AUTOMATIC WORKFLOW PROCESS AND DATA ANNOTATION: DESIGN AND EXECUTION ENVIRONMENT

by

AMRITA BASU

(Under the Direction of Krzysztof J. Kochut)

ABSTRACT

This thesis presents an ontology-based approach to automatic annotation of workflow data. Although much work has been done on the annotation of data found on the Web, little work has been done on the subject of annotation of workflow data, in particular data manipulated by scientific workflows. Such workflows generally try to automate experimental processes which are data and computation intensive. In order to correlate among the vast array of data, as well as for machines to process the data, we need to annotate the data with suitable terms and relationships from the vocabulary of a knowledge describing the protocols of the experiments producing the data. In the Semantic Web terminology, this vocabulary is called the ontology. In this thesis, we introduce a way of annotating experimental data produced from scientific workflows using a web-based annotation definition tool called TOAD (Tool for Ontology based Annotation of workflow Data).

INDEX WORDS: Semantic Web, Ontology, RDF, N3 notation, WFMS, qRT-PCR

AUTOMATIC WORKFLOW PROCESS AND DATA ANNOTATION: DESIGN AND EXECUTION ENVIRONMENT

by

AMRITA BASU

B.A., Jadavpur University, India 2000

M.C.A., Birla Institute of Technology, India, 2003

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF COMPUTER SCIENCE

ATHENS, GEORGIA

2008

AUTOMATIC WORKFLOW PROCESS AND DATA ANNOTATION: DESIGN AND EXECUTION ENVIRONMENT

by

AMRITA BASU

Major Professor:	Krzysztof J. Kochut
Committee:	John A. Miller William York

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
December 2008

DEDICATION

I would like to dedicate this thesis to my husband Akhilesh, my parents Debabrata and Apan and my brother Soumyabrata. I would have never been able to complete this thesis without their love and support. A very special thanks to my husband without whose continuous support, encouragement and cooperation, this work would not have been possible.

ACKNOWLEDGEMENTS

I am extremely thankful to my major professor, Krzysztof J. Kochut for his invaluable guidance and encouragement throughout my research and academic study in UGA. I would also like to thank Dr. John A. Miller and Dr. William York for serving on my advisory committee, and for their advice and guidance in my research.

I would like to thank my lab mates: Maciej Janik, Matthew Eavenson, Arpan Sharma and my friends Ajith Ranabahu, Kavitha Anandan and Sohini Sahu for their help and support throughout my period of study and research at UGA.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

The technological developments of the last few decades, including the development of the World Wide Web, have given us access to far more information than we can reasonably manage. Although the amount of information available within World Wide Web is huge by and large, it is not computer-understandable to easily integrate the various pieces of information that a user really needs. Today, most of the Web information is represented in natural language, making it human but not machine understandable. With the astronomical rise in the volume of information in the Web, attention is being shifted from knowledge acquisition to knowledge management and understanding [1]. Also, important has become the need for integration of information. Only through information integration we can have a common view of all the data and understand their relationships. For the successful implementation of information management and integration, the underlying information needs to be described in a machine processable form. This is the fundamental idea behind Semantic Web, which originated at the end of 1990s [2]. The Semantic Web refers to the evolving Web. "The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation" [2].

The fundamental goal of Semantic Web is the creation and use of semantic metadata. Metadata as the name suggests is defined as "data about data". The goal of incorporating metadata into data sources is to enable the end-user and even more importantly a computer

program to find items and contextually relevant information[3]. Data sources can be structured, semi-structured and unstructured. The process of associating metadata with structured or unstructured data resources (audio, video, web pages, images, spreadsheets etc) is called annotation. Semantic annotation is the process of annotating resources with semantic metadata. Semantic metadata adds relationships, rules, and constraints to data. This metadata describes contextually relevant information about content based on a domain specific metadata model, providing a context for data interpretation. In the Semantic Web terminology such a metadata model is called the Ontology. Ontology according to Gruber is defined as specification of a conceptualization [4]. It defines a concept, its properties and the relationships that exist with other concepts. Thus, ontology captures the meaning associated with the content. If a formal ontology is used for describing the metadata associated with any data, then the data lends itself to machine processability and higher degree of automation. Such machine processable formal annotations on Web resources can result in improved search capabilities and unambiguous resource discoveries.

Although much work has been done in the area of semantic annotation in the Web, not much work has been done on such annotation of data produced from a scientific workflow. A workflow is concerned with the automation of processes where documents, information or tasks are passed between participants, according to a defined set of rules to achieve, or contribute to an overall business goal. A Workflow Management System (WFMS) is one which provides procedural automation of a business process by management of the sequence of work activities and the invocation of appropriate human and/or IT resources associated with the various activity steps [5]. In recent times with more and more availability of scientific data available on the Internet, such as databases of various genes and enzymes, proteins and metabolic pathways

references, scientists have started using workflows for their analysis and computational tasks. Scientific workflow is a new special type of workflow that underlies many large-scale complex e-science (large scale science increasingly being carried out in interdisciplinary fields through distributed collaborations enabled by the Internet) applications run in various problem solving areas as diverse as computational biology, chemistry, genetics, astronomy, chemistry, bio-science and physics. A scientific workflow is the description of a process, often completely automated, that specifies the synchronized execution of multiple tasks. These tasks are software programs, run locally or remotely and increasingly published as Web services [6]. Compared with business workflow, scientific workflow has distinct features such as being data and computation intensive, requiring less human interaction and involving a large number of activities, often required to be done repetitively. Scientific workflow technologies have emerged as a problem-solving environment for researchers by facilitating the creation and execution of experiments given a pool of available data and computation services [7]. Such workflows run on processes which rely on acquiring, managing, storing and ultimately analysing huge volume of data. The analysis is expensive in terms of data as well as time required to run the process, sometimes taking days to complete as well as constant monitoring. These processes are also often repetitive, since scientists often have to re run their experiment with different settings. Automating these computational intensive processes using scientific workflows would not only make the process easier by minimizing user supervision but would also allow more systematic storage and retrieval of data. One problem, however, with these scientific workflows is the interpretation of the massive amount of data they produce. Annotating the data produced from these scientific workflows using metadata would help to make better sense of the data being produced from the workflow, as well as understand correlations existing within data.

This thesis introduces an ontology based approach of annotating scientific workflow data using semantic annotation. A Web based annotation definition tool called TOAD (Tool for Ontology based Annotation of workflow Data) has been developed for the purpose of annotating the data. A detailed description has been provided to utilize this tool for annotating data produced from scientific workflows. As a case study, we use the quantitative real-time reverse transcriptase-polymerase chain reaction (qRT-PCR) [8], a complex biological research process for the regulation of glycan structures in animal tissues. A semi automated WFMS was developed for the qRT-PCR process using jBPM (Java Business Process Management) [9] which is an open source WFMS. Using an ontology which describes the concepts and relationships that exists in the qRT-PCR reaction, the data produced from the qRT-PCR workflow was annotated with the help of TOAD. TOAD produces an Annotation Definition File (ADF) which contains the mapping definitions between workflow data and corresponding ontology concepts. As the workflow runs, there are some special annotation tasks which are called from the workflow and automatically annotate the workflow data with concepts from the ontology using mapping definition from the ADF.

The rest of the thesis is organized as follows. Chapter 2 provides a background and literature review on workflows and processes, scientific workflows, Workflow Management Systems (WFMS) with special emphasis on the architecture and implementation of jBPM, It also presents a brief overview of Semantic Web, ontologies and previous work done on semantic annotation of data. Chapter 3 defines the workflow annotation process, where we describe how data is defined in a workflow, the role of ontology in such annotations and, how we can create a mapping between the workflow data and the ontology for the purpose of annotating the data

produced from the workflow. Chapter 4 discusses the user interface and the implementation issues of the TOAD tool we have developed for the purpose of creating workflow data annotation definitions. Chapter 5 discusses a case study where experimental data produced from qRT-PCR workflow is annotated using the TOAD. Conclusions and future work are provided in Chapter 6.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

This chapter gives a brief description of the concepts which were applied in this thesis. These include workflows and business processes, Workflow Management Systems, Semantic Web and finally a brief review of data annotations developed so far.

2.1 Workflows and Business Processes

Business processes refer to a collection of activities that takes one or more inputs and creates an output that is value to the customer [10]. The process model describes the structure of a business process in the real world [11]. A process model is a template from which an instance of the process model is created which is simply referred to as a process. Processes involve both automated (computerized) and manual tasks. As shown in Figure 1, the parts that run on computer are called a workflow model. Thus, a workflow can be defined as computerized facilitation or automation of a business process, in whole or in part [5]. Traditional infrastructures are too static and inflexible to adequately reflect rapid changes in business process. Business processes are constantly evolving [12]. Workflow management copes with the business process reengineering in a dynamic environment.

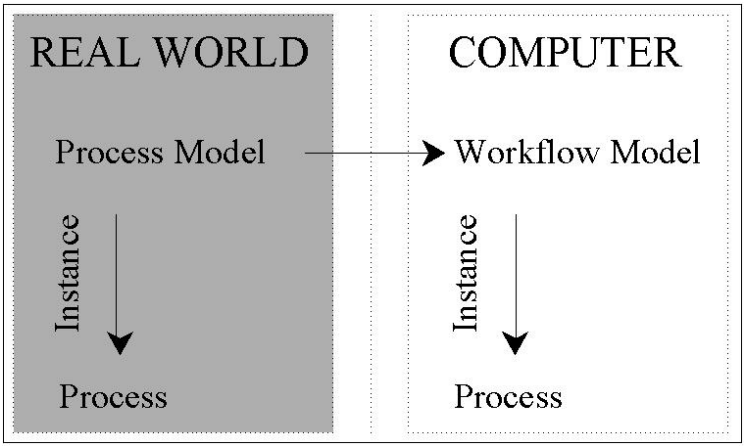


Figure 1: Process vs. Workflow

2.2 Scientific Workflows

Scientific workflow is a new special type of workflow that underlies many large-scale complex e-science applications run in various problem solving areas as diverse as computational biology, chemistry, genetics, astronomy, chemistry, bio-science and physics. In the field of biological sciences, the advent of high-throughput experimental technologies and the growth of publicly accessible biological databases such as KEGG [13] [14], SweetDB [15] and CarbBank [16] to name a few, have greatly increased the amount of data available to the biologists. Analyzing this data effectively has become a significant challenge, which is being increasingly addressed by the use of scientific workflows. They provide biologists with interfaces to databases and powerful tools to do their analysis and computational tasks. A scientific workflow is the description of a process, often completely automated, that specifies the synchronized execution of multiple tasks [6]. These tasks are software programs, run locally or remotely and increasingly published as web services [17]. It is a software model of the way scientists work with their tools and data.

While today's e-science workflow technologies have originated from the business workflows, they have their own distinct features as well. Compared to business workflows, scientific workflows are data and computation intensive, requiring less human interaction and involving a large number of activities, often required to be done repetitively. While business workflows are mainly control and task oriented concentrating on scheduling and task executions, scientific workflows are dataflow oriented, mostly concerned with the throughput of data through highly specialized algorithms, applications and services. Scientific workflows run on processes which rely on acquiring, managing, storing and ultimately analysing huge volume of data. The analysis is expensive in terms of data as well as time required to run the process, sometimes taking days to complete as well as constant monitoring. The processes are also often repetitive, since scientists often have to re-run their experiment with different settings. Ideally, a scientific workflow should offer the scientist:

- The ability to repeat complex operations by allowing changes in parameter settings as and when required, using automation.
- The ability to plug-in any scientific data resource and computational service, either as a local program or in the form of a web-service.
- The ability to reuse existing resources and applications and avoid the details of executing a third party application [18]. The coding of individual modules is an old approach now [19] . The challenge here is in orchestrating and combining pieces of software [19].
- The ability to inspect and visualize data as it is computed, as well as capture enough metadata in the end result. This would allow more accurate record of the scientific process and lead to transparent and reproducible science [20].

Thus scientific workflow technologies should have the ability to emerge as a problem-solving environment for scientists [20].

Scientific workflows are run by Scientific Workflow Management Systems. In Section 2.4 of this chapter, a brief discussion of Scientific Workflow Management Systems is provided. A few of the open source scientific workflows are Kepler [21], Taverna [22] and Triana [23].

2.3 Workflow Management Systems with Special Reference to jBPM

The Workflow Management Coalition (WFMC) is a nonprofit organization founded in 1993 by various vendors and users of Workflow Management Systems (WFMS) which aims at standardizing Workflow Management Systems. Such standardization leads to interoperability of different implementation of WFMS.

According to the WFMC, "A Workflow Management System is one which provides procedural automation of a business process by management of the sequence of work activities and the invocation of appropriate human and/or IT resources associated with the various activity steps." [5]

It goes on to define a Workflow Management System as "A system that completely defines, manages and executes "workflows" through the execution of software whose order of execution is driven by a computer representation of the workflow logic." The WFMC Reference model characterizes WFMS as providing support in three functional areas:

- the Build-time components of a WFMS, concerned with defining, and possibly modeling, the workflow process and its constituent activities,

- the Run-time control components of a WFMS, concerned with managing the workflow processes in an operational environment and sequencing the various activities to be handled as part of each process,
- the Run-time interactions components of a WFMS, concerned with human users and IT application tools that runs the actual business process.

The Workflow Management Coalition's architecture model for a Workflow Management System is called the *workflow reference model*. The WFMC reference model provides a framework for workflow systems identifying their characteristics, functions and interfaces [24], as shown in Figure 2 [5].

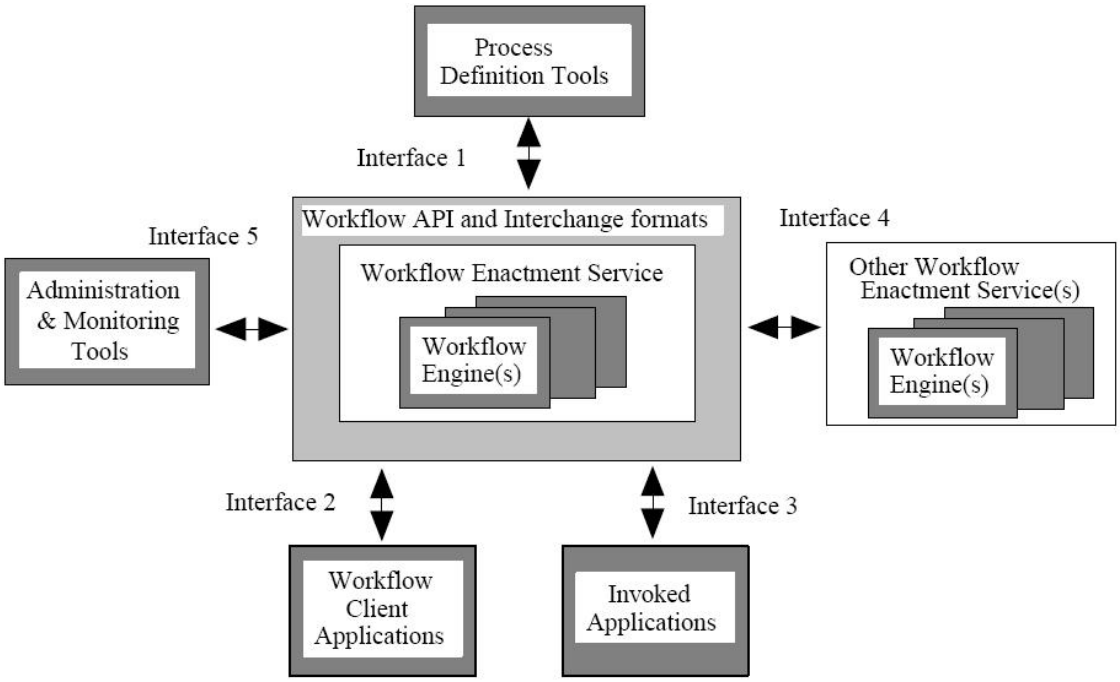


Figure 2: WFMC Reference Model

In order for different Workflow Management Systems to interoperate, The Workflow Management Coalition standardizes a set of interfaces that defines the functions supported by the Workflow Management System [11]. A particular implementation of the WFMS is called a

workflow enactment service. This service runs the workflows. It may contain one or more workflow engines. The focus has been on the five interfaces which surround the workflow engine(s).

Interface 1, as shown in Figure 2 standardizes how different systems can exchange process model and organization information. It allows the business modeling tools to provide the appropriate input to the WFMS [11]. The nature of the interface is an interchange format and API calls, which can support the exchange of process definition information over a variety of physical or electronic interchange media [5]. The process definition language which we have used in the qRT-PCR workflow is called JPDL (jBPM Process Definition Language) [25]. JPDL specifies an xml schema and the mechanism to package all the process definition related files into a process archive which can then be deployed on the server. Interface 2, as shown in Figure 2 consists of a standard set of APIs which may be used in a consistent manner for access from a workflow application to the workflow engine and worklist, irrespective of the nature of actual product implementation [5]. The worklist handler may be supplied as part of a workflow management product or may be written by a user. In our case it has been supplied by the WFMS.

Interface 3, of Figure 2 standardizes the invocation of applications which directly interact with the workflow engine. Interface 4 defines the handling of sub processes between different Workflow Management Systems. Finally Interface 5 defines the structure of the audit trail and the different entries that a Workflow Management System must be able to provide to be compliant [11].

2.4 Scientific Workflow Management Systems

With the progresses in business workflow systems and in Problem Solving Environments (PSE), and in particular by the rapid development in Grid environments [26], a Scientific Workflow Management System (SWMS) provides a meta environment for managing activities and data in scientific experiments, for modeling the dependencies between experimental processes, for prototyping experimental computing systems and for orchestrating the runtime system behavior [27].

Grids [26] have emerged as a global cyber-infrastructure for the next-generation of e-Science applications by integrating large-scale, distributed and heterogeneous resources [28]. In various e-science communities, such as geophysics, astronomy and bioinformatics, Grids are being utilized to share, manage and process large data sets. A Grid infrastructure makes data and computing intensive experiments feasible in PSEs but also requires the management of workflow to support dynamics of the flow execution. Scientific workflow helps automate a scientific process in which tasks are structured based on their control and data dependency. Files and data are passed between participants according to a defined set of rules to achieve an overall goal [5]. Imposing the workflow paradigm for application composition on allows dynamic applications which orchestrate distributed resources as well as integration of multiple teams involved in managing of different parts of the experiment workflow thereby promoting inter-organizational collaborations. SWMS automates activities in the scale of an entire scientific experiment, not only does it automates machine tasks, but also user defined tasks. It includes human users in the runtime loop of a flow execution, and allows an engine to flexibly orchestrate a workflow according to the human decision and the run-time states of the environment [29] .

Thus a SWMS should be able to do the following:

- Manage the entire experiment life cycle.
- Automate experiment routines as well as allow the workflow engine to be orchestrated according to human decisions.
- Rapidly prototype an experiment into a computer system.
- Hide integration details between resources from the end user.

There is no scientific workflow system adopted by all. A few of the open source Scientific Workflow Management Systems are Kepler [21], Taverna [22] and Triana [23].

Taverna is a tool developed by the myGrid [30] project to support 'in silico' experimentation in the bio-informatics field. It is a web-service based SWMS. It is based on Scufl (Simple conceptual unified flow language) [31] language which is a web service based language.

Kepler is another SWMS which uses the MOML (Modelling Markup Language) [32] reference. It uses the Actor-director model for creation and execution of workflows.

Triana is a distributed workflow system which uses the concept of "components". Components are proxies which can represent a java object, a workflow, a Web Service or any distributed file [33]. It supports both data and control flow. Users interact with the system using a task graph.

jBPM, an open source Workflow Management System based on J2EE technology, has been used for developing workflow processes in this research. It supports both business as well as scientific workflow management systems. The following section discusses jBPM in detail.

2.5 jBPM

This section briefly describes jBPM, its architecture, and how workflows are designed, deployed and executed using jBPM.

jBPM is an open source, flexible and extensible Workflow Management System based on the WFMC reference model. jBPM helps in developing workflows with industry-standard orchestration using Business Process Execution Language (BPEL) [34], a flexible and pluggable API, a native process definition language called jPDL, and a graphical modeling tool.
Figure 3 shows the overall jBPM architecture.

jBPM contains the following main components as shown in Figure 3:

- A workflow engine called JBoss jBPM core component which handles the execution of process instances.
- A graph based process definition tool called JBoss jBPM Graphical Process Designer [35]. It is a plug-in to Eclipse [36], which provides support for defining processes in jPDL both in a graphical format as well as in XML format. jPDL (jBPM Process Definition Language) is the process language utilized by the system.
- JBoss jBPM console Web application which has two functions. It is a Web based workflow client whereby, in Home mode, users can initiate and execute processes (as shown in Figure 6). It is also an administration and monitoring tool, which offers a Monitoring mode where users can observe and intervene in executing process instances.
- JBoss jBPM identity component which manages of the definition of organizational information, such as users, groups and roles to which different tasks can be assigned. Currently the definition of all this information is done through the standard SQL insert statements directly into the workflow database [37].

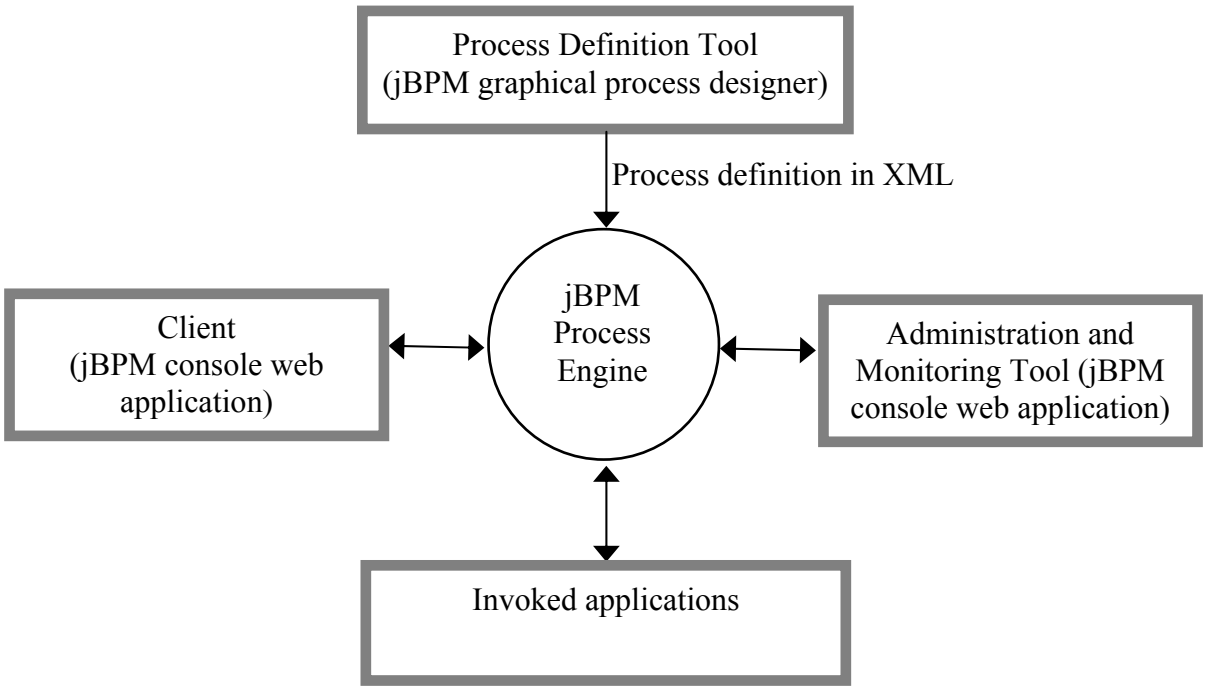


Figure 3: jBPM Architecture

In jBPM, processes are created by first defining them in files called process definition files using JBoss process definition language also known as jPDL. jPDL is a graphic-oriented programming (GOP) language based on a model of nodes, transitions, and actions. In this model, nodes are commands executed as they are encountered during the flow of a process definition. Transitions direct the flow of execution of a process definition, and actions perform specific logic as a node or transition event occurs [38]. Actions are written in action handlers which are instances of Java code that interact with external systems when executed.

In jBPM, process definitions are packaged as process archives. A process archive is a zip file. The main file in the process archive is the processdefinition.xml which contains the formal description of the processes. An example of a processdefinition.xml file is given in Figure 4. A

process definition is based on a directed graph. The graph is composed of one start state and one end state, nodes, and transitions.

```
<?xml version="1.0" encoding="UTF-8"?>
<process-definition
  xmlns="urn:jbpm.org:jpdl-3.1"  name="PCR">
   <swimlane name="initiator1"></swimlane>
   <event type="process-start">
      <action name="action1"
class="com.sample.action.MMActionHandler"></action>
   </event>
   <start-state name="Select Well Template">
      <task name="Well Assignment" swimlane="initiator1">
         <controller>
            <variable name="A01-A03"></variable>
            <variable name="A04-A06"></variable>
            <variable name="A07-A09"></variable>
            <variable name="H10-H12"
access="read,write,required"></variable>
         </controller>
      </task>
      <event type="node-leave">
           <action name="action1"
class="com.sample.action.FileNameActionHandler"></action>
       </event>
      <transition name="toPCR" to="Start PCR Reaction"></transition>
   </start-state>
   <task-node name="Start PCR Reaction">
      <task name="PCR Reaction" swimlane="initiator1">
         <controller>
         <variable name="FileName" access="read" mapped-name="Name
to be entered in PCR">
         </variable>
         </controller>
      </task>
```

Figure 4: jPDL Description of the Process

Process definitions are created easily with the jBPM graphical modeling designer. The designer is installed as an Eclipse plug-in. Figure 5 illustrates a sample screen from the graphical modeling designer. The graphical designer can be used to create process definitions, attach action handlers to events, create process archives, test process definitions, and so on.

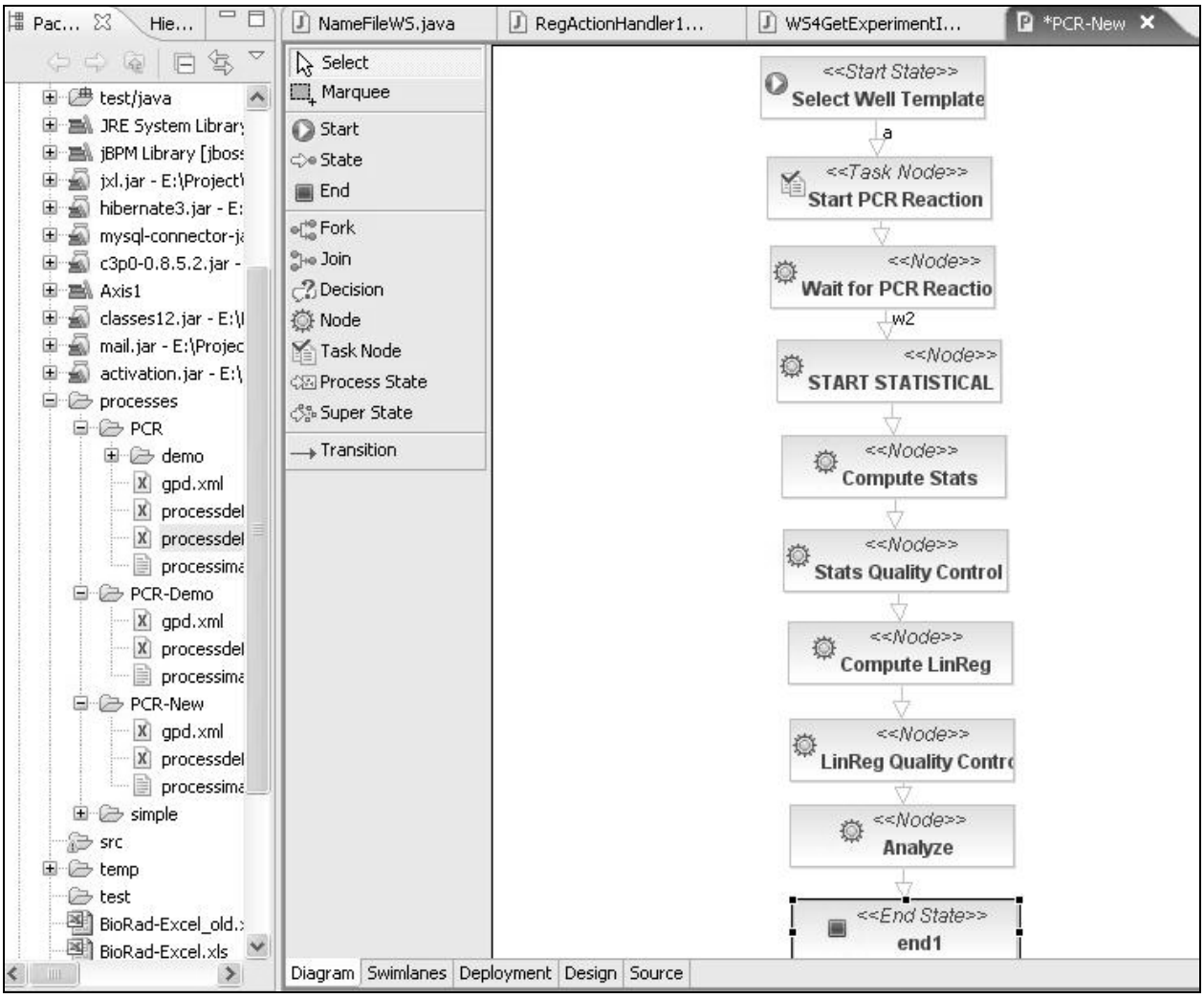


Figure 5: Graphical Representation of the jPDL Process in Figure 4

Once the process definitions get defined they need to be deployed, so that the server can run the workflow. jBPM does persistent storage of process definitions as well as process executions in a database. Therefore, deploying a process into JBoss jBPM involves parsing the process-definition.xml and storing it in the JBoss jBPM database. The process archive is deployed directly from the process designer tool.

While a process definition executes, a process archive is passed to the jPDL process engine for execution. The jPDL process engine finds the process definition file packed in the process archive, traverses the process graph, executes defined actions, maintains process state, and logs all process events. Figure 6 shows the jBPM web console once the workflow has been designed, deployed and executed.

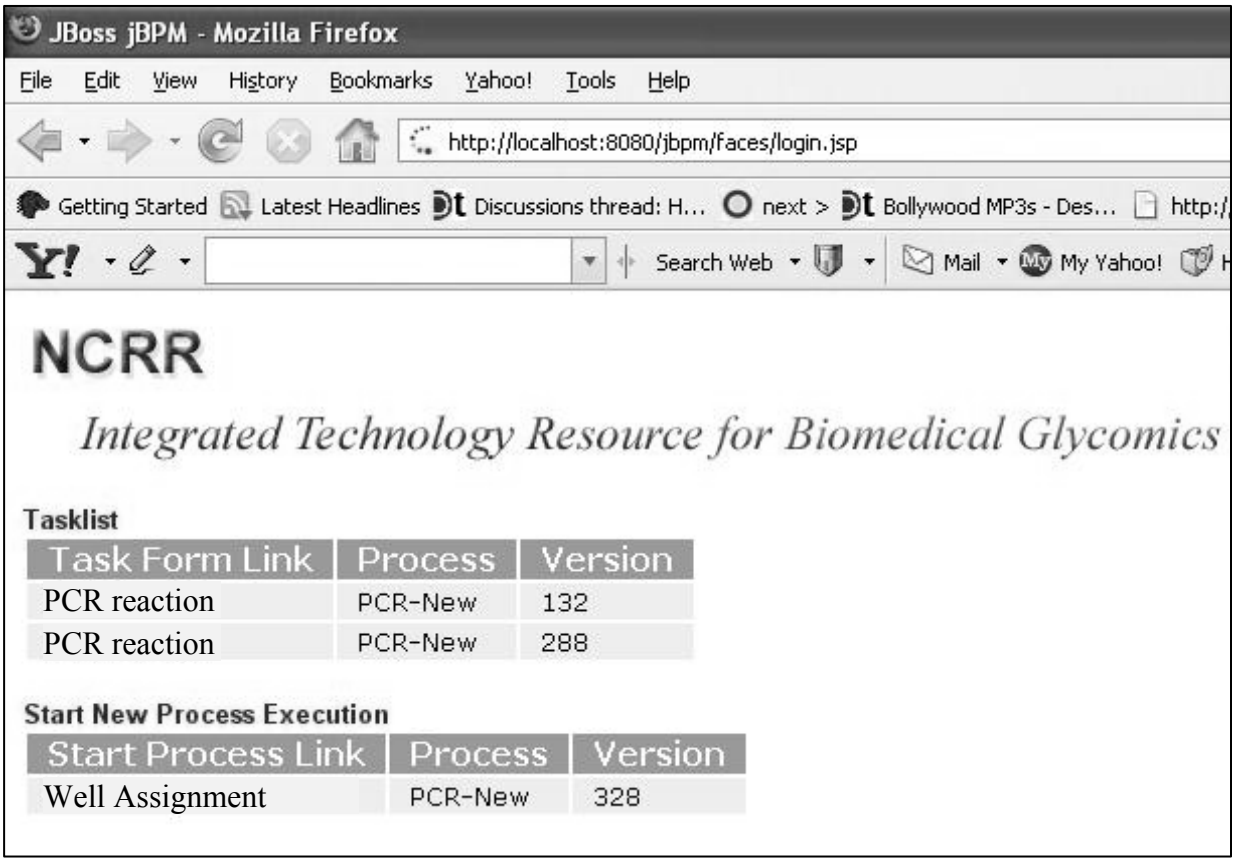


Figure 6: jBPM Web Console

2.6 Semantic Web

An academic study has shown that the volume of information in the public Web has tripled between 2000 and 2003 [39]. At this rate, we soon will need advanced techniques to help us make sense of the huge amount of data, to find what we need and filter out the rest. As a

consequence, attention has shifted in recent years from knowledge acquisition to knowledge management [1].

Just as knowledge management is becoming important, so is the need for integration becoming a key challenge for IT managers [40]. They not only need to integrate, but also "makes sense" out of different set of inter-related data components. They not only need to integrate information within their organization, but also to "make sense" out of the different set of inter-related data components. The first version of the Web, namely the World Wide Web, was meant mainly for human consumption. Semantic web on the other hand aims at adding semantics to the contents of World Wide Web, to make it understandable by the computer. According to Tim Berners-Lee, the father of Web, The "The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation" [2]. This would enable machines to synthesize information from the vast platitude of data already available and would finally lead to automation of tasks. The Semantic Web provides a common framework that allows data to be shared and reused across different applications, enterprises, and community boundaries.

The fundamental goal of Semantic Web is the creation and use of semantic metadata. This metadata tells us about the contents of a document, it may either describe a document, say a web page, or some entities within the document, for example a person. This is in sharp contrast to the metadata which exists in today's Web, encoded in HTML. It merely describes the format in which information can be presented, for example we can specify whether a given string should be presented in bold or not, but not what that string means, that is, whether it means the name of a person or name of a product and so on.

Semantic metadata also helps in organizing and finding data based on meaning not just text. For example, using semantics, systems can understand the context in which a word is used. When searching for references to “Mustang” in the context of cars, it can avoid references to horses. Semantic metadata can also be applied to processes, for example represented as web services. When web services are associated with semantic metadata, it is easier to discover them, and also new web services can be automatically composed from the existing ones.

Semantic Web uses mainly two technologies to add meaning to data namely Extensible Markup Language (XML) [41] and Resource Description Framework (RDF) [42]. XML allows users to add arbitrary structures to their documents by means of user defined tags, thereby annotating the web pages. However, although XML adds structure to the documents, it does not explain anything about what the structure means. Meaning to data is given by the Resource Description Framework (RDF) which is a W3C standard for describing resources on the web. RDF provides a way by which the meaning of a resource can be found out by mentioning how it relates to other resources. RDF identifies resources using Uniform Resource Identifiers (URI). It describes the resources in sets of triples, each having a subject (resource), predicate (resource property) and object (property value). Semantic technology is gaining more and more importance in Bioinformatics these days [43] [44]. Bioinformatics is the science of providing biologists with tools to exploit information from data in biology and other related fields. The advent of high-throughput experimental technologies and the growth of publicly accessible biological databases have greatly increased the amount of data available to the biologists. Also, the data is mostly heterogeneous in nature. Analyzing this data effectively has become a significant challenge, which is being increasingly addressed by the use of database management, scientific workflows and other information retrieval technologies. Such technologies have provided syntactic search,

heterogeneous data access and sharing, and some limited forms of integration. By adding semantic web technology, to the existing suite of technologies used, we can automate the analysis of data to a great extent which would lead to better understanding of biological processes and their interdependencies. Ontology is one area in Semantic Web which aids a lot in data analysis. For example, the Glyco [45] ontology developed in the LSDIS lab, models the structure and functions of glycans with unprecedented accuracy from where implicit knowledge about glycans can be deductively derived as well as experimental results can be validated according to the model. Thus, GlycO is intended to provide both a schema and a sufficiently large knowledge base, which will allow classification of concepts commonly encountered in the field of glycobiology in order to facilitate automated reasoning and information analysis in this domain. In the next section, we briefly describe Ontologies.

2.7 Ontologies

The third basic component of the Semantic Web is the collection of information called ontology. An ontology is an explicit specification of a conceptualization [4], where conceptualization is an abstract, simplified view of the real world that we wish to represent for some purpose. It is a formal model of a domain. Ontologies consist of concepts (classes), relations (properties), instances and axioms. Ontology can thus be thought of as a 4-tuple <C,R,I,A> where C is a set of concepts, R is a set of relations, I is a set of instances and A a set of axioms [46].

In the Semantic Web scenario, ontology can be viewed as a representational vocabulary used to describe a real world model which we wish to conceptualize. The typical kind of ontology for the Web has a taxonomy and a set of inference rules. The taxonomy defines classes

of objects, their properties and relations between different classes. Ontologies are also equipped with inference facilities which help computers to synthesize information on the Web in a more effective way.

The Web Ontology Language (OWL) is the language used for developing ontologies, and it is endorsed by the World Wide Web Consortium. OWL language provides mechanisms for creating all the components of ontology, including concepts, instances, properties and axioms. Properties can be of two types: Object properties relate instances to instances and datatype properties relate instances to datatype values like string or number. Concepts can have super and subconcepts, thereby providing a mechanism for inheritance of properties. Axioms are used to provide information about classes and properties, for example to specify the range of a property. In the Bioinformatics domain, examples of a few populated ontologies (ontologies with instances) are UniProt (http://www.pir.uniprot.org) and Glyco/Propreo [45] [47].

Figure 7 below shows an example of an ontology to be used later in explaining mapping. Figure 8 explains the relationship between different classes in the ontology. This particular ontology models the Loan process. In this ontology, Concepts represent the generic classes of entities in the domain of a loan e.g., Loan. The Concepts are associated with each other using different Relationships. E.g., PERSON *has_Loan* LOAN. So the Concepts PERSON and LOAN are associated with each other using the property *has_Loan*. This object type property *has_Loan* has PERSON as its domain class and LOAN as its range class.

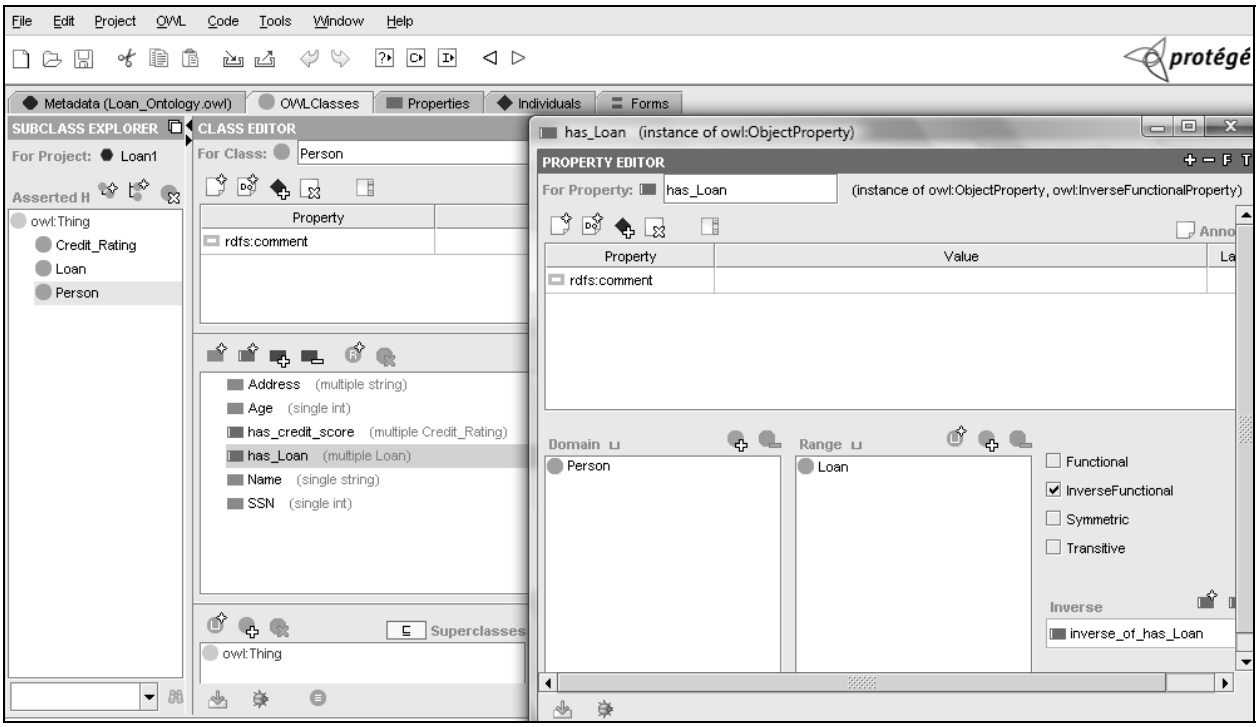


Figure 7: Loan Ontology

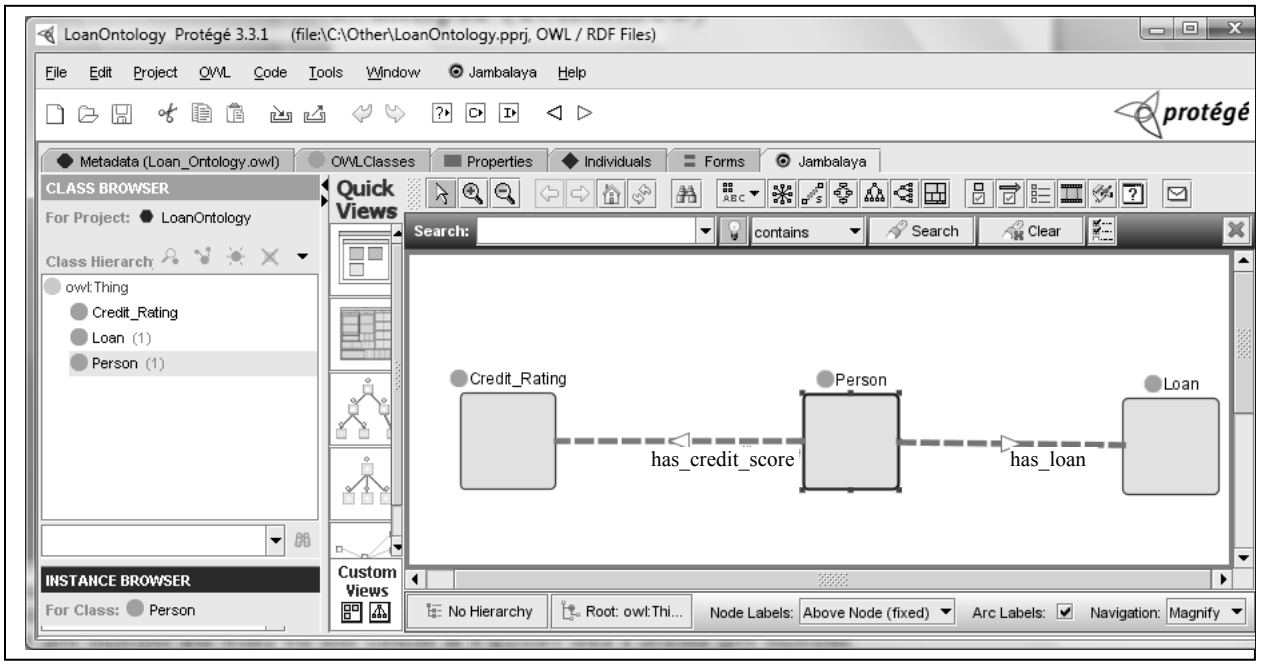


Figure 8: Loan Ontology with the relationship between different classes.

2.8 Semantic Annotation of Data

In order for machines to understand and automatically process Web resources, they need to be annotated with machine understandable metadata. Semantic annotation is the process of identifying items of interest in structured (database), semi-structured (XML files or spreadsheets) as well as unstructured (text files) data. The result which we obtain after semantically annotating a document is a set of assertions indicating named-entities within them. Such assertions can be embedded within the document or be placed in a separate document. Over the last few years, a number of systems have been built to perform the function of semantic annotation by annotating web pages with ontology derived semantic tags [48]. A few of these annotation tools for producing semantic tagging are CREAM [49], Annotea [50], SHOE [51], AeroDAML [52], SemTag [48] and Kim [53]. CREAM [49] allows creation of metadata in two modes, the annotation mode of CREAM allows to create metadata for existing web pages and the authoring mode lets authors create metadata. Annotea provides RDF-based annotation [50]. Simple HTML Ontology Extensions (SHOE) language [51] allows users to define ontologies and associate machine understandable meanings with them. AeroDAML [52] is another annotation tool which takes as an input an ontology and automatically produces a semantically marked up page which can then be checked by a human. SemTag [48] is an application written on Seeker, a platform for large-scale text analytics, to perform automated semantic tagging of large corpora. Similar to semantic annotation of data in the web, work has also been done in the semantic annotation of web services. The knowledge and Information Management (KIM) system is a product of Onto Text lab and is currently being further developed in SEKT [53]. It allows for metadata creation, storage and conceptual search. The ontology based information extraction method in KIM

produces annotations linked both to the ontological classes and to the specific instances in the ontology [40].

Semantic annotation can be thought about as assigning to entities and their relationships in the text, links to their semantic descriptions in the ontology as shown in Figure 9. Semantic annotation enables many new applications as in highlighting, semantic search and smooth traversal between unstructured test and formal knowledge. Semantic annotation is applicable to any kind of content- Web pages, non Web documents, fields in databases, images, audio, video and other media. These annotations can also be applied to Web Services. MSWAF (METEOR-S Web Service Annotation Framework) [54] is a framework for semi-automatically marking up web service descriptions with ontologies. In this thesis, we developed a method and a software tool to semantically annotate data produced from a biological workflow.

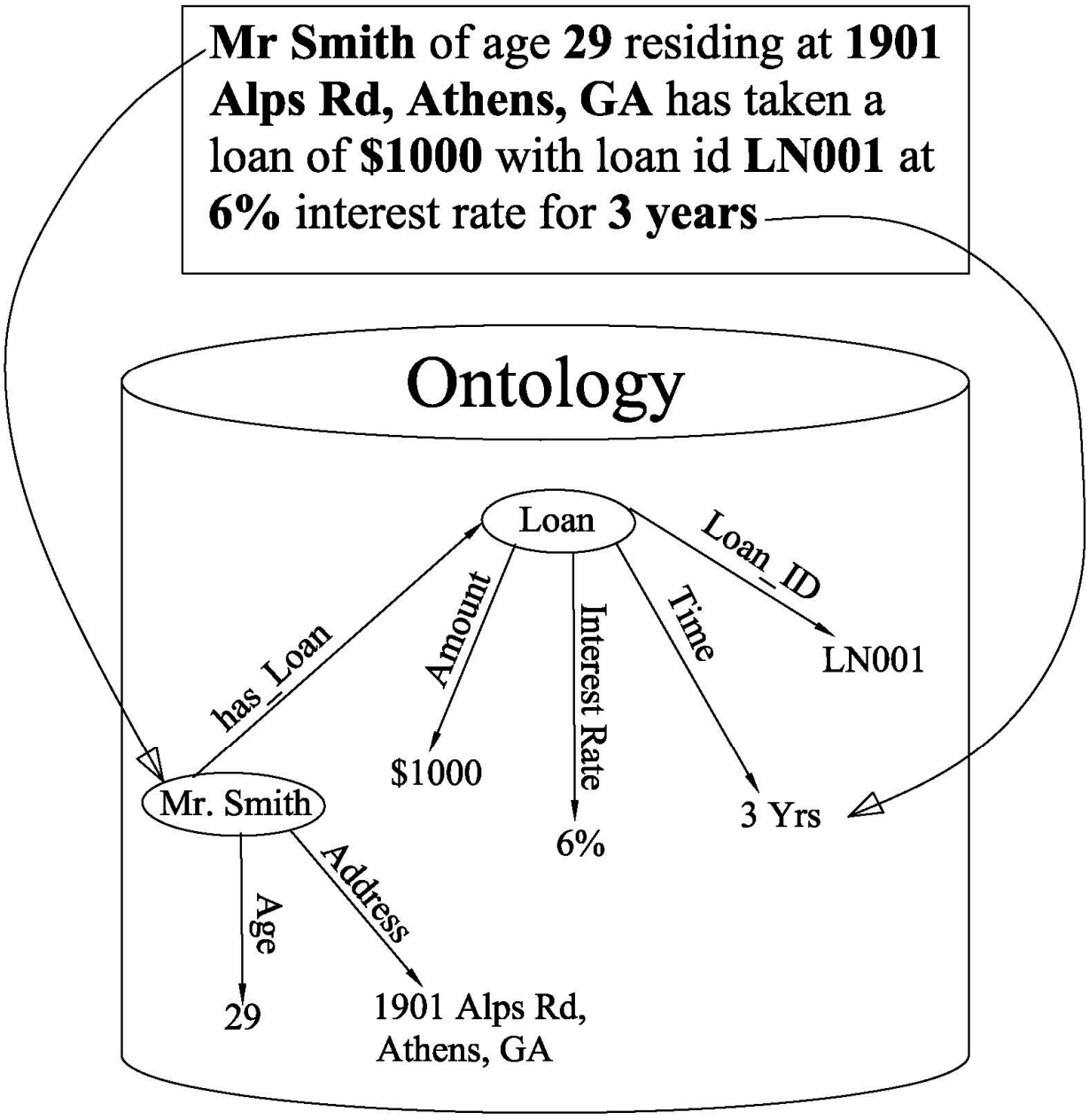


Figure 9: Semantic Annotation

CHAPTER 3

DEFINING WORKFLOW ANNOTATIONS

In this chapter, we present a detailed discussion on the process of workflow data annotation. We begin by discussing the workflow model and how data in workflow is handled. In this context we discuss the different workflow data patterns. Once we know how data in workflow is described, we need an ontology which will model our real world scientific process. Once we have the data from the workflow and the ontology in place, all we need to do is create appropriate mapping between workflow data and ontology resources, in order to automatically annotate the workflow data. The annotated data will enable us to understand and interpret the raw workflow data within our context. We begin our discussion with the workflow model, and the data patterns allowed. Next we discuss about ontology and finally the mapping.

3.1 Workflow Model

Workflow Management Systems typically support activity-oriented models. Activity-oriented workflow models support the following:

- Activity: The workflow activity is a piece of work that will be done by combination of resources and computer applications.
- Control flow: It refers to the order in which the individual tasks of a workflow are executed.
- Workflow resource: A workflow data object, an application or a role required for the execution of an activity.

- Workflow data: data which is shipped from one task to another, e.g. scanned or electronic document, spreadsheet, image file, fax or an email.
- Dataflow: path of the workflow data between activities
- Organizational structure and roles.
- Performers/participants: person or group that fills roles and interacts while performing activities in a particular workflow instance.

Of all these, we are particularly interested in the description of the workflow data, i.e. how data is shipped through the network of tasks in a workflow. To understand data description, we need to know about workflow data patterns which are presented in the following section. We study such patterns from the perspective of scientific workflows. We also discuss them in reference to the jBPM [9] Workflow Management System, which we are going to use for our case study.

3.2 Data Description in Workflow Model with Special Reference to jBPM

The Workflow Patterns Initiative [55], which started in 1999, has documented more than 100 patterns abstracting pieces of functionality supported by various workflow systems. The patterns cover the specification of control-flow dependencies [56], data definition and interaction [57], and various aspects of resource management [58] .

The data patterns describe the way in which data is represented and utilized in the context of a workflow system. The data patterns have been studied based on five main characteristics:

- Data visibility: focuses on the manner in which data elements can be viewed by various components of a workflow process.

- Data interaction: focuses on the various ways in which data elements can be passed between components within a process instance
- Data transfer: focuses on the way in which data elements are actually transferred between one process element and another
- Data routing: focuses on the various ways in which data elements can interact with other perspectives and influence the overall execution of the process.

We will describe the workflow data patterns in the context of scientific workflows.

Scientific workflows are becoming more and more important in various e-science fields as an important unifying mechanism to combine scientific data management, analysis, simulation, and visualization tasks [59]. In contrast to business workflows, scientific workflows often exhibit particular behaviors, e.g., they can be data-intensive, computation-intensive, analysis-intensive, and visualization-intensive, thus covering a wide range of applications from low-level "plumbing workflows" of interest to Grid engineers, to high-level "knowledge discovery workflows" for scientists [60]. Consequently, scientific workflow modeling is mainly dominated by dataflow-oriented modeling approaches, while business workflow modeling is dominated by control, event, and task-oriented approaches [61]. We will be using jBPM [9] for developing our WFMS. Let us now discuss data description in jBPM with reference to the above mentioned workflow patterns.

3.2.1 Data Visibility

Scientific Workflow Systems should have data-tracking facilities throughout the life cycle of the workflow. Users should be able to inspect data as they are being manipulated at each step of the workflow.

In jBPM, by default variables have a scope over the entire workflow process. They are created as part of the definition of an individual task via the controller construct. In jBPM, data elements, are mostly characterized by process variables, so the Case Data pattern (Workflow Data Pattern-5) [62] is directly supported, since data can be accessed by all components of the workflow during the execution of the workflow instance or case.

In jBPM, variables have a scope over the entire workflow process. In addition to process variables, there are also task instance variables. A task instance can have its own set of variables; in addition it has access to all the process variables. The task controller makes the translation from the process variables to the task variables. At creation of a task instance, the task controllers can populate the task instance variables and when the task instance is finished, the task controller can submit the data of the task instance into the process variables, as shown in Figure 10 [63]. In a simple scenario, there is a one-on-one mapping between process variables and the form parameters unless we want to write our own custom task controller. Task controllers are specified in a task element. In this case, the default jBPM task controller can be used and it takes a list of variable elements inside. The variable elements express how the process variables are copied in the task variables.

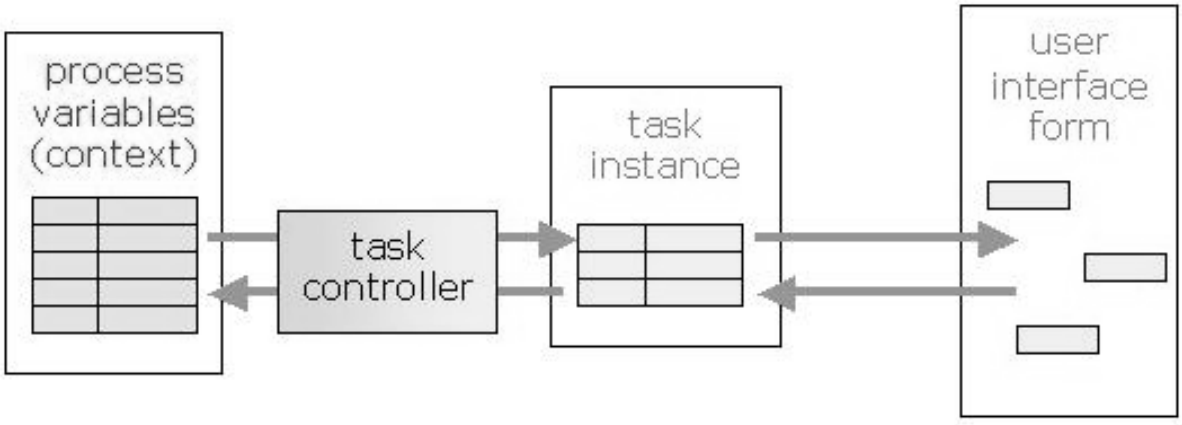


Figure 10: The Task Controller

In Figure 11 we see how we can create separate task instance variable copies based on the process variables:

```
<task name="addition">
<controller>
<variable name="number1" access="read" mapped-name="x" />
<variable name="number2" access="read,write,required" mapped-name="y" />
<variable name="c" access="read" />
</controller>
</task>
```

Figure 11: Task Instance Variables and Process Variables

Here, *number1* and *number2* are the names of two process variables. The mapped name is optional and refers to the name of the task instance variable. The access attribute specifies if the variable is copied at task instance creation, will be written back to the process variables at task end and whether it is required. This information can be used by the user interface to generate the proper form controls. The access attribute is optional and the default access is 'read,write'. If we want the task variable to be written back to the process variable we specify the access attribute as write, else we keep it as read.

As already discussed, because the task variables are mapped to process variables on a one to one basis and the inherent difficulty of defining task variables that do not strictly correspond to existing process variables, the Task Data pattern (Workflow Data Pattern -1) [62] is not considered to be fully supported.

3.2.2 Data Interaction Pattern (Internal)

In scientific workflows often one task requires as input the output of a previous task, so it is desirable that the data is made available from one task to the other.

In jBPM, variables have process scope, hence the Task to Task pattern (Workflow Data Pattern - 8) [62], which is the ability to communicate data elements between one task instance and another within the same case, is supported through the global shared data approach.

3.2.3 Data Interaction Pattern (External)

External data passing involves the communication of data between a component of a workflow process and some form of information resource or service that is operated outside of the context of the workflow engine, such as a web service. The interaction can either be of the form of push based interaction from the outside environment to the task, or a pull based interaction from the task to the environment.

Typically, for a scientific workflow, tasks need to pass data to the environment (e.g. a task might need to pass its result to a database) as well as request data element from services or resources in the environment (e.g., a task might need to get some values from a web service before it can proceed). As such scientific workflows need to support both Push and Pull data patterns which are explained below in context of jBPM.

External data interaction in jBPM is realized at task level, rather than at the process level. The Data Interaction –Environment to Task – Push pattern (Workflow Data Pattern – 16) [62] are implemented through action handlers, which are defined on transitions or the entrance or the exit of nodes. For example, the connection of a jBPM task to an external database and the writing of data to that database can be done by coding an action-handler java class which enables

the required data transfer. On the other hand external data interaction can also be implemented through -Task to Environment – Pull-oriented Workflow Data Pattern -17) [62]. For example Nodes can stop and wait for the receipt of data which they are retrieving from external sources.

3.2.4 Data Transfer Patterns

Ideally, a scientific workflow should be able to copy the values of a set of data elements into its address space at the beginning of execution of a task and be able to copy back any changes to this data once the task completes.

Data is transferred in jBPM using a Copy in/Copy out strategy (i.e. Workflow Data Pattern -28) [62]. When dealing with concurrent tasks, every work item when it is created receives a copy of the data values in the global process variables. Any updates to these values by a concurrent work item remains invisible to that work item. At task completion, the values for each work item are copied back to the global process variables, overwriting the existing values for these variables.

3.2.5 Data-based Routing:

Scientific workflows should be able to provide the ability for the completion of one task instance to trigger one or several subsequent task instances depending on the outcome of an expression based on the values of various workflow data elements. jBPM supports the Data-based routing pattern (Workflow Data Pattern -39) [62] through the Decision Node construct.

Thus, some of the desirable workflow patterns of a scientific workflow are:

- Case Data pattern by the virtue of which data can be accessed by all components of the workflow during the execution of the workflow instance or case.

- Task to Task pattern which gives the workflow the ability to communicate data elements between one task instance and another.
- Ability to push data from the outside environment to the task as well as pull data from the task to the environment.
- Copy in/Copy out strategy which allows the workflow to copy the values of a set of data elements into its address space at the beginning of execution of a task and be able to copy back any changes to this data once the task completes.
- Data-based routing pattern which allows the workflow to be able to alter the control flow as a result of the value of data based expressions.

3.3 Ontologies

This section analyzes the role of ontology in defining workflow annotations. Ontologies help conceptualize real world models which we wish to work on. They define the formal framework for mapping in case of semantic annotations. In the context of ontology, Information Extraction (IE) is briefly discussed below. It is a technology based on analyzing natural language in order to extract information of interest. Ontologies can be used for IE. One main difference between traditional and ontology based IE, is the use of formal domain language as one of the system's resource in case of the latter. Ontology based IE may thus involve reasoning. Another difference is that, ontology based IE not only retrieves the most specific type of the extracted entity, but also identifies it, by linking it to its semantic description in the description of the instance base [40]. This allows entities to be traced across the documents and hence their semantics enriched. In general, ontology provides the vocabulary for referring to terms which are in the subject of our interest. The vocabulary they provide is quite different from human-oriented

vocabularies such as thesauri, which rely on natural languages and are subject to different human interpretations. Ontologies provide logical statements that describe what the terms are, and how they are related to other terms in the domain of our interest. Ontologies also provide inference rules for combining terms and their relations to define extensions to the vocabulary. Ontology specifies the terms with unambiguous meaning, with semantics independent of the reader and the context, so that information becomes machine processable.

Taxonomy refers to a hierarchical classification of entities within the domain of interest. Ontology provides taxonomy in a machine processable form.

Ontologies also enable consistency checking for applications, such as type and value checking with ontologies that include class properties and restrictions.

The major purpose of ontologies is, however not to serve as vocabularies or taxonomies, but to share and reuse knowledge among applications. Each ontology provides a description of concepts and relationships that exists in its domain, and that can be shared and reused among different intelligent agents and applications as a formal specification.

And finally, coming to the context of Semantic Web, ontologies form its basic building block. Semantic level interoperation is possible only if semantics of the Web data is explicitly represented on the Web in the form of machine understandable and processable data, which can be done by the means of ontology. Through automatic use and machine interpretation of ontologies, computers can share the burden of information extraction and interpretation, previously done by humans alone.

Ontologies provide access to a huge network of machine understandable and processable human knowledge, which would enable interoperability as well. Once the essential knowledge of

a certain domain is encapsulated in the web in the form of interconnecting ontologies, it creates a base for further development of knowledge applications in the domain.

More specifically, ontologies play various roles in the Semantic Web:

- They allow Web knowledge processing, sharing and reuse between applications, by sharing of common concepts and specialization of concepts and vocabulary for reuse across multiple applications.
- They allow further levels of semantic interoperability on the Web in the form of mappings between terms within the data.
- They enable intelligent services to operate like information broker, search agents and knowledge management.

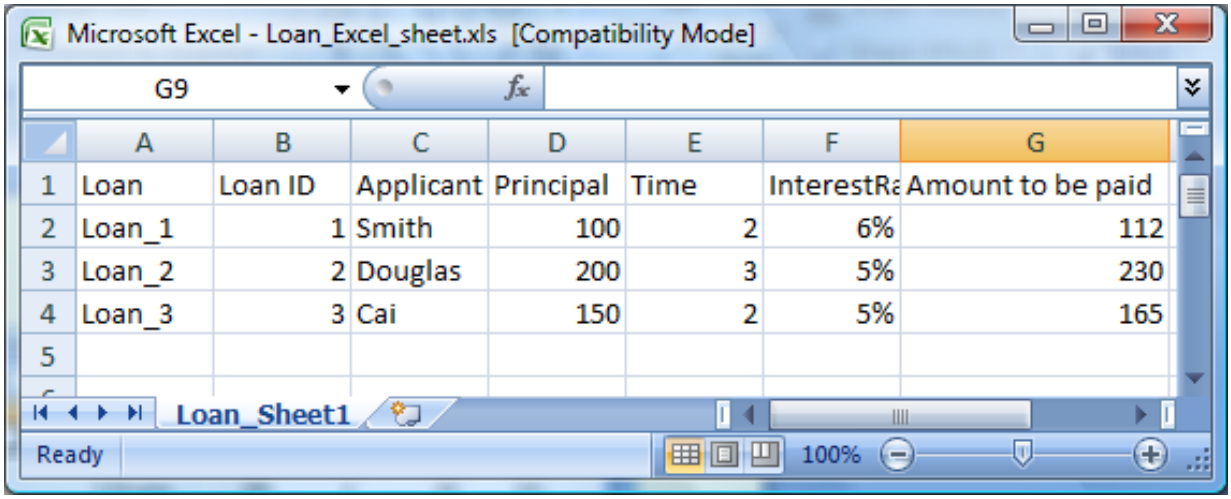
3.4 Mapping

Once we have the ontology and the workflow data, we can create a mapping between the two, so that the data produced from the workflow will be annotated with the corresponding concepts from the ontology. This would enable users to make better sense of the raw data being produced from the workflow in their own context, as well as understand correlations existing within data.

For the purpose of explanation we use the example of a simple loan process. First, we create an Ontology describing the concepts and relationships that exist in our loan process. Figure 7 and Figure 8 shows the hierarchy in the ontology for the loan process. There are three classes namely Person class describing the person who takes the loan, Loan class describing the loan entity and the Credit Score class describing the details about the credit score of an object of

the Person class. Each of these classes in turn has properties describing the attributes of these classes.

For our annotation tool we only consider workflow data in the form of spreadsheet. Figure 12 shows a sample output data from the loan workflow which captures information regarding the loan, and calculates the final loan amount.



	A	B	C	D	E	F	G
1	Loan	Loan ID	Applicant	Principal	Time	InterestRa	Amount to be paid
2	Loan_1	1	Smith	100	2	6%	112
3	Loan_2	2	Douglas	200	3	5%	230
4	Loan_3	3	Cai	150	2	5%	165
5							

Figure 12: Sample output data from Loan workflow

Based on the domain Ontology (Figure 7) and the workflow data (Figure 12), we use our Web based annotator tool: Tool for Ontology based Annotation of workflow Data (TOAD), to create a Annotation Definition File (ADF) to hold the mapping definitions between workflow data and corresponding ontology concepts as shown in Figure 13.

```
@prefix dc:  <http://www.owl-ontologies.com/loan.owl#>

<dc:$Loan>  <rdf:type>  <dc:Loan>

<dc:$Loan>  <dc:Loan_ID>  <$Loan ID>

<dc:$Loan>  <dc:from_person>  <$Applicant>

<dc:$Loan>  <dc:Principal>  <$Principal>

<dc:$Loan>  <dc:InterestRate>  <$InterestRate>

<dc:$Loan>  <dc:Time>  <$Time>

<dc:$Loan>  <dc:Amount>  <$Amount to be paid>
```

Figure 13: ADF for the Loan Process

The entries in the ADF are in Notation3 format [64]. Notation 3, or N3, is a shorthand for writing RDF [42] files. In N3, we can write an RDF triple in the <subject> <predicate> <object> form. As the loan workflow runs, there will be specialized tasks called annotation tasks which get called from the workflow. The annotation task gets the ADF which has the mapping entries, and the data produced from the workflow. It then annotates the workflow data according to the mapping entries in the ADF. Figure 14 shows the annotated data produced from the loan workflow.

```
@prefix dc:  <http://www.owl-ontologies.com/loan.owl#>

<dc:Loan_1>  <rdf:type>  <dc:Loan>

<dc:Loan_1>  <dc:Loan_ID>  <1>

<dc:Loan_1>  <dc:from_person>  <Smith>

<dc:Loan_1>  <dc:Principal>  <100>

<dc:Loan_1>  <dc:InterestRate>  <6%>

<dc:Loan_1>  <dc:Time>  <2>

<dc:Loan_1>  <dc:Amount>  <112>
```

Figure 14: Annotated Workflow Data

The annotated output from the workflow would make better sense as well as help to understand the correlations that exist within the data. For example, from Figure 14, we can infer that the spreadsheet entry **Loan_1** is a type of Loan entity which has Loan_ID as **1**. The owner of this loan is **Smith**. He has applied for a loan of **$100**, which he will be paying for **2** years at **6%** interest rate pa. The final amount he has to return is **$112**. This is how we create a mapping between workflow data and an ontology to make better sense of the data. Figure 15 shows the relation between the annotated data and the ontology concepts.

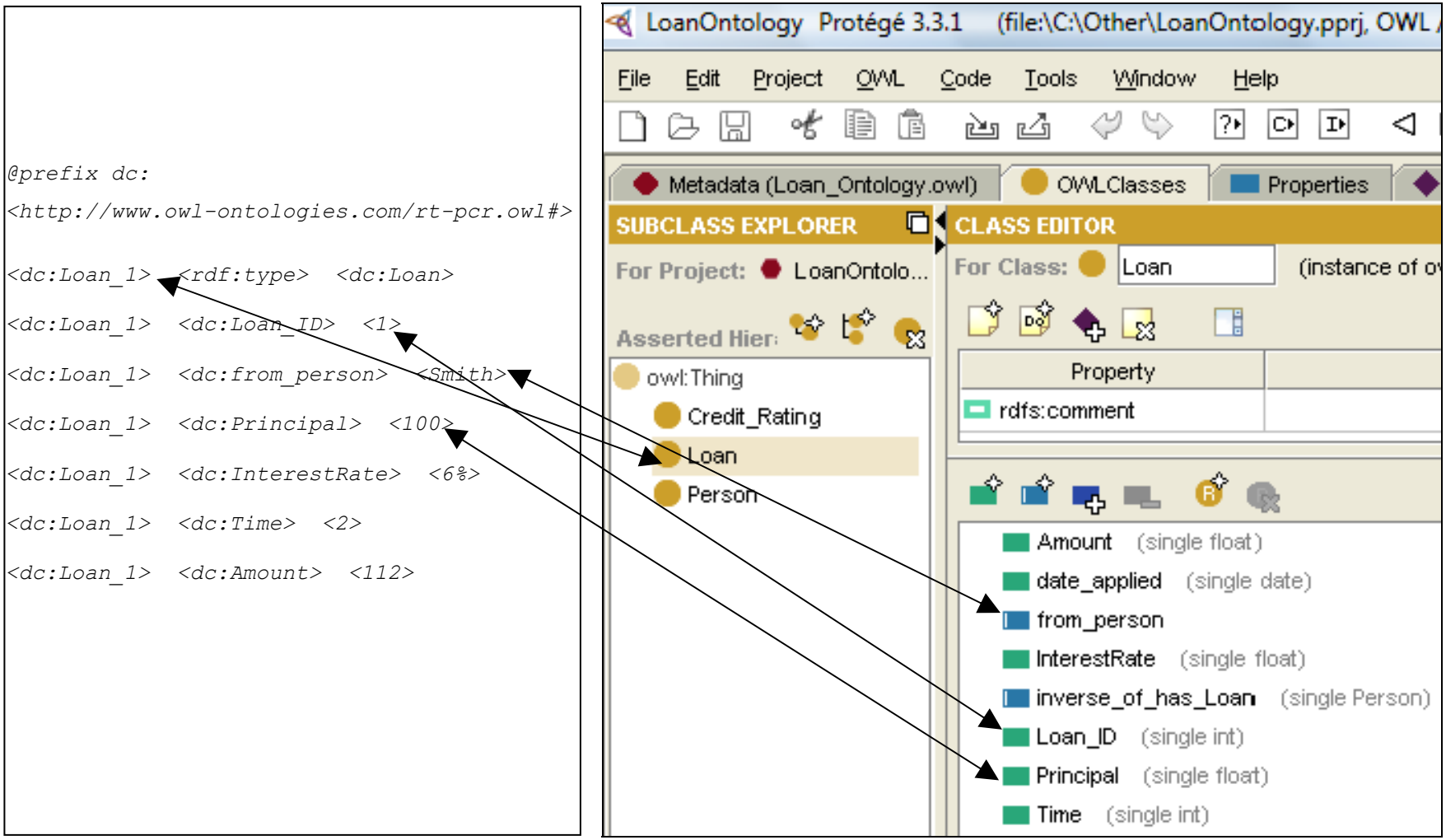


Figure 15: Annotated Loan Data

CHAPTER 4

WORKFLOW ANNOTATION DEFINITION TOOL

The following chapter introduces the workflow annotation definition tool we have developed for annotating scientific workflow data. In addition to describing the annotation definition tool, we also describe the implementation details of the annotation task which gets called from the workflow to annotate the workflow data according to the mapping definitions created by the annotation definition tool. We also describe how the annotation task is incorporated in a jBPM workflow. We use the Loan ontology to annotate the sample Loan Workflow data, discussed in Chapter 3 as an illustration.

4.1 TOAD: Tool for Ontology based Annotation of Workflow Data

TOAD is an automated generalized web-based tool for defining annotations for scientific workflow data. It needs two inputs:

- The workflow data which needs to be annotated. For the purpose of this thesis we have just considered data in the form of spreadsheet file. But the annotation definition tool can be extended to include any type of output data, such as text files, XML files and SQL files. In any of the cases it would be possible to parse the data file and annotate the data of interest with concepts from ontologies which model the workflow producing the data.
- The ontology which models the process for which the workflow is created.

For the purpose of explaining the tool we refer back to the Loan workflow data and Loan ontology which we mentioned in Chapter 3. Figure 16 and 17 shows the UI interface for the TOAD tool.

The TOAD tool first asks the user to upload the spreadsheet containing some sample workflow data and the Ontology file with which the data needs to be annotated. Once uploaded, TOAD shows the UI screen as shown in Figure 16. The Left hand side shows the class and property hierarchy of the ontology uploaded. The right hand side shows the spreadsheet data rendered in HTML [65]. The tool asks the user to enter a prefix name for identifying the ontology namespace. A namespace helps to uniquely identify an ontology and is usually identified by an URI (Uniform Resource Identifier) over the Internet. Since namespaces are usually long, so prefixes are used as abbreviations for namespaces. Next the user selects the spreadsheet data he wants to annotate and the ontology class (or property) with which he wants to annotate it. Whenever the user selects an ontology class or property, or any cells in the spreadsheet file, TOAD has the ability to programmatically know what value the user has clicked on and accordingly show the user the subject, property and object he has chosen. Figure 17 shows that the user has clicked on the *Person* class in the Loan ontology and the *Applicant* column in the spreadsheet. When the user hits the Create Mapping button, the mapped triple is shown below. The user can create multiple mappings. Once he is done, he supplies a name for the Annotation Definition File (ADF) which will be produced and hits the Create Template File button. TOAD writes the triples created into the ADF. The ADF becomes the mapping template for the workflow annotation tasks which annotates the workflow data with the ontology concepts, based on the mapping entries in the ADF.

Using TOAD, a user will have the ability to annotate a column, a row, a range of cells and an individual cell in the spreadsheet with corresponding concepts of an ontology.

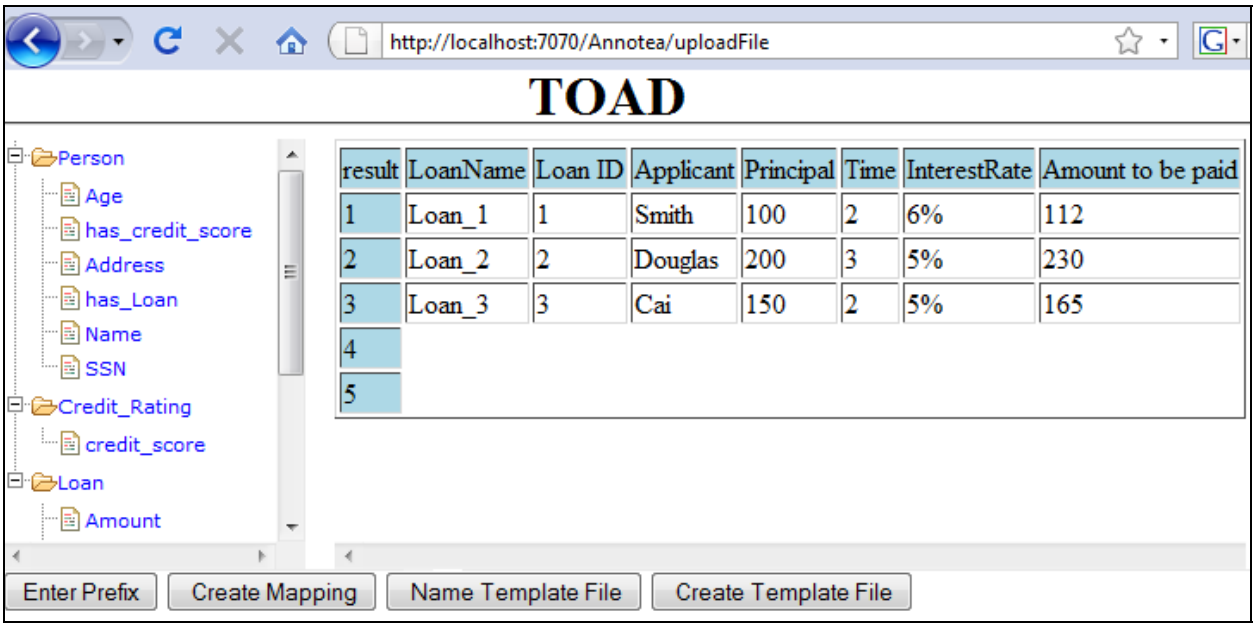


Figure 16: UI Interface for TOAD – I

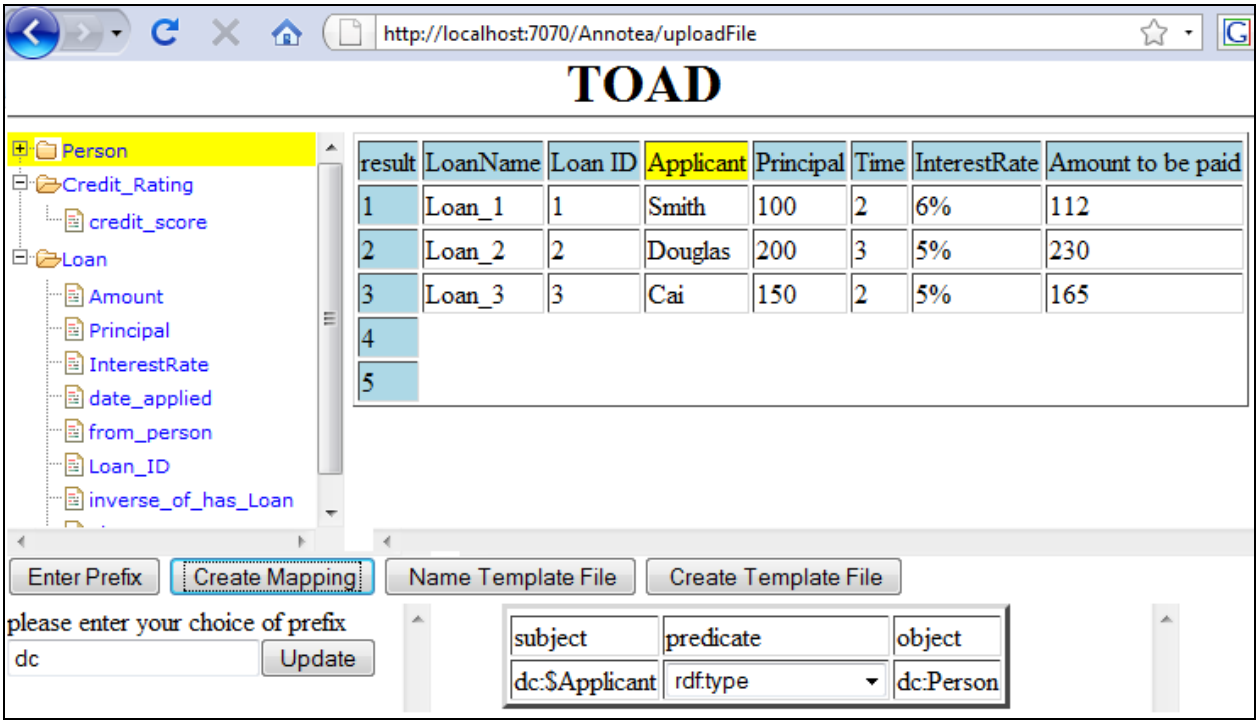


Figure 17: UI Interface for TOAD – II

4.2 TOAD Implementation.

In TOAD we desire a system which will enable users to map workflow data with related concepts from an ontology with ease. In order to satisfy this requirement, we develop TOAD as a web-based tool, which is implemented using Java and Ajax (Asynchronous JavaScript and XML) [66]. Ajax refers to a group of interrelated web development techniques used for creating interactive web applications. Using Ajax, it is possible for web applications to retrieve data from the server asynchronously in the background without interfering with the display and behavior of the existing page. In the development of TOAD we have used the JavaScript library provided by jQuery [67] since it simplifies HTML document traversing, event handling and Ajax interactions, making web development relatively faster and easier.

In TOAD, using Ajax we render the ontology file uploaded by the user into a hierarchical representation of ontology classes, subclasses and properties. TOAD also has the ability to know programmatically the ontology class or property name which the user clicks on for mapping. The spreadsheet file uploaded by the user is rendered as a HTML table. The user can click on a column name, row name, an individual cell or a range of cells which he wants to annotate. TOAD internally creates the mapping between the workflow data and ontology class/property and stores the mapping as N3 [64] triples in the Annotation Definition File (ADF).
When the workflow runs, special tasks called annotation tasks which we discuss in the next section, get called from the workflow and it annotates the data which the workflow produces using information from the mapping entries in the ADF.

4.3 Annotation Task Implementation.

Workflow annotation task is a special type of workflow task which helps to annotate the workflow data with the Annotation Definition File (ADF) created from TOAD. It is a run time specification and is attached to a workflow node which requires annotation. In jBPM, it is implemented as an extra action handler class to the workflow node which produces the data. This action handler class is generally triggered on the node-leave event of the corresponding node, so that the data which is produced from that node gets annotated by the annotation task as control-flow leaves the node.

For one workflow, it should be possible to have multiple ADF depending on the data to be annotated. From TOAD, we can define the ADF names and in the workflow annotation task, we set the name of the ADF according to which we would like to annotate the data in the action handler class for that task. Also in the action handler class, we set the name of the data file from which the data needs to be annotated. Once the action handler has the name of the ADF and the workflow data file, it is able to annotate the data file using mapping definitions from the ADF. The annotated data is written in N3 notation as shown in Figure 15 which is another way of representing RDF files.

4.4 Handling of different data representation by TOAD.

Although at present, TOAD has been designed to work only with spreadsheet data, it should be possible to extend it to work with other representation of data such as structured data (e.g., relational database), semi-structured data (e.g., XML file) and also unstructured data (e.g., text file). In case of text files, we can parse the text file using some regular expression, or using a combination of line number and column number. Once the user specifies the line number and

column number or some regular expressions in the text file, TOAD will create mapping entries in the ADF indicating the coordinates of the data which needs to be mapped. The workflow annotation task will read the ADF, parse the text file to retrieve the data indicated by the coordinates in the ADF and map it with ontology concepts as mentioned in the ADF. In case of structured data, such as data stored in a relational database, the user can select database column names which he wants to map to some ontology concepts and TOAD can store that information as mapping entries in the ADF. Later the annotation task can run SQL queries to retrieve the data corresponding to the column names in the ADF and map those to the ontology concepts present in the mapping entries. In case of semi-structured data represented as an XML file, the user can select data based on tag names. TOAD can query the data based on tag names using XQuery and then annotate the query result with corresponding ontology concepts as entered in ADF.

CHAPTER 5

CASE STUDY

In this chapter, we present a case study for the annotation tool, TOAD which we have developed. Using TOAD, we annotate the experimental data obtained from running the RT-PCR workflow. We discuss briefly the qRT-PCR process, the RT-PCR ontology which we have developed to model the process, the RT-PCR workflow which semi-annotates the qRT-PCR process and, finally, the semantic annotation of the data produced from the workflow using TOAD.

5.1 qRT-PCR Process

Glycan structures are complex carbohydrates vital to the functioning and development of multicellular organisms [68]. In the field of glycobiology, one of the major research topic focuses on understanding how glycan structures are regulated in abundance and the effect these changes have on an organism [69]. Most models for glycan regulation have hypothesized the fact that changes in glycan abundance have a strong correlation with the by the alteration in transcript expressions of corresponding biosynthetic enzymes. The relatively low abundance of transcripts for many glycan related genes poses serious challenges to the study of transcript profiling of glycan-related genes. As a part of the Glycomics [70] studies at the Complex Carbohydrate Research Center [71] at the University of Georgia, scientists are developing methods for the detection and quantification of transcripts encoding “glycan-related” genes by

quantitative real-time PCR (qRT-PCR). qRT-PCR shows changes in glycan-related transcripts for both highly expressed genes as well as low abundance transcripts. Figure 18 shows the qRT-PCR methodology as being used in CCRC. Over 700 glycan-related genes of mouse were collected from various sources. Next, primer pairs are designed for glycan related genes as well as housekeeping genes using a restricted set of conditions. After the primer pair and gene are selected, a mixture is created along with SYBR Green and fed into a 96 well plate in the PCR machine. The machine runs 40 amplification cycles for around 2.5 hours. Primer pairs are tested in triplicates and the average of the cycle threshold (Ct) values is compared to that of a housekeeping gene. The output of the PCR machine is a binary file which is given as an input to the MyIQ or iCycler machines developed by Bio-Rad [72]. After the data acquisition phase, there is a data analysis phase where in the data obtained from the PCR reaction is passed through various quality control steps. The Ct values as well as the raw fluorescence data are copied from the Bio-Rad machine and pasted into a spreadsheet file. Triplicate values with a standard deviation of greater than 0.5 Ct are re-assessed. The raw fluorescence data is also analyzed using the LinReg [73] software to determine the amplification efficiency of the individual reactions. If the standard deviation of the efficiency was found to be greater than 5%, the experiment has to be re-run.

5.2 RT-PCR Ontology

The RT-PCR Ontology developed by Arpan Sharma of the LSDIS lab at UGA, models bio-science related scientific experiments involving RT-PCR. It models the various protocols associated with such experiments and helps to find out correlation among various elements

involved in the experiment. Figure 19 and Figure 20 shows the hierarchy in the RT-PCR ontology using the Protégé editor [74].

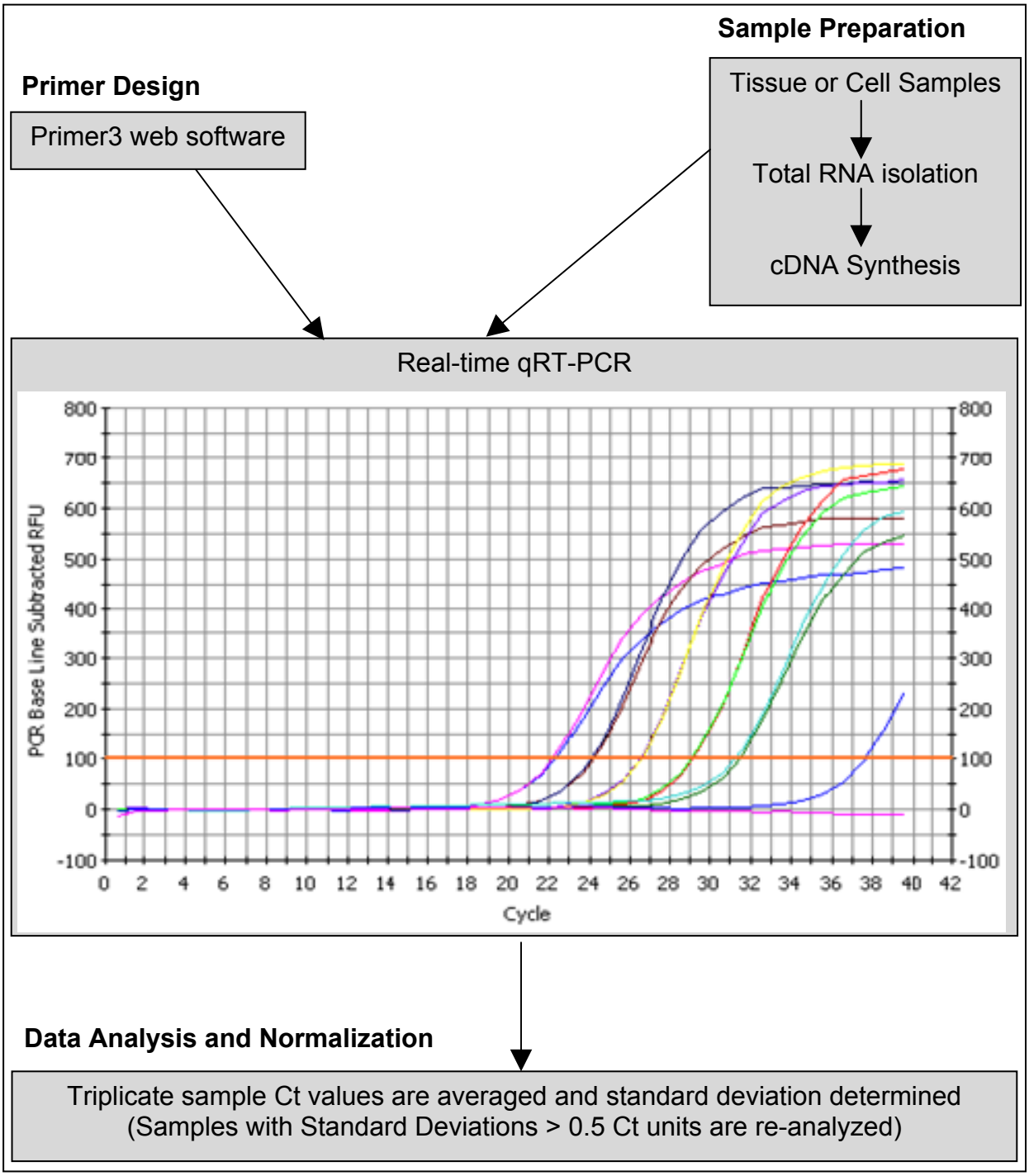


Figure 18: RT-PCR Methodology

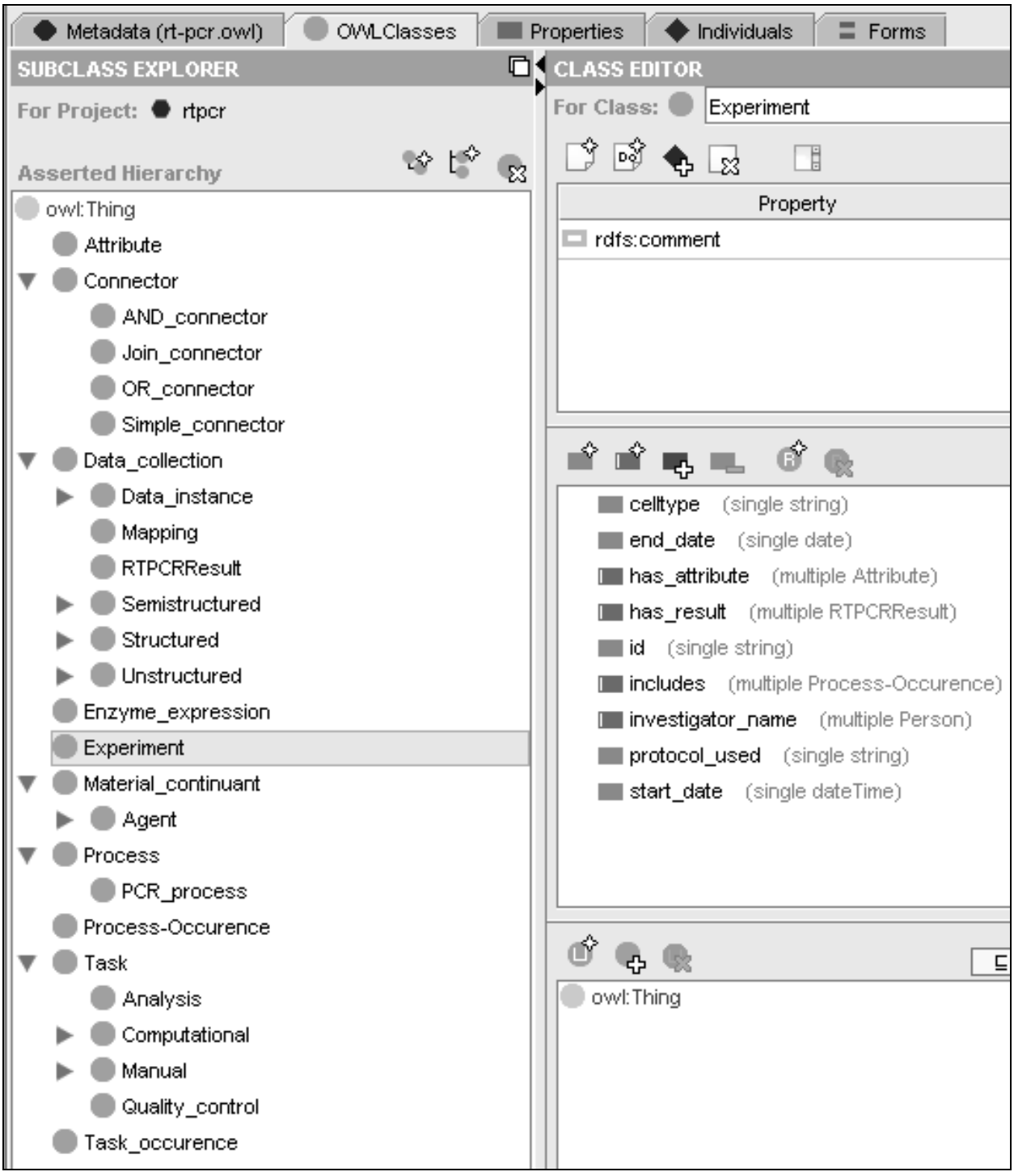


Figure 19: RT-PCR Ontology (Courtesy of ontology creator Arpan Sharma)

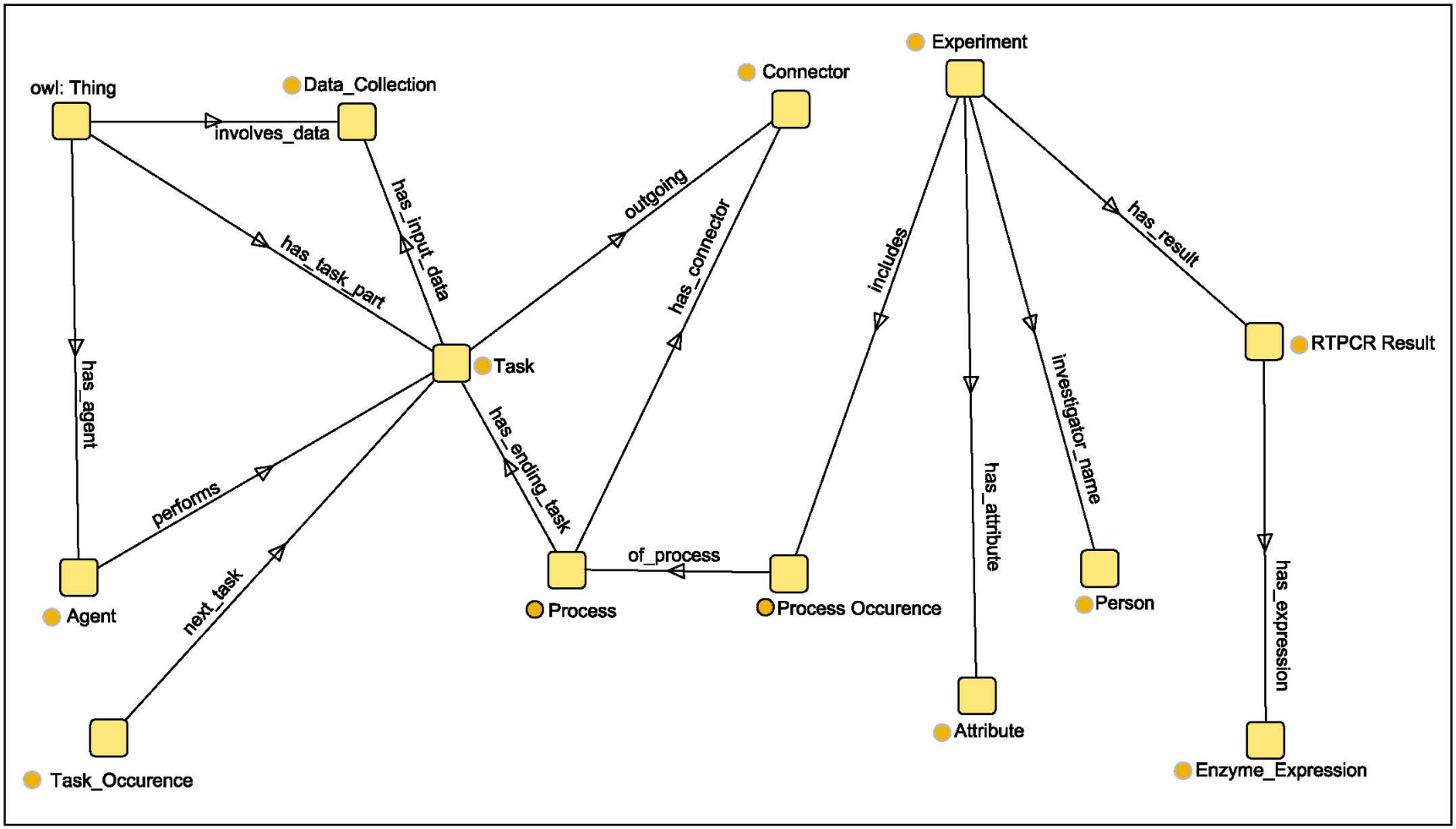


Figure 20: RT-PCR Ontology showing the relation between the classes.

It has a root classes called *Experiment*, which models a real life experiment. *Experiment* has data type properties which described the various attributes of an experiment including the *experiment id*, the *start date* and end *date* (biological experiments sometimes spans over days), the *cell type* for which the experiment was run, the *protocol* used (for e.g. RT-PCR). It also has some object type properties, such as *investigator name*, which relates to an object of the type *Person*. *Person* in turn is a subclass of the class *Agent* which describes various agents needed by an Experiment – it can be a *chemical agent*, *hardware*, *Manufacturer* or a *Person* who runs the experiment. The *Experiment* class, also has an object type property called *has_result* which relates to the *RTPCRResult* class which is a subclass of the *Data_collection* class. *Data_Collection* class describes the various output data formats that can be expected from an *Experiment* as for instance the data can either be in *structured* format (e.g. spreadsheets), s*emistructured* (e.g., an XML file) or *unstructured* (e.g., a text file). *RTPCRResult* has an object type property called *has_expression* which relates to the *Enzyme_expression* class. An object of *RTPCRResult* object can have multiple *Enzyme_expression* objects each for a triplet in the RT-PCR reaction. The *Enzyme_expression class* has 4 datatype properties *Average, enzyme_name, has_plate* and *Standard_deviation.* These properties describe the attributes which each well triplet should have in the RTPCRResult. Namely, they include the well plate number, the name of the enzyme which was fed into that well, the average and standard deviation of Ct values for that enzyme triplet.

The role of the RT-PCR Ontology is to semantically correlate between associated objects in a bio-science experiment such as RT-PCR. It would enable semantic metadata extraction from all forms of experimental data, be it in the structured, semi-structured or unstructured form. It

models various aspects of the experiment starting from the protocol used, the person conducting the experiment, the various inputs involved to the final output. In creating this formal representation of the entire scientific experiment, we allow for representation of metadata about the experiment [75]. As for example, for an experiment id, we will be able to find out which person conducted the experiment, which gene was put in which plate and ultimately what results did it yield. Accordingly, scientists can make their findings about the experiment.

5.3 RT-PCR Workflow

In scientific research, it is common that experiments are conducted over a long time and across many different locations involving many different people. By automating various tasks involved in such processes using WFMS, we not only make the processes easier, but also help in systematic storage and retrieval of data.

The PCR research currently ongoing at the CCRC lab at UGA involves a large number of experiments involving large volume of data and interaction of humans at various stages of the experiments. Also, the data exchanged between any two steps of an experiment may be of heterogeneous nature, requiring modifications to make them homogeneous. In the current scenario, all these steps require human intervention at various stages of the experiment. Also, in the absence of systematic storage of the experiment results obtained, there are chances of data getting lost, or data not being able to be reused when needed.

Our goal has been to develop a WFMS for providing semi-automate support for the RT-PCR process and systematic storage of the experimental data produced. We developed the WFMS using jBPM. Figure 5 shows the process definition of the RT-PCR workflow as designed by the jBPM graphic process designer. The workflow at first allows the user to select a template

for a well assignment from an available list of templates. Next, the genes are fed into a 32X3 well plate in the PCR machine according to the templates selected. The reaction runs for approximately two and a half hours and generates a binary file with the results. Once the output file is generated, the workflow runs the Bio-Rad software which outputs the cycle threshold data (Ct) as well as the fluorescence data values extracted from the binary file. The data is then exported to spreadsheet and the workflow performs some statistical calculations on the data. Next, the workflow invokes the LinReg web service which performs the logic which LinReg software uses to calculate the PCR Efficiency value of the fluorescence data.

In order to automate the qRT-PCR process, we developed the LinReg web service, so that instead of having to call a third party software, we can directly invoke the service from the RT-PCR workflow we have developed. As discusses before, in the PCR machine we input genes into a 96 (32 X 3) well plate. Using the LinReg web service we find the PCR efficiency value corresponding to the the fluorescence data for each well of the 96 well plate. It has an operation called *PCREfficiencyOperation* which does the function of calculating the PCR Efficiency value. For each well plate, we use an algorithm to find consecutive fluorescence values which would have the highest correlation coefficient as well as have a high slope. If we fit the fluorescence values in a curve, such values would actually correspond to the coordinates in the linear section of the curve. Once we get the fluorescence values of interest, we find the slope for each of the 96 well plates, and then find their corresponding PCR efficiency value by the simple formula: PCR efficiency = 10^{slope} . The *PCREfficiencyOperation* has an input message called *PCREfficiencyOperationRequest* which is a two dimensional array [40 X 96] of doubles containing the fluorescence data. Its output message is called *PCREfficiencyOperationResponse*

which is a single dimensional array of doubles of size 96, containing PCR efficiency value corresponding to the 96 well plates.

After the LinReg web service gets invoked, some quality control checks are run on the results of the statistical calculations as well as data obtained from LinReg web service. If the data obtained for some samples fails the quality check, the experiment needs to be repeated for the samples. Finally, the results are shown to the user for analysis and then stored in the database. Figure 21 shows a pictorial representation of the RT-PCR workflow.

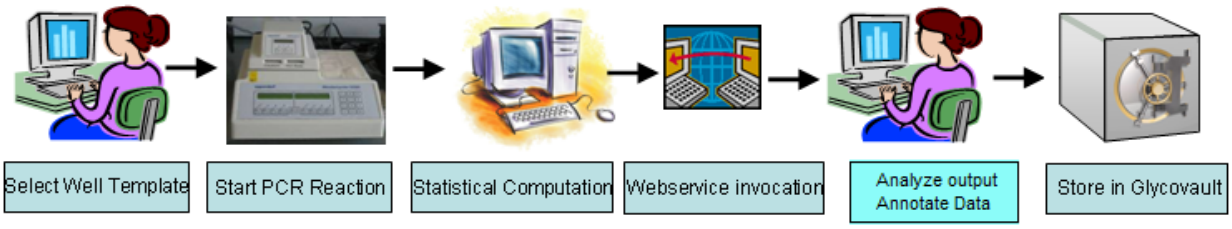


Figure 21: RT-PCR Workflow

5.4 Annotation of RT-PCR Workflow Data

RT-PCR workflow being a data and computation intensive workflow produces large volumes of data. The data is mostly output in the spreadsheet files. In order to make sense out of this workflow data as well as to understand the various correlations that exist within the data, we semantically annotate this data using the RT-PCR ontology which is a formal representation of the RT-PCR reaction. Figure 22 shows a sample workflow data and Figure 19 shows the RT-PCR ontology. We used TOAD, the annotation tool we have developed, to select the workflow data we want to annotate by a corresponding otology concept. TOAD created the ADF which

contains the mapping template. Figure 23 shows an example ADF for the RT-PCR workflow data produced by TOAD. As the workflow is executed, special annotation tasks are called to annotate the workflow data according to mapping entries in the ADF.

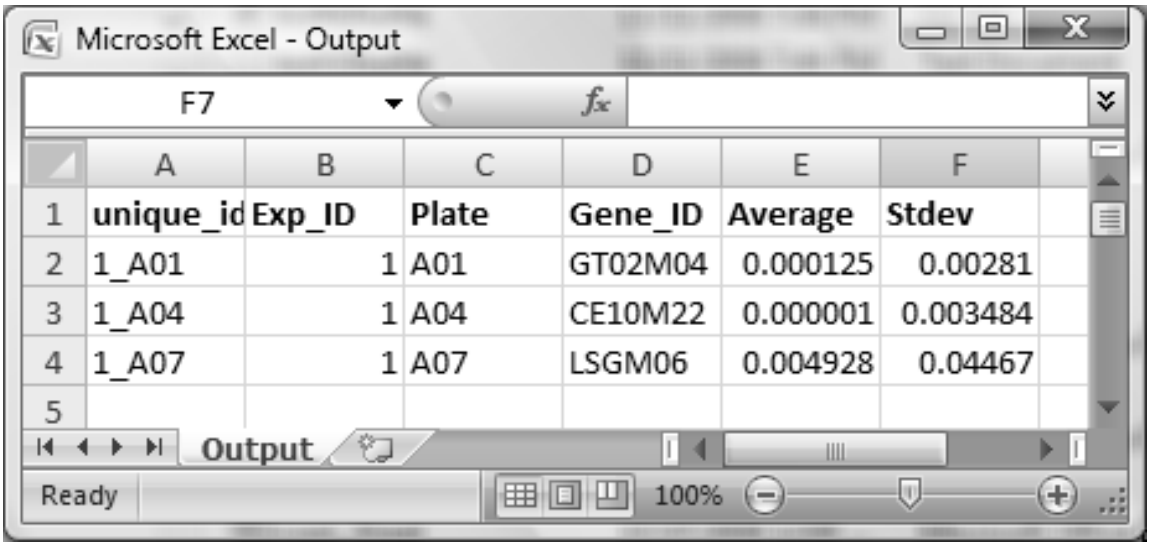


Figure 22: Sample RT-PCR Workflow Data

```
@prefix dc:  <http://www.owl-ontologies.com/rt-pcr.owl#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
<dc:Experiment_$Exp_ID>  <rdf:type>  <dc:Experiment> .
<dc:Experiment_$Exp_ID >  <dc:id>  < $Exp_ID > .
<dc:Experiment_$Exp_ID>  <dc:has_result>  <$unique_id> .
<dc:$unique_id>  <rdf:type>  <dc:RTPCRResult> .
<dc:Expression_Level_$unique_id> <rdf:type>  <dc:Enzyme_expression> .
<dc:RTPCRResult >  <dc:has_expression> <Expression_Level__$unique_id > .
<dc:Expression_Level__$unique_id> <dc:has_plate> <$Plate> .
<dc:Expression_Level__$unique_id> <dc:Average> <$Average> .
<dc:Expression_Level__$unique_id> <dc:Standard_deviation> <$Stdev> .
<dc:Expression_Level__$unique_id> <dc:Enzyme_name> <$Gene_ID> .
```

Figure 23: ADF for RT-PCR Workflow

The ADF is written using the N3 notation. The "$" symbol in a ADF indicates that the value following it is a name of a spreadsheet element such as a column name, and serves as a placeholder for the values which are present in the spreadsheet corresponding to that element. As an example, consider the third mapping entry in the ADF in Figure 23:

<dc:Experiment_$Exp_ID > <dc:has_result> < $unique_id> .

The annotation tasks in the workflow will replace the *$Exp_ID* and the *$unique_id* by the values which are in the spreadsheet for the column names *Exp_ID* and *unique_id respectively.* So, the annotated output for that mapping entry is as follows:

```
<dc:Experiment_1>  <dc:has_result>  <1_A01> .
<dc:Experiment_1>  <dc:has_result>  <1_A04> .
<dc:Experiment_1>  <dc:has_result>  <1_A07> .
```

This is how using TOAD we can annotate values in the spreadsheet data file with the corresponding ontology concepts. Figure 24 shows a part of the final annotated output and the relationship between the annotated output and the ontology concepts. Such annotation of the data with semantic metadata will help researchers to make a better sense of the data as well as to correlate between various data elements. As an example, from Figure 24 we can determine that the value 1_A01 in the spreadsheet data (Figure 22) is of the type RTPCRResult in the RT-PCR Ontology. It corresponds to the result obtained from plate A01 for the Experiment with ID as 1. It was run with the Enzyme GT02M04. The Average and Standard Deviation values for this gene triplet were 0.000125 and 0.00281, respectively. Thus, we see by annotating the data with semantic concepts, not only do we get more information about the data, but also learn about the correlations that exist within various parts of the data. Such information becomes particularly beneficial for researchers working on scientific workflows. Being data intensive, such workflows produce huge volumes of data, which often becomes incomprehensible for humans. Automating the semantic annotation of such experimental data will help researchers understand the data better.

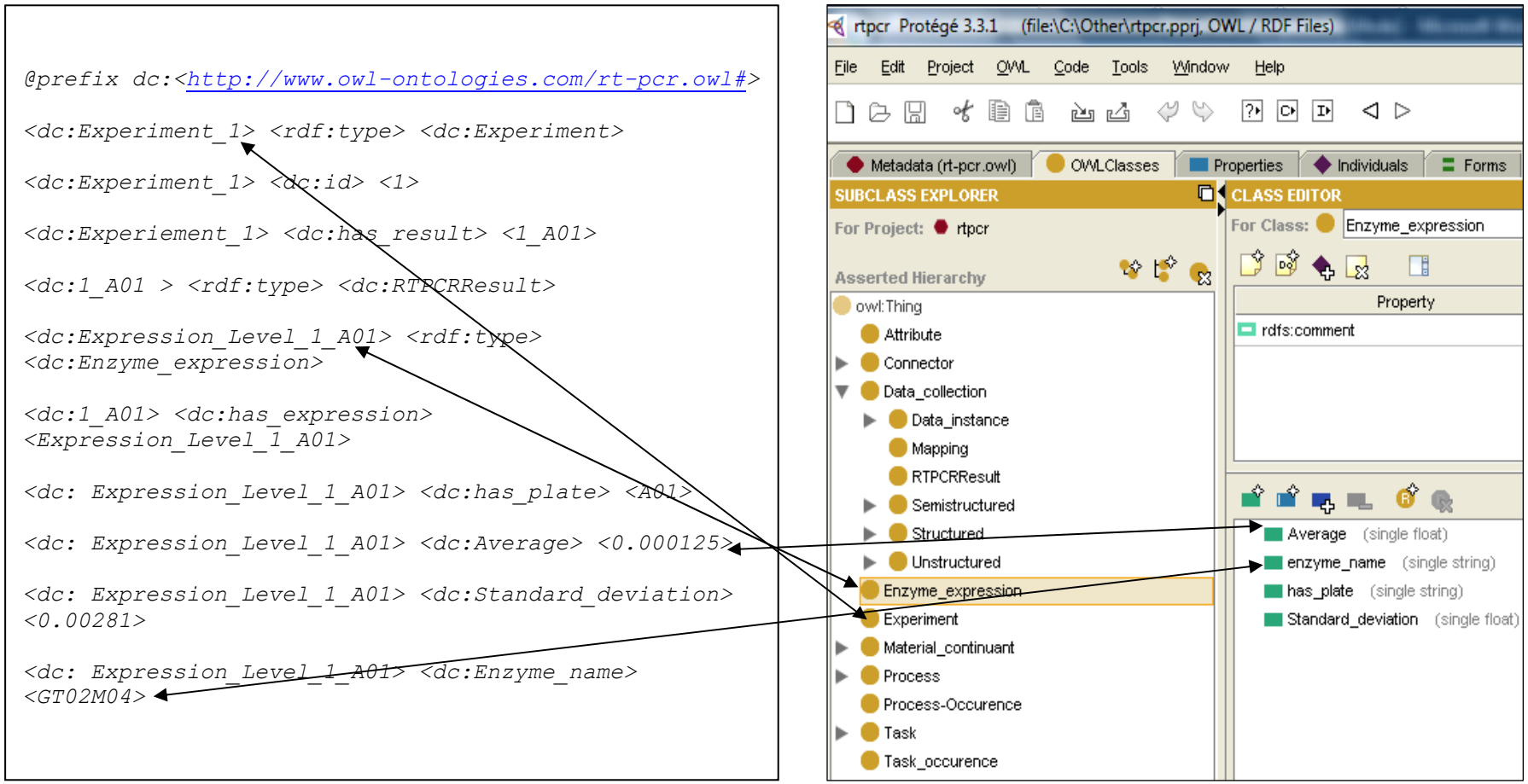


Figure 24: Annotation of RT-PCR Data

CHAPTER 6

CONCLUSION AND FUTURE WORK

Although much work has been done on the annotation of data found on the Web, very little work has been done on the subject of annotation of workflow data, in particular scientific workflows. Such workflows generally try to automate experimental processes which are data and computation intensive. The Web based workflow annotation tool, TOAD which we have In order to correlate among the vast array of data, as well as for machines to process the data, we need to annotate the data. We annotate such data with suitable concepts and relationships from the Ontology describing the protocols of the experiments producing the data. In this thesis, we introduced a way of annotating experimental data produced from scientific workflows using a Web-based annotation definition tool called TOAD (Tool for Ontology based Annotation of workflow Data). It is a generalized tool which works for any given ontology and any workflow data (in spreadsheet format for now) we choose. By adding semantic meta-data to the workflow data, we not only help the user to better understand the data, but also help him to find out correlations which hold within various constituents of the data.

Although at present, the tool has been designed to work with spreadsheet data only, it can easily be extended to include other sources of data such as structured data (in a relational database), semi-structured data as in a XML file and also unstructured data as in a text file. In case of text files, we can parse the text file using some regular expressions, or using line number, column number and extract data from there and annotate them with ontology concepts. In case of

semi-structured data represented as an XML file, we can query the data based on tag names using XQuery [76] and then annotate the query result using ontology concepts. And for structured data, for example, stored in a relational database, we can use SQL queries to retrieve the data and then annotate the data with the ontology concepts.

REFERENCES

1. Eds., H.C., *Handbook on Knowledge Management.* Springer, 2002.
2. Berners-Lee, T., J. Hendler, and O. Lassila, *The Semantic Web.* Scientific American, 2001
3. Seth, J.C.a.A.P., *Semantic Web Services, Processes and Applications.* 2006.
4. Gruber, T.R., *A Translation Approach to Portable Ontology Specifications.* Knowledge Acquisition, 1993. **5**(2): p. 199-220.
5. Hollingsworth, D., *Workflow Management Coalition: The Workflow Reference Model.* 1995.
6. Goble;, C., P. Missier;, and D.D. Roure, *Scientific workflow.* .
7. Pignotti, E., *Semantic Workflow Management.*
8. Bustin, S.A., *A-Z of Quantitative PCR.* International University Line, La Jolla, 2004.
9. jBPM, URL: *http://www.jboss.com/products/jbpm.*
10. Hammer, M.J.C., *Reengineering the Corporation: A manifesto for business Revolution.* Harper Business, 1993.
11. Frank Leymann, D.R., *Production Workflow concepts and techniques.*
12. Jablonski, S., *On the Complementarity of Workllow Management and Business Process Mode.* SIGOIS Bulletin, 1995. **16**(1): p. 33-38.
13. KEGG, URL: *http://www.genome.ad.jp/kegg/.*
14. M. Kanehisa, S.G., *KEGG: Kyoto Encyclopedia of Genes and Genomes.* Nucleic Acids Research, 2000. **28**(1): p. 27-30.

15. SweetDB, URL: http://www.glycosciences.de/sweetdb/index.php.

16. CarbBank, URL: http://biol.lancs.ac.uk/gig/pages/gag/carbbank.htm.

17. Web Services Architecture, URL: http://www.w3.org/TR/ws-arch/.

18. Roure, C.G.P.M.D.D., *Scientific workflows.*

19. W. M. P. van der Aalst, A.H.M.t.H., B.Kiepuszewski, and A. P. Barros,, *Workflow patterns.* Distributed and Parallel Databases, 2003. **14**(1): p. 5-51.

20. Bertram Ludascher, K.L., Shawn Bowers, Efrat Jaeger-Frank, Boyan Brodaric and Chaitan Baru, *Managing Scientific Data: From Data Integration to Scientific Workflows.*

21. I. Altintas, A.B., K. Baldridge, W. Sudholt, M. Miller, C. Amoreira, Y. Potier, and B. Ludaescher., *A Framework for the Design and Reuse of Grid Workflows.* International Workshop on Scientific Applications on Grid Computing (SAG'04), 2005. **LNCS 3458**.

22. T. Oinn, M.A., J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver and K. Glover, M.R. Pocock, A. Wipat, and P. Li., *Taverna: a tool for the composition and enactment of bioinformatics workflows.* Bioinformatics, 2004. **20**(17): p. 3045-3054.

23. I. Taylor, M.S., and I. Wang., *Resource management for the Triana peer-to-peer services.* 2003.

24. Dogac..., A., *Workflow Management Systems and Interoprability.*

25. jPDL, URL: http://docs.jboss.org/jbpm/v3/userguide/jpdl.html.

26. *The Grid: Blueprint for a Future Computing Infrastructure*, ed. I.F.a.C. Kesselman. 1999.

27. Hertzberger, Z.Z.A.B.H.Y.P.S.B., *Dynamic Workflow in a Grid Enabled Problem Solving Environment.*

28. Buyya, J.Y.a.R., *A Taxonomy of Workflow Management Systems for Grid Computing.*

29. Zhiming Zhao Adam Belloum Adianto Wibisono Frank Terpstra Piter T. de Boer, P.S.B.H., *Scientific workflow management: between generality and applicability.*

30. myGrid, URL: *http://www.mygrid.org.uk/*.

31. SCUFL, URL: *http://www.gridworkflow.org/snips/gridworkflow/space/SCUFL*.

32. MOML, URL: *http://www.gridworkflow.org/snips/gridworkflow/space/MoML*.

33. Ian J. Taylor, E.D., Dennis B. Gannon, Matthew Shields, *Workflows for E-science: Scientific Workflows for Grids*. 2007.

34. F. Curbera, Y.G., J. Klein, F. Leymann, D. Roller, S. Thatte, and S. Weerawarana., *Business Process Execution Language for Web Services, Version 1.1*. 2003.

35. jBPM Graphical Process Designer, URL: *http://docs.jboss.org/jbpm/v3/gpd/*.

36. ECLIPSE, URL: *http://www.eclipse.org/*.

37. Petia Wohed, B.A., Arthur H.M. ter Hofstede, Nick Russell, and Wil M.P. van der Aalst., *Patterns-based Evaluation of Open Source BPM Systems:The Cases of jBPM, OpenWFE, and Enhydra Shark.*

38. Hanson, J. *Manage your business processes with JBoss jBPM*. URL:http://www.javaworld.com/javaworld/jw-05-2006/jw-0522-jbpm.html.

39. Lyman, P.a.H.R.V., *How Much Information", 2003.* 2005.

40. John Davies, R.S., Paul Warren, *Semantic Web Technologies trends and research in onotology-based systems.* 2006.

41. XML, URL: *http://www.w3.org/XML/*.

42. RDF, URL: *http://www.w3.org/RDF/*.

43. Giovanni Aloisio, M.C., Italo Epicoco, Sandro Fiore, Maria Mirto, *A Semantic Grid-based Data Access and Integration Service for Bioinformatics.* IEEE International Symposium on Cluster Computing and the Grid, 2005.

44. Amit Sheth, W.Y., Christopher Thomas, Meenakshi Nagarajan, John A. Miller, Krys Kochut, Satya S. Sahoo, Xiaochuan Yi, *Semantic Web technology in support of Bioinformatics for Glycan Expression.*

45. Christopher J. Thomas, A.P.S., William S. York, *Modular Ontology Design Using Canonical Building Blocks in the Biochemistry Domain.* In Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS). 2006.

46. Staab S, S.R.E., *Handbook on Ontologies. International Handbooks on Information Systems.* Springer, 2004.

47. S.S. Sahoo, C.T., A.P. Sheth, W.S. York, S. Tartir, *Knowledge Modeling and its application in Life Sciences: A Tale of two Ontologies.* 2006: p. 317-326.

48. Stephen Dill, N.E., David Gibson, Daniel Gruhl, R. Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John A. Tomlin, and Jason Y. Zien, *SemTag and Seeker: Bootstrapping the Semantic Web via Automated Semantic Annotation.* Proceedings of the 12th international conference on World Wide Web, 2003: p. 178-186.

49. Siegfried Handschuh, S.S., *Authoring and annotation of web pages in CREAM.* Proceedings of the 11th international conference on World Wide Web, 2002: p. 462 - 473.

50. Koivunen, J.K.a.M.-R., *Annotea: an open RDF infrastructure for shared web annotations.* World Wide Web, 2001: p. 623-632.

51. Hendler., J.H.a.J., *Searching the web with shoe.* AAAI-2000 Workshop on AI for Web Search,, 2000.

52. Holmes., P.K.a.W., *AeroDAML: Applying information extraction to generate DAML annotations from web*

pages. 2001.

53. Atanas Kiryakov, B.P., Damyan Ognyanoff, Dimitar Manov, Angel Kirilov, Miroslav Goranov, *Semantic Annotation, Indexing, and Retrieval.* Journal of Web Semantics, 2005. **1**(2).

54. Abhijit Patil, S.O., Amit Sheth, Kunal Verma, *Meteor-s web service annotation framework.* Proceedings of the 13th international conference on World Wide Web, 2004: p. 553-562.

55. Workflow Patterns, URL: http://www.workflowpatterns.com/.

56. N. Russell, A.H.M.t.H., W.M.P. van der Aalst, and N. Mulyar., *Workflow Control-Flow Patterns: A Revised View.* Technical Report BPM Center Report BPM-06-22,BPMcenter.org, 2006.

57. N. Russell, A.H.M.t.H., D. Edmond, andW.M.P. van der Aalst, *Workflow Data Patterns: Identification, Representation and Tool Support.* editor, Proc. of the 24th Int. Conf. on Conceptual Modeling (ER 2005), 2005. **3716**: p. 353-368.

58. N. Russell, W.M.P.v.d.A., A.H.M. ter Hofstede, and D. Edmond, *Workflow Resource Patterns: Identification, Representation and Tool Support.* Proc. of the 17th Int. Conf. on Advanced Information Systems Engineering (CAiSE'05), 2005. **3520**: p. 216-232.

59. Lud¨ascher, S.B.a.B., *Actor-Oriented Design of Scientific Workflows.*

60. B. Lud¨ascher, I.A., D. H. Chad Berkley, E. Jaeger-Frank, M. Jones, E. Lee, J. Tao, and Y. Zhao, *Scientific Workflow Management and the Kepler System.*

61. W. M. P. van der Aalst, A.H.M.t.H., B. Kiepuszewski, and A. P. Barros., *Workflow Patterns. Distributed and Parallel Databases.* 2003. **14**(1): p. 5-51.

62. Nick Russell, A.H.M.t.H., David Edmond and Wil M.P. van der Aalst, *Workflow Data Patterns.*

63. jBPM user guide document, URL: *http://docs.jboss.org/jbpm/v3/userguide/taskmanagement.html.*

64. Notation 3, URL: *http://www.w3.org/DesignIssues/Notation3.html.*

65. *HTML 4.01 Specification.* 1999.

66. *Ajax*. URL: http://en.wikipedia.org/wiki/AJAX.

67. *jQuery.*

68. Glycan Structure Search using KCaM, URL: *http://www.genome.jp/ligand/kcam/.*

69. Nairn, A.V., York, W.S., Harris, K., Hall, E.M., Pierce, J.M. and Moremen. K.W., *Regulation of Glycan Structures in Animal Tissues: Transcript profiling of Glycan-Related Genes.* J. Biol. Chem., 2008. **283**(25).

70. GLYCOMICS, URL: *http://glycomics.ccrc.uga.edu/.*

71. CCRC, URL: *http://www.ccrc.uga.edu/.*

72. Bio-Rad, URL: *http://www.bio-rad.com/.*

73. Ramakers, C., Ruijter, J. M., Deprez, R. H., and Moorman, A. F., *Assumption-free analysis of quantitative real-time polymerase chain reaction (PCR) data.* Neuroscience Letters, 2003. **339**(1): p. 62-66.

74. *Protégé Ontology Editor and Knowledge Acquisition System.* URL: http://protege.stanford.edu/.

75. Sheth, A. *Semantic Web Technology in Support of Bioinformatics for Glycan Expression.* in *W3C workshop on Semantic Web for Life Sciences*. 2004. Cambridge MA.

76. *XQuery 1.0: An XML Query Language.*