web technology [23].

3.2 RepoX—An XML Repository For METEOR Workflow Systems

In the workflow process definition phase, many types of metadata are produced for workflow specification and task instances. “A Workflow Process Repository is needed to store metadata about workflow designs, organizations, informational resources, and computational resources” [24]. RepoX, an XML repository management tool, is developed in the METEOR Workflow System environment for the purposes of managing XML-based metadata efficiently and providing other basic repository functions. The metadata stored in the repository contain the definition of workflow processes and are in the form of XML documents.

RepoX is a Client-Server model repository with a relational or object-relational database at the backend. RepoX helps workflow application developers to produce workflow specifications and also manage all the XML documents related to workflow process definitions. The architecture of the RepoX repository is shown in Figure 3. The client and the server communicate with each other using Java Remote Method Invocation (RMI). The client side is a GUI tool integrated with a graphical workflow design tool developed by the Naval Research Center. A workflow developer can design workflows using the design tool and then save the XML-based workflow process definition metadata to the RepoX repository or first save to the local file system then export it to the repository some time later. RepoX also provides a user-friendly query tool as part of its client side, so that a user can browse and navigate the repository easily and give the query command to select interested objects.

On the server side, spec definitions can be produced according to the DTD specifications. The DTD are used to generate schema for Oracle 8i or MySQL (could use others). Elements needed are extracted from the XML documents according to the spec definitions and put into database tables. Also the whole XML document is stored in a specific “xmldoc” table as Character Large Object by the CLOB utility supported by the object-relational database. The document is correlated with the table where objects extracted from it are stored by setting this table’s foreign key to the key of the “xmldoc” table. For the later retrieval phase, the whole workflow process definitions together with related information such as input or output data definitions, and role definitions can be exported to the client side in one retrieval operation. XML document storage and retrieval will be addressed in detail in section 4.

4. Storage and Retrieval in XML Repository

A repository is the central area for holding all the information of an organization. Therefore, the storage and retrieval of data objects are the most basic services that a repository should provide. For an XML repository or XML database, recent research work has focused on the efficient storage of XML document to take advantage of its structure [25].

4.1 Traditional Approaches to Implement XML Repository Storage

There are obvious mismatches between traditional storage solutions and XML, which has its own specific structure. XML documents associates semantics with data and its DTD in the case of valid documents contains the structure definition. The following table compares several straightforward approaches for storing XML documents.

Many current commercial XML data servers use an object-relational system as a backend database. In these systems, a storage schema is derived from the typing information given by the DTD [25].

4.2 A Native XML Database Approach

A native XML database exploiting the semi-structure of the documents is a database designed especially to store XML documents. It is another approach to be considered to store data, especially in environments having a large collection of XML documents. The main difference between a native XML database and other relational database management systems or object database management systems is that the native approach's internal model is based on XML structure and not something else. As other systems map XML documents to their own structures, additional layers are introduced between logical data and physical storage. So updating and query processing will be less efficient [26]. Tamino from Software AG is a famous native XML Database in the XML database market.

4.3 RepoX's Approach for Storing and Retrieving Workflow Designs

RepoX is an XML repository developed especially for storing and retrieving workflow designs. The storage and retrieval strategy of RepoX does not simply fall into one exact approach as described above. RepoX works with either a relational or an object-relational database (currently Oracle8i or MySQL) at the backend and builds other operations on top of it to provide more functions.

4.3.1 Modeling Process

In RepoX, the metadata model is first created according to the structure of the XML documents from the design tool. All these XML documents are related to workflow design definitions such as network task definition, data type definition, domain environment, and role domain definition. The DTD specifications for those definitions are defined by the design tool developers. Following the diagram in Figure 4, a user can define the database schema beginning with the DTD specifications.

In step one, a workflow application developer can map the DTD specifications to the UML meta model. During this phase, a user needs to analyze the DTD specifications and decide on the tables (classes) to be created, the attributes to appear in the tables, and the relationships between these tables. A simplified version of the DTD specifications for the definition of network task is given in Figure 5.

Step one is the most important phase because all the tables are defined here. It can be in an ad-hoc manner so that some important or obvious tables are created first. Later, more tables and relationships between tables can be added. In Figure 6, a UML diagram of the meta model for RepoX is given.

In step two, everything created in step one, including tables, attributes in the tables, and relationships between tables, are all defined in the Java file named MapXML2Tables.java. Figure 7 shows the code that gives the definition of tables.

Step three is actually automatic. The output of calling MapXML2Tables class can be used directly by a database to create the database schema. For example, the definitions of the table “xmldoc” and the table “task” for MySQL are given in Figure 8.

If the workflow process definition is changed when the corresponding DTD specifications are modified, the specification definitions will also need to be modified. How to reflect these changes quickly in a repository and at the same time keep the previous versions of documents valid is a problem. The XSL Transformations (XSLT) can transform an XML document in one format to an XML document in another format. Because of the changes made in the DTD, the old XML document may need to be constructed. An application probably with a GUI tool should be developed to support online editing and modifications of the specification definitions. New schema can be generated according to the new specification definitions. This will be left to future work to support regenerating schema and still keeping compatibility with old data.

4.3.2 Data Filtering, Extraction, and Retrieval

A data extraction and filtering component is used to extract and validate the data taken from external data sources. Data filtering not only selects the XML document elements that need to be extracted, but also checks data for validity and consistency. According to the results of filtering, extraction will store atomic XML elements and XML attributes (with types such as CDATA or PCDATA) as the values of the attributes in the relational table. Composite XML elements, which contain other XML elements, are stored as relational tables.

During the extraction process, the DTD files of the XML document files are used by Dynamic XML (DXML, from ObjectSpace, Inc.) to generate a Java Class for every XML element. DXML makes it easier for the XML developer to develop XML applications by providing this Class with a way for "random access" of XML element. A developer can access any part of the data at any time as a regular Java object. When the XML document is viewed as a tree structure, the spec definition looks like a path expression starting from the root node down to the desired leaf node. The destination is the target atomic element that will be pulled out as the value of the attribute. An element can also be referred to by substituting its name with the symbol '*'. The symbol '*' means a collection and will return all elements that are the children of the element in the current context, regardless of their tag names. For example, the spec definition is given as

{'NetworkTask', 'SimpleSubTaskList', 'Task', '*'}.

A spec definition is similar to expressions with XPath syntax. Figure 9 is an instance of network task. According to the spec definition above, a collection of elements related to the tag "task" will be extracted and stored in the table "task" created in the schema generation phase. The whole XML document is saved as Character Large Objects (CLOBs) in the table called "xmldoc". In addition, the XML elements extracted and stored in relational tables can be related to the XML document where they come from by setting the foreign key to the table "xmldoc". For more details of data filtering and

extraction from XML documents, see [24].

A network task can be a workflow if it is the root, or it can be a sub-workflow if it is a component of another network task. A specific extraction utility is developed for an XML document representing a network task. If a user chooses a network task and wants to export it to the repository, the XML document representing this network task will be parsed and all the network tasks it contains will also be exported to the repository. It is a recursive procedure that all the related XML documents representing those network tasks will be saved to the repository.

The RepoX repository has a GUI with a workflow design tool integrated on the client side. A single XML document can be selected and retrieved. The whole workflow process definition together with related information, such as input or output data definitions and role definitions, can also be retrieved to the client side as one retrieval operation. When a user wants to get a workflow design from the repository and review it by using the design tool, all the metadata related to the workflow process definition are already there.

5. Version and Configuration Management

Version management is widely used in software development projects. Some well-known version control systems are Revision Control System (RCS; 1980s), Concurrent Versions System (CVS; 1986), and Source Code Control System (SCCS; 1972). Version management is also important in workflow application development.

5.1 Version Management

It is important to decide at which level to implement versioning. The simplest approach is to store the entire XML document as different versions. However, too much storage space will be wasted. As an XML document has a tag-based structure, we can treat each tag as a versionable item. If the XML document is mapped to an Object-

Oriented database, every tag can have its own OID. This approach will be too complicated if the XML document has many levels. Therefore, deciding the level for versioning is application and user dependent. In RepoX, the level is set at the level of simple task and sub-workflow for network task definition. A simple task is atomic, and a network task, which can contain other network tasks, is composite. A version of a network task is only stored once no matter how many times it is a component in other network tasks. Figure 10 shows a network task C that is referenced both in network task A and network task B. However network task C will be stored only once; the additional storage includes the links from network task A to network task C and from network task B to network task C. Storage space is not wasted for duplicate storage in this approach. If the structure of a network task is modified for a new version, it is easy to trace the changes.

For efficient version storage, one common solution is to store each new version as the delta changes of the previous version. Another popular solution is to use reverse deltas to reconstruct versions. Because the current version is always used most, it is stored entirely in the repository. The previous versions can be recomputed using the reverse deltas. This solution has an advantage over forward delta storage. As more and more revisions are added, the faster retrieval time for the latest version becomes more significant.

In the RepoX repository, several developers can work cooperatively on the same workflow project, possibly remotely, at the same time. A user can obtain a copy of the metadata for a specific workflow design from the repository and store it locally (Check out). The user can then view the workflow definition graphically and make modifications. When all the changes are made, the user can export it back to the repository (Check in). Figure 11 shows the process of check-in and check-out.

It may be necessary for more than one developer to work on the same project and modify the same version of an XML document at the same time. One strict but simple choice is to lock the whole XML document. Other users can not make changes on this version item until it is checked back in. Consistency is maintained, since only one user at a time can work on a specific document. Another open but more complicated solution is to allow concurrent revision in a regulated fashion via a temporary branch in the version history of an item [27]. Merging may occur in a later phase. In the RepoX repository, the whole XML document is locked when a user checks it out. It will be unlocked when explicitly checked in again. This is implemented by maintaining a consistent and persistent lock table in the RepoX.

5.2 Configuration Management

Configuration Management is used to coordinate the work of different developers who are working on the same project at the same time. Lack of configuration management will cause many problems, such as missing documents, inability to track the changes, and inability to maintain different versions. “A configuration is a named collection of atomic entities and other configurations” [28]. Unlike version management that treats atomic entities, configure management controls composites or configurations by applying selection mechanisms.

To consider all atomic entities at the same time, the number of possible combinations of these versions will be too large. Attempts to deal with all the combinations manually are impossible and configuration item selection becomes very important. Selecting too many configuration items will cause hampered visibility and poor management rather than giving improved control. On the other hand, selecting too few configuration items will cause not only loss of visibility down to the required level for maintenance or modification, but also difficulty in managing the changes effectively.

Intentional versioning uses formulated selection rules (such as the latest version) to choose the particular variant and version of an atomic entity, thus automating the selection process and reducing the number of the combinations. Many existing configuration management systems, such as Rational ClearCase and CVS (Concurrent Versions System), use intentional versioning.

Many state-based systems (a version is characterized by its state, *e.g.*, revision running under Solaris) use extensional versioning model [28]. A versioned item is defined as a version set by recursively enumerating its members. Its member can be another versioned item. Each version is identified by a unique version number and is immutable if it has been checked in. SCCS (Source Code Control System) and RCS (Revision Control System) use extensional versioning.

In the RepoX repository, a task is the atomic object and a network task is the composite object that may contain tasks and other sub network tasks. As the Gene workflow in Figure 12 shows, there are four simple tasks, “Retrieve”, “StoreCosmid”, “SetReads”, and “Annotation” as well as one sub network task called “HTBLAST”. As the definition and creation of a composite object may cause configuration problems, extensional versioning is used for the configuration of workflow design structure and versions.

Network tasks can be shared and invoked as sub network tasks in other network tasks. There is a relation between a network task and its sub network task, because the sub network task can be invoked in different conditions by different network tasks. Suppose network task N_1 has a task T_1 and a sub network task N_3 with a relation R_{13}, and network task N_2 has a task T_2 and a sub network task N_3 with a relation R_{23}. So N_3 is shared both by N_1 and N_2.

$$N_1 = \{T_1, N_3; R_{13}\} \quad (1)$$

$$N_2 = \{T_2, N_3; R_{23}\} \quad (2)$$

If we change the property of N_3 in N_2 and make it a new version to N_3’, N_2 will be

$$N_2 = \{T_2, N_3'; R_{23}\} \quad (3)$$

If we do not take the versioning into consideration, the specifications for both network task N_1 and N_2 will both include N_3'.

$$N_1 = \{T_1, N_3'; R_{13}\} \quad (4)$$

N_1 will have an unexpected change as N_3 is changed to a new version N_3', so N_1 may fail. To avoid this, we introduce the rule in RepoX that a network task must be identified by both its name and version. Therefore, N_1 is not changed in this case.

$$N_1 = \{T_1, N_3; R_{13}\} \quad (1)$$

To represent the structure and version of the components of a network task, a directed acyclic graph (DAG) is stored for every version. An edge in the graph means that the end node, which represents a sub network task, is a component of the start node. For network task N_1 in the previous example, the graph has only one edge with two nodes as $(N_1; v1.0) \rightarrow (N_3; v1.0)$. This indicates that network task N_1 with version 1.0 has one sub network task N_3 with version 1.0. In this way, unexpected changes can be avoided as all network tasks are identified by both name and version.

6. Use RepoX in An Adaptive Workflow Application

6.1 Concept of Adaptive Workflow

The unpredictable and evolving environments make it necessary for the current workflow systems to be able to adapt dynamically. To handle unexpected situations or failures, support for ad-hoc processes and dynamic process models are required [29]. An adaptive workflow system should be able to redesign and make native on-the-fly modifications due to the changing conditions. Tasks executed for some specific purposes in the workflow systems need to have the ability to extend, replace, and re-order [30]. In such a way, new situations and unexpected difficulties can be handled during the execution of workflow instances. Some requirements for an adaptive workflow are as follows:

- be able to capture and maintain knowledge from external environments
- be able to make knowledge-based decisions
- be able to select right resources
- be able to synthesize plans and schedules
- be able to re-plan and re-execute processes
- be able to handle unexpected exceptions

The ORBWork is one of the implementations of the METEOR workflow enactment services. One of the design goals of ORBWork is to provide an enactment framework that is suitable for supporting dynamic and adaptive workflow. The architecture of the enactment system in ORBWork has been designed to support dynamic changes and serve as a platform for conducting research in the area of dynamic and collaborative workflows [21].

6.2 RepoX in Fungal Genome Project

The fungal genome project is sponsored by the Genetics Department of the University of Georgia. The current goals of this project are to create high-resolution of physical and genetic maps, determine the genome's complete DNA sequences, and to identify, map, and determine the functions of all genes [31]. Many tasks, such as experimenting, data analysis, and annotation, can all be integrated in an automated workflow with much less human interference. These tasks can be organized into a workflow to carry out specific functions.

The immediate goal of the RepoX repository project is to help developers to develop workflow applications in the fungal genome project. The Gene workflow as illustrated in Figure 10 is a small part in the whole fungal genome workflow. It first retrieves all the sequence files generated by other workflow applications from a different machine. These sequence files are used to do HTBLAST search against the public database [32]. The searching process may take several hours or longer. The results are

stored in our private database, and the annotation steps can be carried out to analyze the search results.

The RepoX repository provides supports at the design time. There are many different ways to develop workflow applications. It can be developed directly by some script languages like Perl or by a workflow management systems (WfMS's) like ORBWork. The RepoX supports both approaches. After the design time, the RepoX can create some skeleton codes for Perl scripts that follow the logical design of the workflow. The following table compares the ORBWork approach and the Perl's approach to develop workflow applications with the support of RepoX.

With the help of the RepoX repository, the designs of the fungal genome workflow can be kept persistently and developers can review any version of a workflow application freely. Many bioinformatics workflow applications are developed by the language Perl. Because the RepoX repository generates skeleton codes following the logical design of the workflow application, developers can concentrate on developing various applications. Application codes can be integrated in these skeleton codes easily.

6.3 Searching, Querying, and Versioning

The repository stores all the metadata of the workflow systems including process definition, data definition, and role definition. Because adaptive workflow systems have the ability to capture and maintain knowledge of the changing environments to make knowledge-based decisions, the repository should support searching, querying, and retrieving the existing workflow definitions that match those decisions. All these operations can be done through a GUI query tool or an API that directly visits the repository. Any task or network task, which is the result of searching, can be reviewed, modified, or incorporated to replace other task or network task. The modified part of the whole workflow then can be recompiled and re-executed. In this way, all the changes are made in an on-the-fly way to achieve re-planning and problems-fixing goal during

execution time.

The RepoX repository provides full support for searching, querying, and versioning. An XML document can be modeled as a "rooted, directed, ordered, labeled tree" [33]. To access and manipulate the XML document as a tree structure, the Document Object Model (DOM) core interfaces, which provide an open solution for a variety of tools to manage documents, are used in the RepoX. The DOM represents the XML document as a hierarchical tree of elements and attributes. A user can view and browse the XML elements as a tree structure (expanding from the current non-leaf node) using the application that implements the DOM API. Figure 13 gives an example of a DOM tree of a network task called "Gene".

With the help of a navigation tool, a user can select the XML document and then click on the node representing the current object to see the next detailed level until the leaf node level. Figure 14 shows the navigation of the network task "Gene" and a user can browse it using the functions provided by Java JTree. Every non-leaf node can be clicked to show the next detailed level.

A user-friendly GUI tool is developed to make it easy for a user to query the repository. Given some key words such as the task name, all the related versions of this task, already extracted to the repository, will appear on the screen. A user can choose the latest version or any previous version. Figure 15 is a screenshot of the GUI from the client side.

7. Conclusions and Future Work

The repository is the foundation to provide the mechanisms for storage, access, and control management of all the information related to an enterprise. As XML becomes the new universal data format, an XML repository is needed to manage all the XML documents. Typically, an XML repository uses a database as the backend to store XML documents. Software is also needed to transfer data from an XML document to the

database and from the database to an XML document.

In this paper we introduced RepoX, an XML repository that provides such basic functions as storage/retrieval management, version control, configuration management, searching and browsing, as well as support for adaptive workflows. The architecture of RepoX is discussed, and some of it features are presented. The main goal of this project is to develop an XML repository for workflow designs and specifications.

In RepoX, the storage of XML documents is different from some traditional approaches. We developed a modeling process procedure to help users produce a schema from DTD specifications. RepoX is very useful in developing workflow applications. It has a graphical workflow design tool integrated on the client side. At design time, a user can use the tool to design the workflow logically and then export the workflow definitions in the XML document format to the repository. Provided with version and configuration control, a workflow developer can review and make modifications on any version of a workflow application stored in the repository by means of check-in and check-out. The whole design history is kept, which makes it is easy to move back to any previous version. If a workflow application is developed in the Perl language, RepoX can generate skeleton code that follow the workflow logical design. Other applications can be integrated into the skeleton code easily. RepoX is used in the fungal genome project, sponsored by the Genetics Department of the University of Georgia, for developing workflow applications.

Some future work is still left to do. An application with a GUI can be developed to support the online editing and modifications of the specification definitions from different DTD specifications. Another area of improvement relates to the way that versions are stored. For different versions of the same network task workflow definition, only the delta changes should be stored for efficiency. A function to compute the minimum changes between different versions of the same item should also be provided. In this way, a previous version can be computed from the latest version with the reverse

delta changes. In RepoX, a versionable item is a simple task or a network task. Support for versioning with low-level granularity should also be provided. Security issues should also be considered to keep track of users who are currently working with the repository. So far only the user authentication information and the lock table are kept. Access control needs to be provided to limit the access rights of an authenticated user or user group on a given document resource. To support more concurrency, locking should be at the document fragment level rather the whole document. In this way, fragments of an XML document can be checked out and locked independently. Different types of locks, such as read locks and write locks, should also be provided to increase concurrency.

In the RepoX repository, the elements of an XML document are mapped to Java classes by the product called DXML (Dynamical XML) and those elements are stored in tables of a database by the predefined schema for efficient filtering and storage. A native XML database, which exploits the semi-structured data, may be another approach to be considered. An XML database can give fast retrieval and avoid generating schema at runtime for the translation to tables or any other non-XML data structures. It should support the XQuery language [34], which simplifies the processing of the XML documents by utilizing the XML structure.

REFERENCES

[1] Workflow Management Coalition. *Workflow Standards – Interoperability Wf-XML Binding.* Document Number WFMC-TC-1023, pp.8-28, May 2000.

[2] Chengfei Liu, Hui Li, and Maria E Orlowska. *Object-Oriented Design of Repository for Enterprise Workflows.* CRC for Distributed Systems Technology and Computer Science Department, The University of Queensland, 1996.

[3] Carma McClure. *The Three Rs of Software Automation: Re-engineering, Repository, and Reusability.* Prentice Hall, Englewood Cliffs, NJ 07632. pp. 157-160, 1992.

[4] Adrienne Tannenbaum. *Implementing a Corporate Repository: The Models Meet Reality.* John Wiley & Sons, Inc. pp. 275-286, 1994.

[5] Bonnie M. Edwards and John A. Miller. *Implementing and Evaluating Common Repository Services.* UGA-CS Masters Thesis, 1995.

[6] John A. Miller, Walter D. Potter, Krys J. Kochut, Sunderratnan Krishnan, Bonnie Edwards, Wensheng Zhang and Jayesh Sahasi. *Design of a WSRC Repository with an End-User Emphasis.* UGA-CS Technical Report, pp. 50-53, 1994.

[7] Horiuchi Hajime. *Standardization of Information Resource Dictionary System.* IPSJ MAGAZINE, Vol.37 No.07, 2000.

[8] Ronny G. Flatscher. *An Overview of the Architecture of EIA's CASE Data Interchange Format (CDIF).* 1996.

[9] Lois Wakeman and Jonathan Jowett. *PCTE: The Standard for Open Repositories.* Prentice Hall, pp. 1-14, 1993.

[10] Elizabeth Castro. *XML—For the World Wide Web*. Peachpit Press, 2001.

[11] Dirk Bartels and Matthew Gertner. *A Scalable XML Repository. SGML World 1997*, Washington D.C., Nov 1997.

[12] Betty Harvey, Denis Hill, Ron Schuldt, Martin Bryan, Dick Rarman, Gerard Freriks and David Webber. *White Paper on Global XML Repository for XML/EDI*. The XML/EDI Group, pp. 8-14, Feb 1999.

[13] D. Hollingsworth. *The Workflow Reference Model*. Technical Report TC00-1003, Issue 1.1. The Workflow Management Coalition, Brussels, Belgium, pp. 21-27, November 1995.

[14] Yong Jiang and Krys J. Kochut. *The Repository System of Meteor$_2$ Workflow Management System*, UGA-CS Masters Thesis, 1998.

[15] Ketan A. Bhukhanwala and Krys J. Kochut. *jFlow: Workflow Interoperablility for the Meteor Workflow Management System*, UGA-CS Masters Thesis, 1999.

[16] Rob Allen. *Workflow: An Introduction*. Workflow Management Coalition, pp. 24-32, 2001.

[17] Mike Marin. *Workflow Process Definition Interface—XML Process Definition Language*. The Workflow Management Coalition Specification, pp. 15-31, 2001.

[18] N. Krishnakumar and A. Sheth. *Managing Heterogeneous Multi-system Tasks to Support Enterprise-Wide Operations*. The Journal on Distributed and Parallel Database Systems, 3 (2), 1995.

[19] A. Sheth, K. Kochut, J. Miller, D. Worah, S. Das, C. Lin, D. Palaniswami, J. Lynch and I. Shevchenko. *Supporting State-Wide Immunization Tracking Using Multi-Paradigm Workflow Technology*. Proc. of the 22nd Intl. Conf. On Very Large Database (VLDB96), 1996.

[20] D. Worah and A. Sheth. *Transactions in Transactional Workflows*. Advanced Transaction Models and Architectures, S. Jajodia and L. Kerschberg, Eds., Kluwer Academic Publishers, 1997.

[21] Krys J. Kochut, Amit P. Sheth, and John A. Miller. *Optimizing Workflow-Using a CORBA-based, fully distributed process to create scalable, dynamic systems.* Component Strategies, Vol. 1, No. 9, pp. 45-57, 1999.

[22] S. Das, K. Kochut, J. Miller, A. Sheth, and D. Worah. *ORBWork: A Reliable Distributed CORBA-based Workflow Enactment Systems for METEOR.* Technical Report UGA-CS-TR-97-001, University of Georgia, Department of Computer Science, 1997.

[23] John A. Miller, Devanand Palaniswami, Amit P. Sheth, Krys J. Kochut and Harvinder Singh. *WebWork: METEOR2's Web-Based Workflow Management System*. Journal of Intelligent Information Systems, Special Issue on Workflow Management Systems, Vol. 10, No. 2, pp. 185-215, 1997.

[24] I. Budak Arpinar, John Miller, and P. Sheth. *An Efficient Data Extraction and Storage Utility for XML Documents.* Proc of the 39th Annual ACM Southeast Conference (ACMSE'01), Athens, Georgia. pp. 293-295, Mar 2001.

[25] Serge Abiteboul, Peter Buneman, and Dan Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann Publishers, pp. 235-238, 2000.

[26] Kanne, C.-C. and G. Moerkotte. *Efficient Storage of XML Data*. Proc. of the 16th Int. Conf. On Data Engineering (ICDE), San Diego, 2000.

[27] David Whitgift. *Methods and Tools for Software Configuration Management.* John Wiley and Sons Ltd., 1991.

[28] Ulf Asklund, Lars Bendix, Henrik B. Christensen, and Boris Magnusson. *The Unified Extensional Versioning Model.* System Configuration Management, Proc of the 9th International Symposium, SCM-9, pp. 100-122, Sep 1999.

[29] Dragos A. Manolescu and Ralph E.Johnson. *Dynamic Object Model and Adaptive Workflow*. Department of Computer Science, University of Illinois at Urbana-Champaign. Technical Report, 1998.

[30] W.M.P. van der Aalst. *How to handle dynamic change and capture management information? An approach based on generic workflow models.* Department of Information and Technology, Eindhove University of Technology. pp. 5-8, 1999.

[31] David Hall, John A. Miller, Jonathan Arnold, Krys J. Kochut, Amit P. Sheth, and Michael J. Weise. *Using Workflow to Build an Information Management System for a Geographically Distributed Genome Sequence Initiative.* Genomics of Plants and Fungi, R.A. Prade and H.J.Bohner, Editors, 2001.

[32] Nick Camp, Haruna Cofer and Roberto Gomperts. *High-Throughput BLAST*, Whiter Paper, pp. 2-8, Sep 1998.

[33] Frank Tompa. University of Waterloo. *Providing Flexible Access in a Query Language for XML*. QL 98, position paper, 1998.

[34] W3C Working Draft. XQuery: A Query Language for XML, *www.w3.org/TR/2001/WD-xquery-20010215*. Feb, 2001.

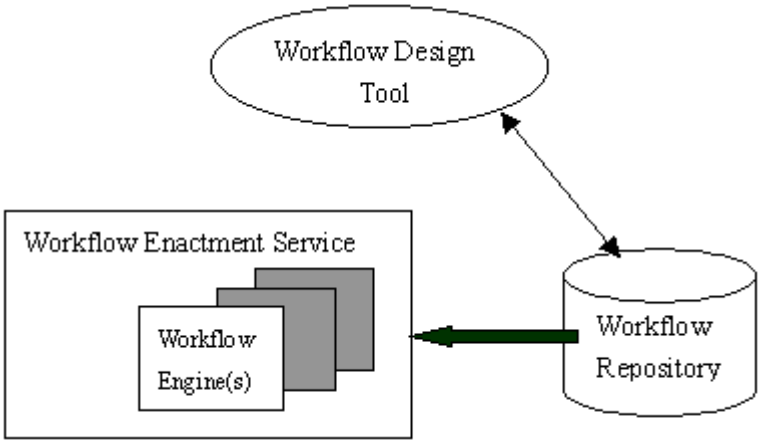

Figure1: Workflow Components Diagram

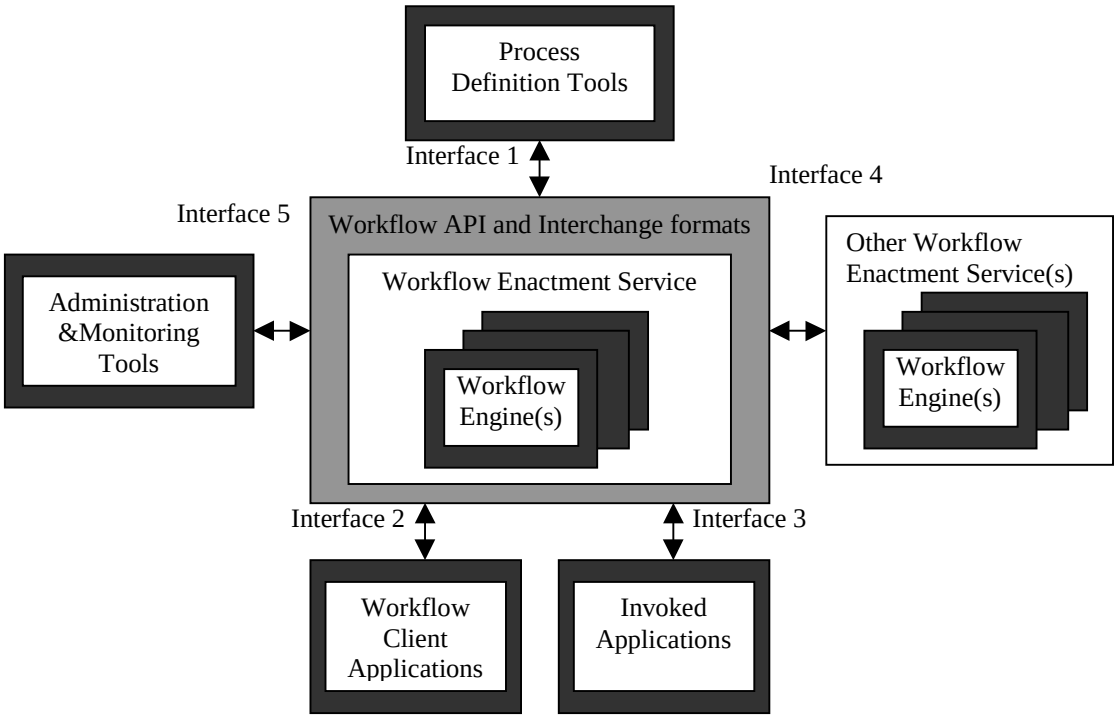

Figure 2: Five Interfaces of Workflow Reference Model [13]

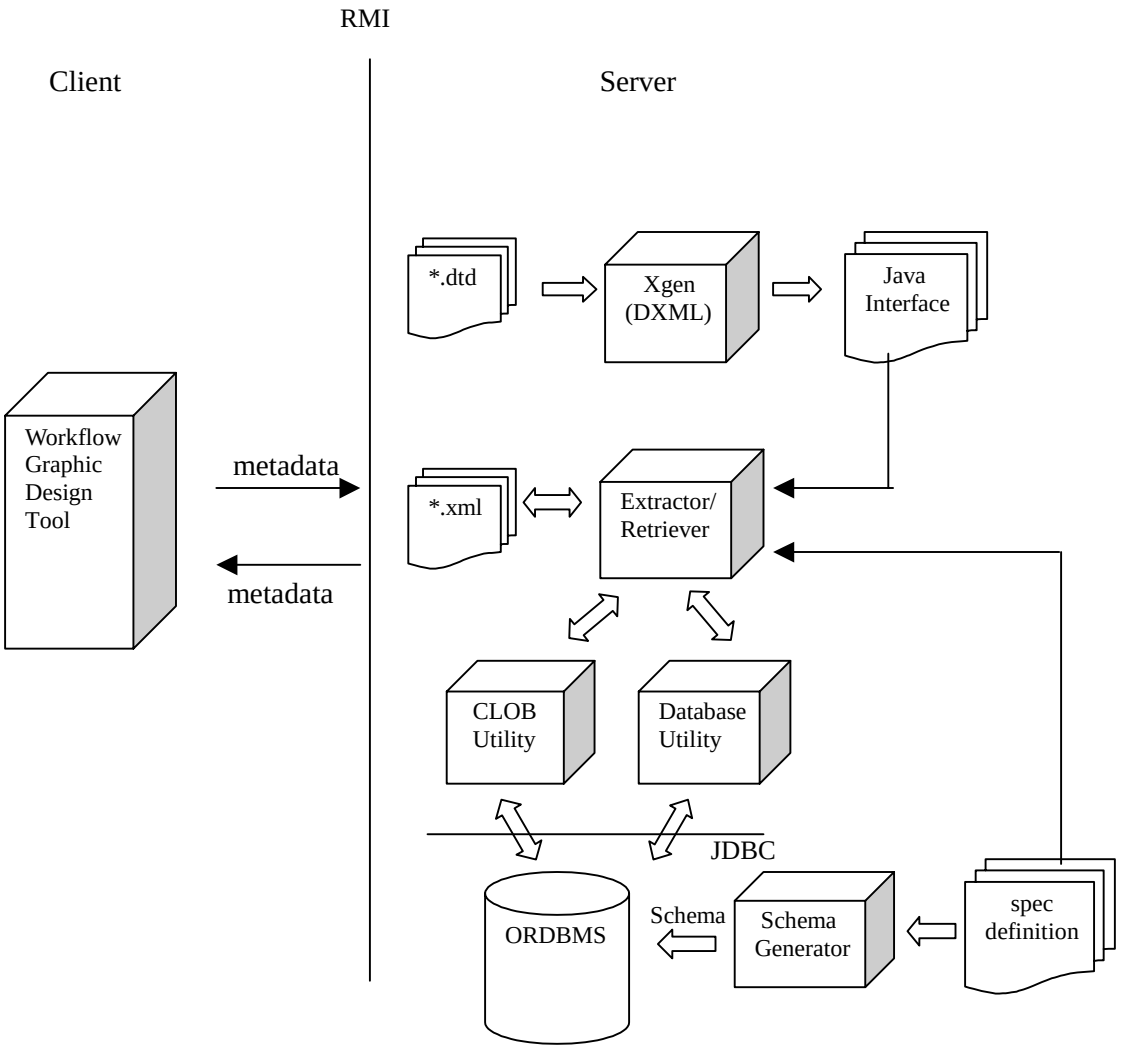

Figure 3: RepoX Repository's Architecture

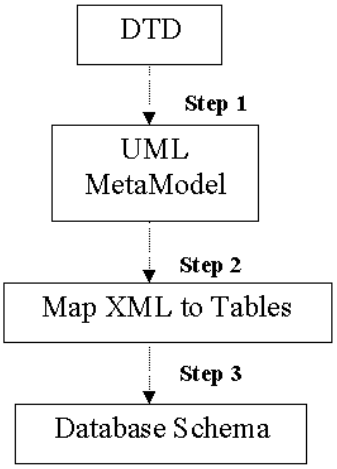

Figure 4: Diagram of Modeling Process to Get Database Schema

```
<!--
    Workflow TaskList Data Type Definition (DTD)
    Revision: 1.2
    Date:  March 08 1999
 -->
        <!ELEMENT NetworkTask (Task, SimpleSubTaskList)>
 <!-- Task describes NetworkTask
 -->
 <!ATTLIST NetworkTask id ID #REQUIRED
                  date CDATA #IMPLIED>
        <!ELEMENT Task (Name, Description, TaskType, Host, ForeignTask)>
 <!ATTLIST Task id ID #REQUIRED>
 <!ELEMENT SimpleSubTaskList (Task)*>
 <!ELEMENT Name (#PCDATA)>
 <!ELEMENT Description (#PCDATA)>
 <!ELEMENT TaskType (#PCDATA)>
 <!ELEMENT Host (#PCDATA)>
<!ELEMENT ForeignTask (#PCDATA)>
```

Figure 5: A Simplified DTD of Network Task

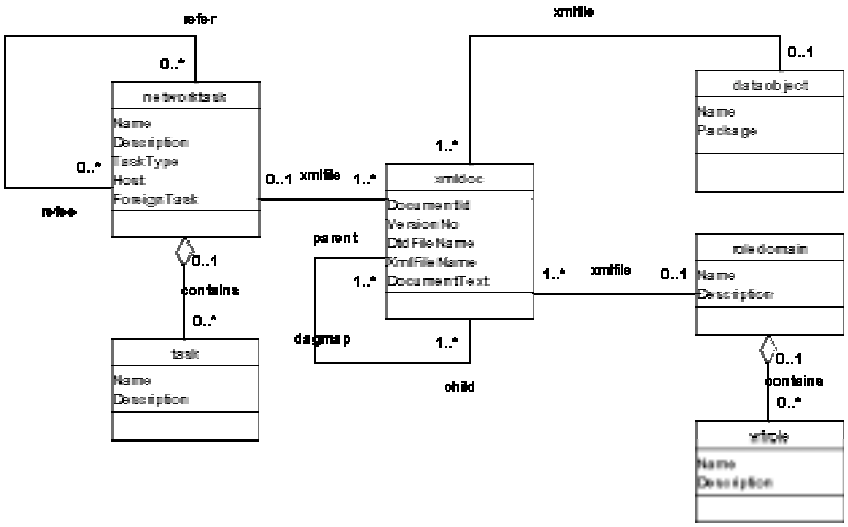

Figure 6: Meta Model in RepoX

```
public static final String [] TABLE = {
            "xmldoc",
            "networktask",
            "task",
            "taskref",
            "dataobject",
            "roledomain",
            "wfrole"
    };
```

Figure 7: Tables Defined in MapXML2Tables.java

```
CREATE TABLE xmldoc (
        DocumentId      int     not null        auto_increment,
        VersionNo       int     not null,
        DtdFileName     varchar (254),
        XmlFileName     varchar (254),
        DocumentText    longblob,
        primary key (DocumentId, VersionNo)

   );

CREATE TABLE task (
        Name            varchar (64)    not null,
        Description     varchar (64),
        TaskType        varchar (64),
        Host            varchar (64),
        ForeignTask     varchar (64),
        xmldoc_DocumentId       int     not null,
        xmldoc_VersionNo        int,
        primary key (Name, xmldoc_DocumentId),
        foreign key (xmldoc_DocumentId, xmldoc_VersionNo)
        references xmldoc (DocumentId, VersionNo)
  );
```

Figure 8: Definitions of "xmldoc" and "task"

```
<?xml version="1.0"?>
 <!DOCTYPE NetworkTask SYSTEM "NetworkTask.dtd">
 <NetworkTask id="NetworkTask_HTBLAST">
      <Task id="HTBLAST">
      <Name>HTBLAST</Name>
      <Description>do HTBLAST search</Description>
      <TaskType>Non-transactional WorkFlow</TaskType>
                  <Host></Host>
       <ForeignTask>false</ForeignTask>
      </Task>
      <SimpleSubTaskList>
        <Task id="start">
          <Name>start</Name>
          <Description>start HTBLAST</Description>
          <TaskType>Simple Non-transactional Task</TaskType>
           <Host></Host>
           <ForeignTask>false</ForeignTask>
        </Task>
      </SimpleSubTaskList>
   </NetworkTask>
```

Figure 9: An Example of XML Document for Network Task

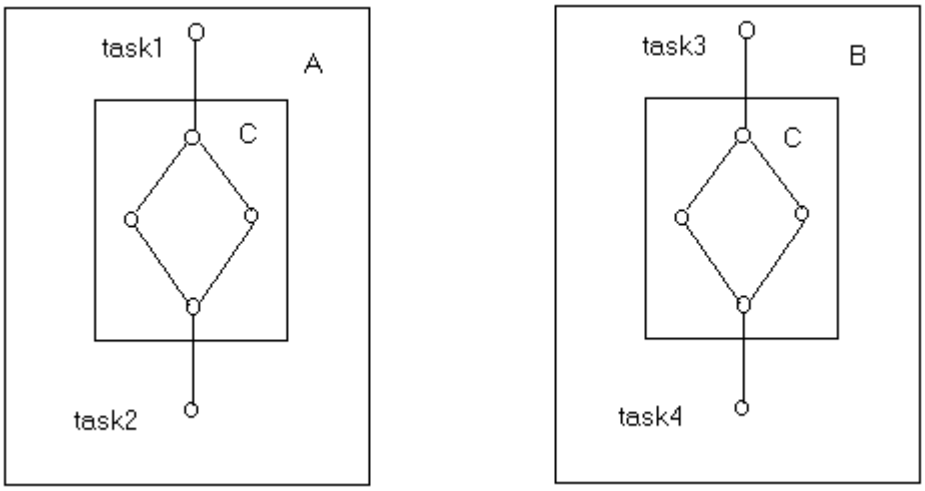

Figure 10: Network Task C both in Network Task A and B

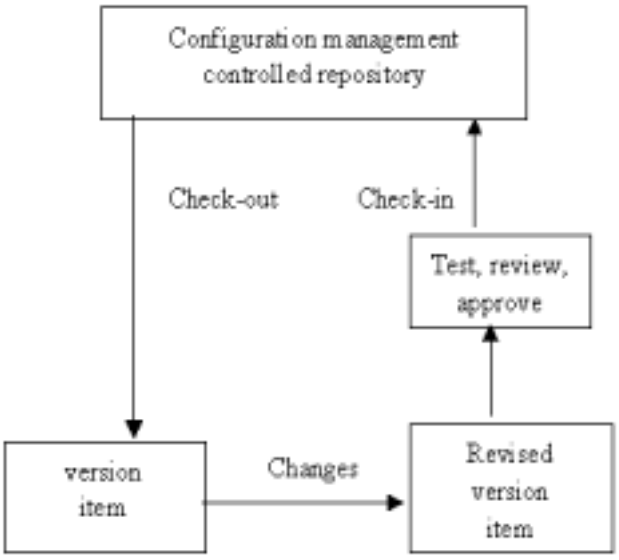

Figure 11: Check in and Check out

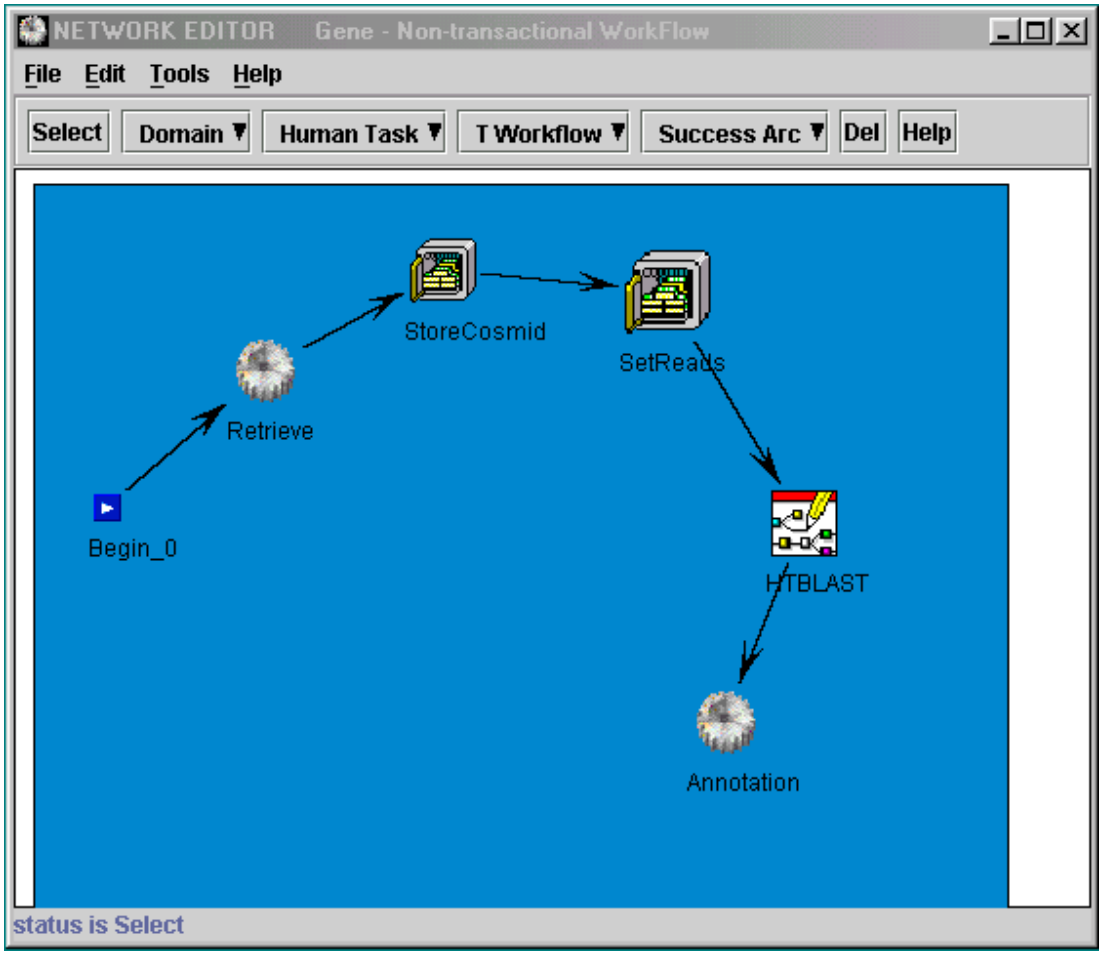

Figure 12: Gene Workflow

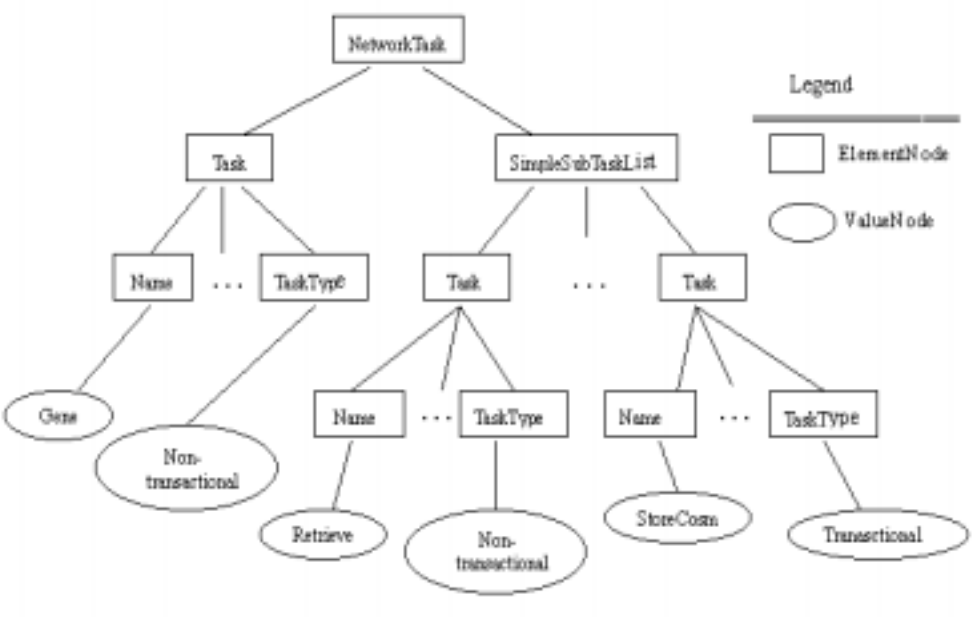

Figure 13: DOM Tree of Network Task

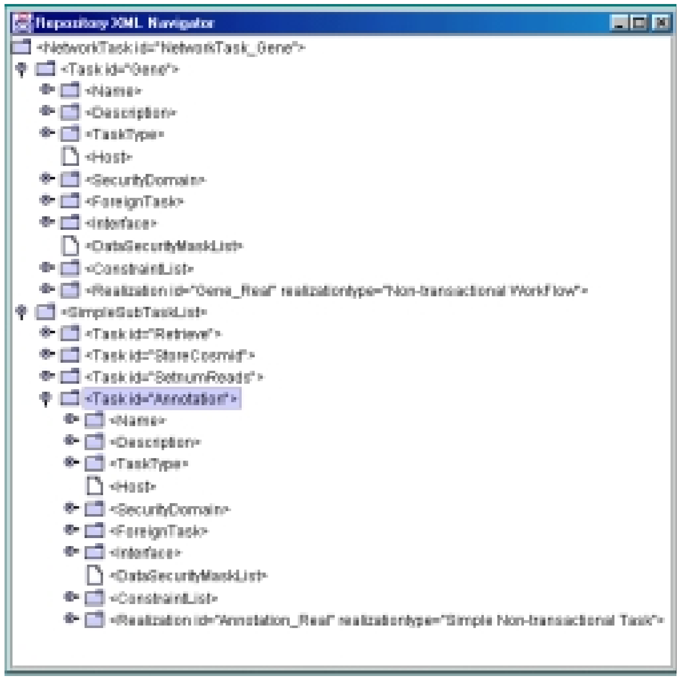

Figure 14: Navigate Network Task “Gene”

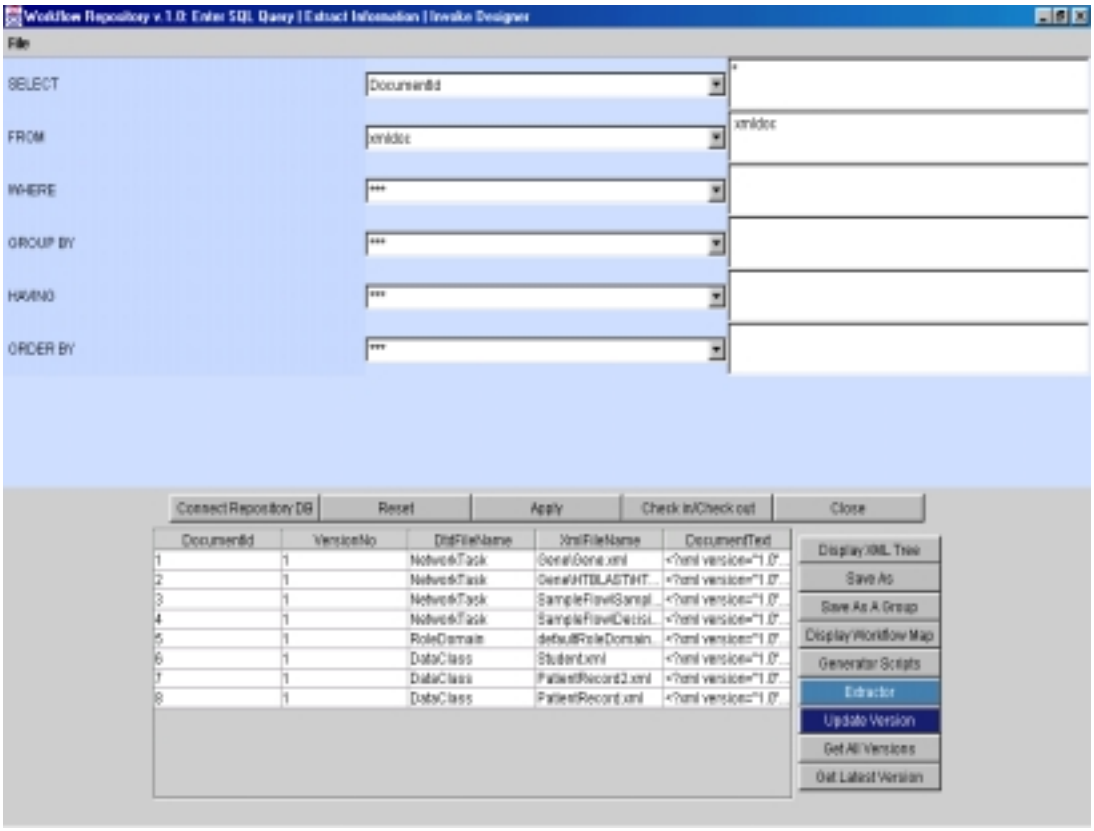

Figure15: Screenshot of GUI at the Client Side

Approach	Advantages	Drawbacks
A simple file	1. Free technology 2. Simple 3. Available on all systems	1. Need parsing to generate XML structure for every access 2. No query searching 3. No concurrent access control 4. No scalability
A relational database	1. Provide DBMS functionality and reliability 2. Widely available 3. Standardized query (SQL)	1. No support for references, hierarchical data structures and multiple value fields 2.Expensive mapping from XML model to relational model 3. Scalability hampered
An Object solution	1. Provide DBMS functionality and reliability 2. Standardized (ODMG) 3. Better distributed support (*e.g.*. CORBA, OID) 4. Support for query and navigation access 5. Scalability	1. prone to deterioration in performance for large data volume

Table 1. Comparison of Traditional XML Repository Storage Approaches

	ORBWork	Perl
Design	Same	Same
Generate Code	.spec and .java files	skeleton files in Perl
Custom Coding	Special tasks	Bulk of coding
Compile	Built-in the design tool	Perl compiler or interpreter
Installation	Packaged and only need one installation command	By developer
Run	Can be invoked from browser to choose from task list	Run *.pl to start the workflow

Table 2: Comparison of ORBWork and Perl's Approaches