P-3619

SEMANTIC WEBSERVICE DISCOVERY AND PROPAGATION

BASED ON MULTIAGENTS

A PROJECT REPORT

Submitted by

MANIMEKALA S **0710108028**

SARASWATHI R **0710108045**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

An Autonomous Institution Affiliated to Anna University of Technology,

COIMBATORE - 641 049

APRIL 2011

KUMARAGURU COLLEGE OF TECHNOLOGY,

COIMBATORE - 641 049

BONAFIDE CERTIFICATE

Certified that this project report "**SEMANTIC WEB SERVICE DISCOVERY AND PROPAGATION BASED ON MULTIAGENT**" is the bonafide work of "**MANIMEKALA S** and **SARASWATHI R**" who carried out the project under my supervision.

[illegible]	[illegible]
SIGNATURE	**SIGNATURE**
Mr.V.Subramani	Mrs.P.Devaki
SUPERVISIOR	**HEAD OF THE DEPARTMENT**
Associate Professor/CSE	Computer Science and Engineering
Kumaraguru College of Technology	Kumaraguru College of Technology
Coimbatore 641 049	Coimbatore 641 049

Submitted for the University Examination held on 20.04.11

[illegible]

Internal Examiner

[illegible]

External Examiner

DECLARATION

We

MANIMEKALA S	**0710108028**
SARASWATHI R	**0710108045**

Hereby declare that the project entitled **"SEMANTIC WEB SERVICE DISCOVERY AND PROPAGATION BASED ON MULTIAGENT"** is a record of original work done by us and to the best of our knowledge, a similar work has not been submitted to Anna University or any Institutions, for fulfillment for the award of the degree of bachelor of computer science and engineering of Autonomous Coimbatore-641 049.

Place: Coimbatore

Date: 20.04.11

S. Manimekala

MANIMEKALA .S

[illegible]

SARASWATHI .R

ACKNOWLEDGEMENT

We extend our sincere thanks to our principal Dr.Ramachandran, Kumaraguru College of Technology, Coimbatore for being a constant source of inspiration and providing us with necessary facility to work on this project.

We would like to make a special acknowledgement and thanks to **Dr.S.THANGASAMY,Ph.D.,** Dean , Professor and Head of Department of Computer Science and Engineering, for his support and encouragement throughout the project.

We express deep gratitude and gratefulness to our guide, **Mr.V.SUBRAMANI M.Tech.,** Department of Computer Science and Engineering, for his supervision, enduring patience, active involvement and guidance. We would also like to thank our project coordinator,**Mrs.P.DEVAKI M.S.,** Assistant Professor, Department of Computer Science and Engineering, for her support during the course of our project.

We would like to convey our honest thanks to Mr.V.Subramani for his consistent guidance throughout the project. We also thank all members of staff of the department and wealth of experience from which we greatly benefited.

S. Manimekala

MANIMEKALA .S

[illegible]

SARASWATHI .R

ABSTRACT

In Semantic web service discovery and propagation based on Multi agents in which seven components are used for efficient communication .The four components are inbuilt remaining three components designed in this project.

The first component is Request Analyzer in which query parameter extraction, input file is XML file, output file is search query parameters. XML parsers (DOM –Document Object Model or SAX-Simple API for XML) are used to extract the values from the XML document. The second component is Ontology management which taken domain of knowledge, extract the knowledge from the selected domain. It represents the information in terms of classes, slots, forms as well. Finally it builds the tree structure of domain. The third component is Recommender which validates the WSDL document using XML spy 2011. This project helps to improve the efficient of communication between semantic web service and FIPA(Foundation of Intelligent Physical Agents) multiagent .

ABBREVIATIONS

SWS	- Semantic Web Service
FIPA	- Foundation for Intelligent Physical Agent
WSDL	- Web Service Definition Language
SOAP	- Simple Object Access Protocol
UDDI	- Universal Description Discovery and Integration
RDF	- Resource Description Framework
OWL	- Web Ontology Language
DF	- Directory Facilitator
ACL	- Agent Communication Language
CU	- Control Unit
QOS	- Quality of Service

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO
	ABSTRACT	v
	LIST OF SYMBOLS	vi
1. INTRODUCTION		**1**
1.1	OVERVIEW	1
1.2	SEMANTIC WEBSERVICE DISCOVERY	
1.3	FIPA MULTIAGENT	
1.4	COMPONENTS	2
1.4.1	CONTROL UNIT(CU)	
1.4.2	WSDL2DF	
1.4.3	SOAP2ACL	
1.4.4	ONTOLOGY MANAGEMENT	3
1.4.5	OWL-S2WSDL	
1.4.6	RECOMMENDER	
1.4.7	REQUEST ANALYSER	4
2	**ANALYSIS OF PROBLEM**	**5**
2.1	EXISTING SYSTEM	
2.2	DRAWBACKS OF EXISTING SYSTEM	
2.3	PROPOSED SYSTEM	
2.4	ADVANTAGES OF PROPOSED SYSTEM	6
3	**SYSTEM ARCHTECTURE**	**7**
3.1	UDDI DIRECTORY	
3.2	DIRECTORY FACILITATOR	8
3.3	ACL	
3.4	SOAP	
4	**DESIGN AND IMPLEMENTATION**	**9**
4.1	MODULE DESCRIPTION	
4.1.1	REQUEST ANALYSER	
4.1.2	ONTOLOGY MANAGEMENT	11
4.1.3	RECOMMENDER	12

4.2	REQUIREMENT SPECIFICATION	13
4.2.1	HARDWARE REQUIREMENT	
4.2.2	SOFTWARE REQUIREMENT	
4.3	IMPLEMENTATION	
4.3.1	DESCRIPTION OF TOOLS	14
5	**TESTING**	**16**
	CONCLUSION	**17**
	APPENDIX	**18**
	SAMPLE CODING	**34**
	OUTPUT	**35**
	SCREEN SHOT	**37**
	REFERENCES	**58**

INTRODUCTION

1.INTRODUCTION

1.1 Overview

The aim of the semantic Web service is to describe and implement Web services so as to make them more accessible to automated agents. The key idea is to represent the functionality of a Web service. Seven components are used between the semantic web service (SWS) and FIPA (Foundation of Intelligent Physical Agents).

1.2 Semantic Web service Discovery

Web service is an application programming interfaces or web API's that can be accessed over a network, such as network.

Similar meaning of data can be discovered & accessed by the semantic Web service.

1.3 FIPA Multi agent

Foundation of intelligent physical agent which is an international non-profit association of companies promoting intelligent agents by developing specifications.

It is a system composed of multiple interacting intelligent agents which can be used to solve problems which are different to solve impossible for an individual agent or monolithic system.

1.4 Components

- Control Unit (CU)
- WSDL2DF
- SOAP2ACL
- Ontology management
- OWL-S2WSDL

- Recommender
- Request analyzer

1.4.1 Control Unit (CU)

It is responsible for communicating with FIPA multi agent and semantic Web service to provide Web services for them. From agents, it will receive queries regarding searching and publishing, in ACL (Agent Communication Language) format. Broker executes those queries on UDDI and sends reply in ACL message. From semantic Web services, it will receive queries regarding searching and publishing, in SOAP format.

1.4.2 WSDL2DF

Component is used by CU to translate WSDL to equivalent DF description

Component is used by CU to translate DF description to equivalent WSDL.

1.4.3 SOAP2ACL

Component is responsible for conversion of SOAP message to equivalent FIPA ACL message.

ACL2SOAP

Component is responsible for translation of FIPA ACL message to equivalent SOAP message.

1.4.4 Ontology management

Component is responsible for creating new generalized ontology by merging user ontology with a general ontology (i.e.WordNet, Yago, Wikipedia…).

1.4.5 OWL-S2WSDL

Component is used to translate new generalized ontology in OWL-S to equivalent WSDL because in this paper we assume that Web services are stored in UDDI in WSDL format.

1.4.6 Recommender

It is used to recommend created WSDL to selected Web service providers.

1.4.7 Request analyzer

It is responsible for analyzing the user request and recognizing the request type (discovery or propagation).

ANALYSIS OF PROBLEM

2 Analysis of problem

2.1 Existing System

1. Control Unit,

2. WSDL2DF, DF2WSDL,

3. SOAP2ACL, ACL2SOAP

4. OWL-S2WSDL

These four components are used .Even though these are not sufficient to develop the system.

2.2 Drawbacks of existing system

In that system there are four components. It doesn't provide the efficient communication between semantic web service discovery and FIPA multi agents.

This system doesn't have to extract the values and as well as it doesn't have the capability to represent the knowledge from the large domain.

There is no validation of WSDL document.

2.3 Proposed System

This system contains following components which are,

Request Analyzer

Ontology Management

Recommender

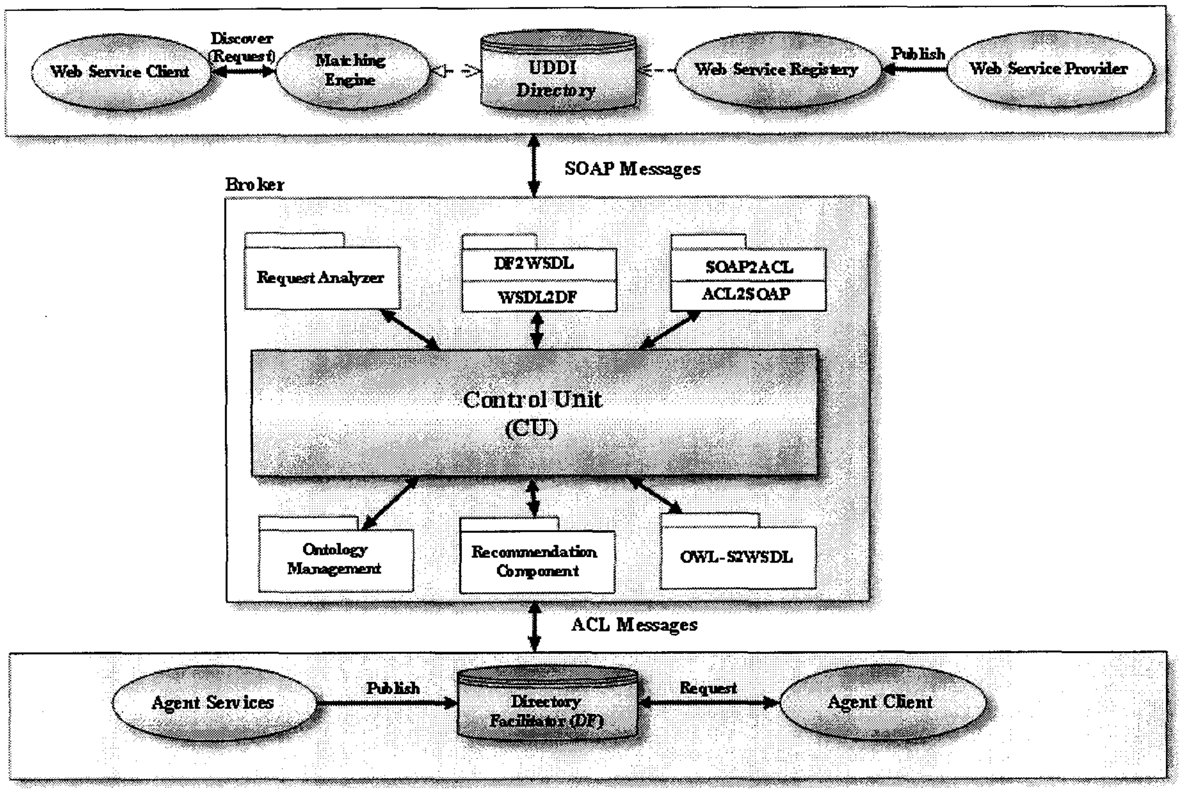
2.4 Advantages of proposed system

The proposed system which is having the ability to extract the query parameters from the document , Given clear cut idea to represent large domain concepts in terms of classes, slots and instances.

It can be represented by using Protégé 2000, v 1.8. Finally Tree structure generated for the whole organization. Recommender is used to validate the WSDL document.

SYSTEM ARCHITECTURE

3. SYSTEM ARCHTECTURE



3.1 UDDI DIRECTORY

A UDDI directory entry is an XML file that describes a business and the services it offers. There are three parts to an entry in the UDDI directory. They are commonly referred to as "*white pages", "yellow pages" and "green pages"*

3.2 DIRECTORY FACILITATOR

- Centralized registry of entries which associate service descriptions to agent IDs
- Considered the "yellow pages" of the system
- Agents can register their services w/ the DF
- Agents can search for other agent services in the DF

3.3 ACL

The FIPA Agent Communication Language (ACL) is based on speech act theory messages are actions, or *communicative acts*, as they are intended to perform some action by virtue of being sent. The specification consists of a set of message types and the description of their pragmatics, that is the effects on the mental attitudes of the sender and receiver agents. Every communicative act is described with both a narrative form and a formal

Semantics based on modal logic.

3.4 SOAP

It is important for application development to allow Internet communication between programs.

Today's applications communicate using Remote Procedure Calls (RPC) between objects like DCOM and CORBA, but HTTP was not designed for this. RPC represents a compatibility and security problem; firewalls and proxy servers will normally block this kind of traffic.

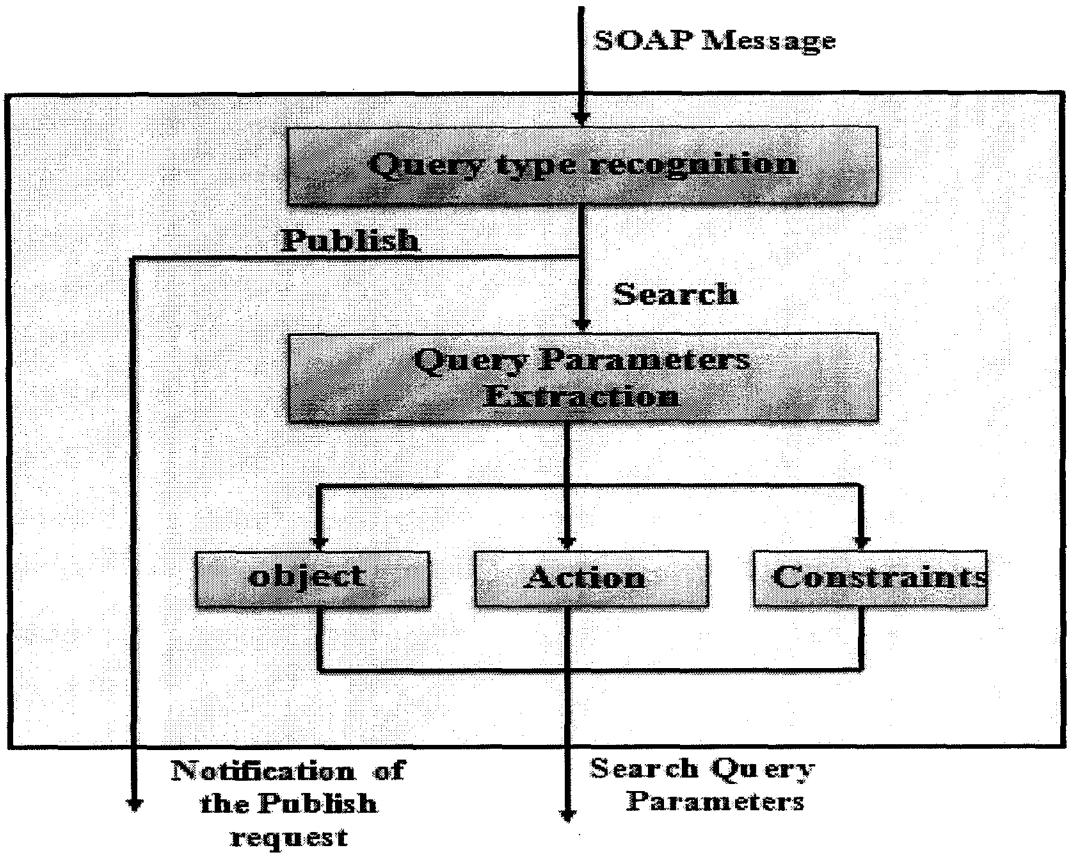
A better way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

DESIGN AND IMPLEMENTATION

4 Design and Implementation

4.1 Module Description

4.1.1 Request Analyzer

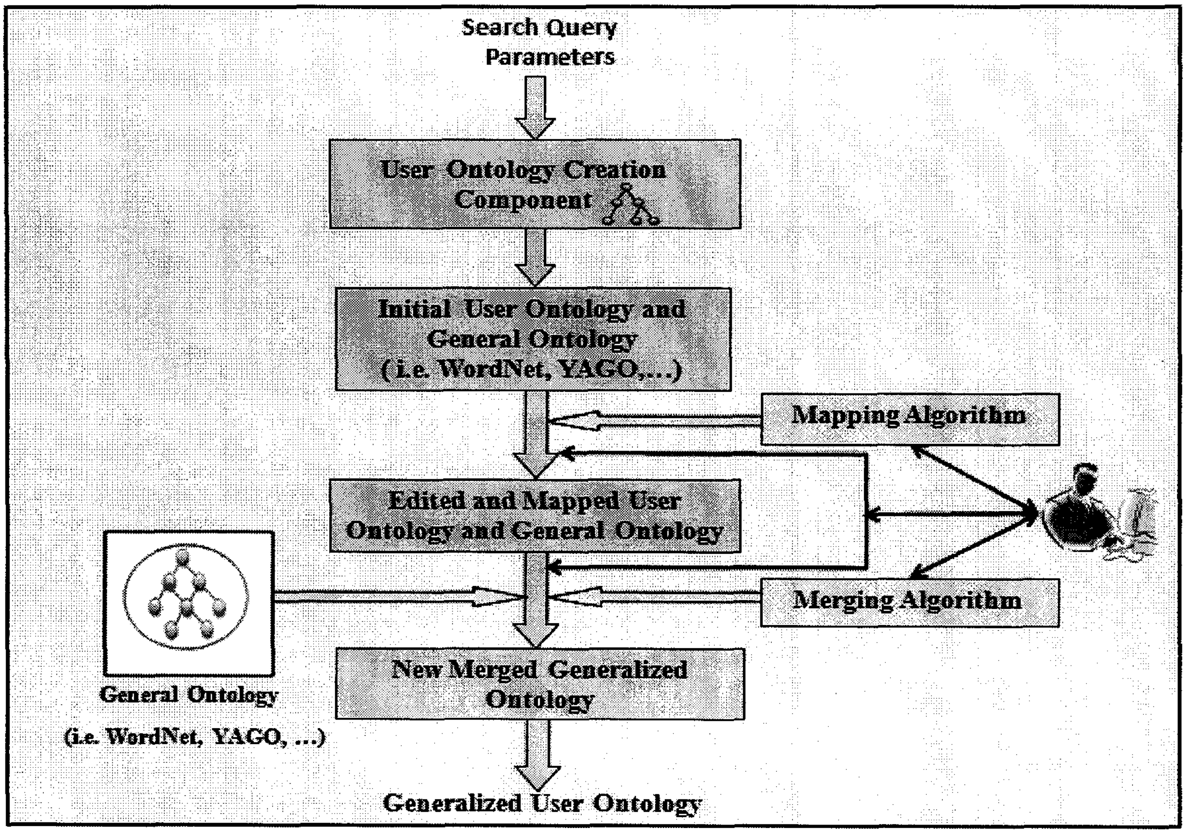


Once the CU receives SOAP message, it passes it to the request analyzer. The request analyzer receives the request, after being analyzed, if the request is propagation then a notification is sent to CU that the request is a kind of propagation, else if the request is a kind of search then it extracts out three major parameters based on Web services basic functional semantic description. Fb =< object, action, constraint >

Where object expresses the object of a service. Action

Expresses the behavior which is put on to one object. Constraint expresses the restraints must be obey when service is used.

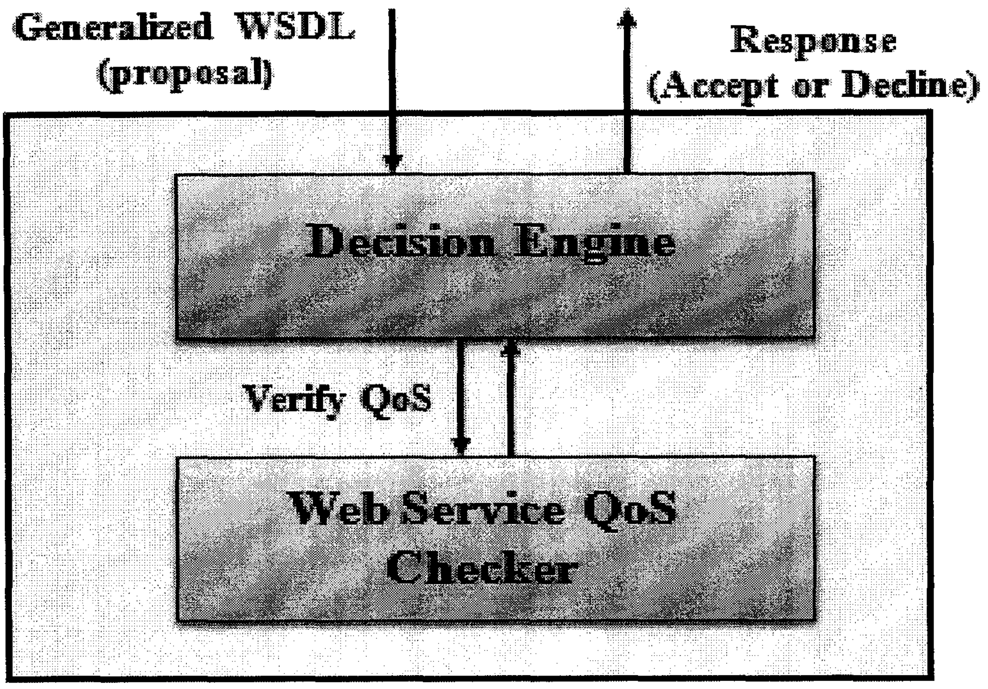
4.1.2 Ontology Management Component


Search Query Parameters
User Ontology Creation Component
Initial User Ontology and General Ontology (i.e. WordNet, YAGO,...)
Mapping Algorithm
Edited and Mapped User Ontology and General Ontology
Merging Algorithm
General Ontology (i.e. WordNet, YAGO, ...)
New Merged Generalized Ontology
Generalized User Ontology

Ontology management component receives extracted search query parameters and creates a user ontology based on main parameters and then it creates new generalized user ontology to discover appropriate Web services by merging user ontology with a general ontology

After the merging process, new generalized ontology is created and passed to OWL-S2WSDL component which describes whole scenario.

4.1.3 Recommender



It explains the proposed architecture of recommender in detail. In this article we assume that the decision engine is an independent component to evaluate generalized WSDL quality based on the functional and non-functional requirements for web service quality evaluation by Web service QoS checker. The decision engine sends the generalized WSDL to Web service QoS checker to evaluate QoS based on QoS measurement algorithm.

The generalized WSDL quality is evaluated and the response is sent to the decision engine whether the generalized WSDL has accepted or declined, after that decision engine forwards the response to CU.

Finally, CU recommends accepted WSDL to selected Web service providers, because if advertised Web services describe themselves more complete, their retrieval probability in the related queries will increase.

4.2 Requirement Specification

P- 3619

4.2.1 Hardware Requirement

1. Processor Type : INTEL Pentium Dual core inside

2. Processor Speed: 1.4 MHZ

3.RAM : 2 GB and above

4.Hard Disk capacity : 20GB

5.Monitor :14''

6.Mouse : Scroll Type

7.KeyBoard :101 keys

4.2.2 Software Requirement

1.operating System : Microsoft windows XP

2.Programming Language: JAVA

4.3 Implementation

4.3.1 Description of Tools

TOOL 1

Protégé 2000,v 1.8

It is used for knowledge representation.

ONTOLOGY

Ontology is a formal explicit description of concepts in a domain of discourse (**classes** (sometimes called **concepts**)), properties of each concept describing various features and attributes of the concept (**slots** (sometimes called **roles** or **properties**)), and restrictions on slots (**facets** (sometimes called **role restrictions**)). An ontology together with a set of individual **instances** of classes constitutes a **knowledge base**.

Classes are the focus of most ontologies. Classes describe concepts in the domain. A class can have **subclasses** that represent concepts that are more specific than the super class.

Slots describe properties of classes and instances

Developing ontology includes:

- defining classes in the ontology,
- arranging the classes in a taxonomic (subclass–super class) hierarchy,
- defining slots and describing allowed values for these slots,
- Filling in the values for slots for instances.

Why would someone want to develop ontology? Some of the reasons are:

- To share common understanding of the structure of information among people or software agents
- To enable reuse of domain knowledge
- To make domain assumptions explicit
- To separate domain knowledge from the operational knowledge
- To analyze domain knowledge

TOOL 2:

XMLSPY 2011

Altova XMLSpy 2011 Professional Edition is the industry-leading XML editor and XML development environment, with intuitive editing views and powerful XML utilities to model, edit, transform, and debug XML-related technologies quickly and easily.

In addition to the world?s most popular XML editor, XMLSpy Professional includes a graphical XML Schema editor that allows you to design and document complex schemas with ease. Schema flattening and subset creation are also supported. Advanced error handling and hyperlinking capabilities in its XML validator make troubleshooting a breeze. It also includes a schema-aware XSLT processor and an XSLT 1.0/2.0 debugger for perfecting stylesheets, plus support for stylesheets that use Java, C#, JavaScript, and VBScript. An XPath 1.0/2.0 analyzer and intelligent XPath auto-completion assist in the building and testing of XPath expressions, which can then be evaluated against multiple files.

A schema-aware XQuery processor and an XQuery debugger facilitate intelligent querying of XML data. Support for XBRL and XBRL Dimensions allows viewing and validating documents that conform to these standards. For Web developers, XMLSpy includes a JSON editor and JSON XML converter.

XMLSpy supports all major databases, including IBM DB2, SQL Server, Oracle, MySQL, Sybase, PostgreSQL, and others. You can connect to and query a relational database, generate an XML Schema from a DB, import and export data based on DB schemas, generate DBs from XML Schemas, and edit relational or XML data.

XMLSpy includes COM and Java APIs plus OLE and ActiveX controls, which let you access XMLSpy?s powerful capabilities in a programmatic way. The optional integration of XMLSpy with Microsoft Visual Studio or Eclipse allows you to seamlessly access XMLSpy editing views and XML development tools from within these popular IDEs.

TESTING

5. Testing

5.1 Module Level Testing

Module is defined as combination of related features to
Perform one major task in the application.

5.2 Specification Testing

An approach to testing wherein the testing is restricted to verifying the system/software meets the specification.

5.3 Validation Testing

It is the different set of activities which ensures that the software that has been built is traceable to the client requirements.

5.4 Performance Testing

Performance testing is testing that is performed, to determine how fast some aspect of a system performs under a particular workload. It can also serve to validate and verify other quality attributes of the system, such as scalability, reliability and resource usage.

5.5 Unit Testing

Unit testing is a method by which individual units of source code are tested to determine if they are fit for use. A unit is the smallest testable part of an application. In procedural programming a unit may be an individual function or procedure.

Conclusion

We must overcome some important challenges in order to realize the approach described above such as inconsistencies during the merge and evaluation of created WSDL based on the functional and non-functional requirements (i.e. quality of services). In the future work, the system may be applied on e-commerce. By recommending the generalized WSDL and improving Web service descriptions, the system can decrease the user interaction in these services and give Web services consumers and providers some confidence about the quality of service of the discovered and published Web services.

APPENDIX

Appendix

Sample coding

Request Analzer

```
import java.io.File;
import org.w3c.dom.Document;
import org.w3c.dom.*;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

public class newsservice
{

  public static void main (String argv [])
  {
  try
   {

     DocumentBuilderFactory docBuilderFactory = DocumentBuilderFactory.newInstance();
     DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder();
     Document doc = docBuilder.parse (new File("news.xml"));

     System.out.println();
     System.out.println();
     System.out.println("REQUEST ANALYZER");
     System.out.println("================");
     System.out.println();
     System.out.println();

     System.out.println("Search Query Parameters Extraction");
System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
System.out.println();

doc.getDocumentElement ().normalize ();
System.out.println ("Root element of the doc is " +
  doc.getDocumentElement().getNodeName());
System.out.println("************************************");
System.out.println();

NodeList listOfPersons = doc.getElementsByTagName("person");
int totalServices = listOfPersons.getLength();
System.out.println("Total no of Services : " + totalServices);
System.out.println("*****************************");
System.out.println();
for(int s=0; s<listOfPersons.getLength() ; s++)
 {
  Node firstPersonNode = listOfPersons.item(s);
  if(firstPersonNode.getNodeType() == Node.ELEMENT_NODE)

  {
   Element firstPersonElement = (Element)firstPersonNode;
   System.out.println("***AUTHOR INFORMATION***");
   System.out.println("------------------------");
   NodeList nameList = firstPersonElement.getElementsByTagName("name");
   Element nameElement = (Element)nameList.item(0);

   NodeList textNameList = nameElement.getChildNodes();
   System.out.println("Name : " +
     ((Node)textNameList.item(0)).getNodeValue().trim());
   //---------

   NodeList ageList = firstPersonElement.getElementsByTagName("age");
   Element ageElement = (Element)ageList.item(0);
```

```
NodeList textAgeList = ageElement.getChildNodes();
System.out.println("Age : " +
   ((Node)textAgeList.item(0)).getNodeValue().trim());

//----------

NodeList desgList = firstPersonElement.getElementsByTagName("desg");
Element desgElement = (Element)desgList.item(0);

NodeList textDesgList = desgElement.getChildNodes();
System.out.println("Desgination : " +
   ((Node)textDesgList.item(0)).getNodeValue().trim());

//----------
NodeList salaryList = firstPersonElement.getElementsByTagName("salary");
Element salaryElement = (Element)salaryList.item(0);

NodeList textSalaryList = salaryElement.getChildNodes();
System.out.println("Salary : " +
   ((Node)textSalaryList.item(0)).getNodeValue().trim());

//----------
NodeList addressList = firstPersonElement.getElementsByTagName("address");
Element addressElement = (Element)addressList.item(0);

NodeList textAddressList = addressElement.getChildNodes();
System.out.println("Address : " +
   ((Node)textAddressList.item(0)).getNodeValue().trim());

//----------
NodeList emailList = firstPersonElement.getElementsByTagName("email");
Element emailElement = (Element)emailList.item(0);
```

```
NodeList textEmailList = emailElement.getChildNodes();
System.out.println("Email : " +
   ((Node)textEmailList.item(0)).getNodeValue().trim());

//-----------
NodeList phonenoList = firstPersonElement.getElementsByTagName("phoneno");
Element phonenoElement = (Element)phonenoList.item(0);

NodeList textPhonenoList = phonenoElement.getChildNodes();
System.out.println("Phone no: " +
   ((Node)textPhonenoList.item(0)).getNodeValue().trim());

//-----------
NodeList expList = firstPersonElement.getElementsByTagName("exp");
Element expElement = (Element)expList.item(0);

NodeList textexpList = expElement.getChildNodes();
System.out.println("Experience : " +
   ((Node)textexpList.item(0)).getNodeValue().trim());

//-----------
//--------------------------------------------------------------

System.out.println();
System.out.println("***NEWS SERVICE INFORMATION***");
System.out.println("------------------------------");

NodeList contactList = firstPersonElement.getElementsByTagName("contact");
Element contactElement = (Element)contactList.item(0);

NodeList textContactList = contactElement.getChildNodes();
System.out.println("Contact Person : " +
   ((Node)textContactList.item(0)).getNodeValue().trim());

//-----------
NodeList newnoList = firstPersonElement.getElementsByTagName("newno");
Element newnoElement = (Element)newnoList.item(0);

NodeList textNewnoList = newnoElement.getChildNodes();
System.out.println("News Service No : " +
   ((Node)textNewnoList.item(0)).getNodeValue().trim());

//-----------
NodeList newagencyList = firstPersonElement.getElementsByTagName("newagency");
Element newagencyElement = (Element)newagencyList.item(0);

NodeList textNewagencyList = newagencyElement.getChildNodes();
System.out.println("News Service Agency : " +
   ((Node)textNewagencyList.item(0)).getNodeValue().trim());

//-----------
//--------------------------------------------------------
System.out.println();
System.out.println("***COLUMNIST INFORMATION***");
System.out.println("---------------------------");


NodeList name1List = firstPersonElement.getElementsByTagName("name1");
Element name1Element = (Element)name1List.item(0);

NodeList textName1List = name1Element.getChildNodes();
System.out.println("Name : " +
   ((Node)textName1List.item(0)).getNodeValue().trim());

//---------
```

```
NodeList salary1List = firstPersonElement.getElementsByTagName("salary1");
Element salary1Element = (Element)salary1List.item(0);

NodeList textSalary1List = salary1Element.getChildNodes();
System.out.println("Salary : " +
   ((Node)textSalary1List.item(0)).getNodeValue().trim());

//-----------

NodeList dateList = firstPersonElement.getElementsByTagName("date");
Element dateElement = (Element)dateList.item(0);

NodeList textdateList = dateElement.getChildNodes();
System.out.println("Date Hired : " +
   ((Node)textdateList.item(0)).getNodeValue().trim());

//-----------
NodeList currentList = firstPersonElement.getElementsByTagName("current");
Element currentElement = (Element)currentList.item(0);

NodeList textcurrentList = currentElement.getChildNodes();
System.out.println("Current Job Title : " +
   ((Node)textcurrentList.item(0)).getNodeValue().trim());

//-----------
NodeList phoneno1List = firstPersonElement.getElementsByTagName("phoneno1");
Element phoneno1Element = (Element)phoneno1List.item(0);

NodeList textPhoneno1List = phoneno1Element.getChildNodes();
System.out.println("Phone no: " +
   ((Node)textPhoneno1List.item(0)).getNodeValue().trim());
```

```
//-----------
//------------------------------------------------------

System.out.println();
System.out.println("***EDITOR INFORMATION***");
System.out.println("--------------------------");

NodeList name2List = firstPersonElement.getElementsByTagName("name2");
Element name2Element = (Element)name2List.item(0);

NodeList textName2List = name2Element.getChildNodes();
System.out.println("Name : " +
   ((Node)textName2List.item(0)).getNodeValue().trim());

//---------

NodeList salary2List = firstPersonElement.getElementsByTagName("salary2");
Element salary2Element = (Element)salary2List.item(0);

NodeList textSalary2List = salary2Element.getChildNodes();
System.out.println("Salary : " +
   ((Node)textSalary2List.item(0)).getNodeValue().trim());



//-----------

NodeList jobList = firstPersonElement.getElementsByTagName("job");
Element jobElement = (Element)jobList.item(0);

NodeList textjobList = jobElement.getChildNodes();
System.out.println("Job Name : " +
   ((Node)textjobList.item(0)).getNodeValue().trim());
```

```
//----------
NodeList sectionList = firstPersonElement.getElementsByTagName("section");
Element sectionElement = (Element)sectionList.item(0);

NodeList textsectionList = sectionElement.getChildNodes();
System.out.println("Section Name : " +
   ((Node)textsectionList.item(0)).getNodeValue().trim());

//----------

NodeList date1List = firstPersonElement.getElementsByTagName("date1");
Element date1Element = (Element)date1List.item(0);

NodeList textdate1List = date1Element.getChildNodes();
System.out.println("Date Hired : " +
   ((Node)textdate1List.item(0)).getNodeValue().trim());
//----------
NodeList phoneno2List = firstPersonElement.getElementsByTagName("phoneno2");
Element phoneno2Element = (Element)phoneno2List.item(0);

NodeList textPhoneno2List = phoneno2Element.getChildNodes();
System.out.println("Phone no: " +
   ((Node)textPhoneno2List.item(0)).getNodeValue().trim());

//----------
//-----------------------------------------------------------


System.out.println();
System.out.println("***REPORTER INFORMATION***");
System.out.println("-------------------------");
```

```
NodeList name3List = firstPersonElement.getElementsByTagName("name3");
Element name3Element = (Element)name3List.item(0);

NodeList textName3List = name3Element.getChildNodes();
System.out.println("Name : " +
   ((Node)textName3List.item(0)).getNodeValue().trim());

//---------

NodeList salary3List = firstPersonElement.getElementsByTagName("salary3");
Element salary3Element = (Element)salary3List.item(0);

NodeList textSalary3List = salary3Element.getChildNodes();
System.out.println("Salary : " +
   ((Node)textSalary3List.item(0)).getNodeValue().trim());


//-----------
//-------------------------------------------------------


System.out.println();
System.out.println("***CONTENT INFORMATION***");
System.out.println("--------------------------");



NodeList expdateList = firstPersonElement.getElementsByTagName("expdate");
Element expdateElement = (Element)expdateList.item(0);

NodeList textExpdateList = expdateElement.getChildNodes();
System.out.println("Expiration Date : " +
   ((Node)textExpdateList.item(0)).getNodeValue().trim());
```

```
//-----------------------------------------------------------


System.out.println();
System.out.println("***ADVERTISEMENT INFORMATION***");
System.out.println("------------------------------");

NodeList name4List = firstPersonElement.getElementsByTagName("name4");
Element name4Element = (Element)name4List.item(0);

NodeList textName4List = name4Element.getChildNodes();
System.out.println("Name : " +
   ((Node)textName4List.item(0)).getNodeValue().trim());

//---------
NodeList expdate1List = firstPersonElement.getElementsByTagName("expdate1");
Element expdate1Element = (Element)expdate1List.item(0);

NodeList textExpdate1List = expdate1Element.getChildNodes();
System.out.println("Expiration Date : " +
   ((Node)textExpdate1List.item(0)).getNodeValue().trim());
//---------

NodeList pageno1List = firstPersonElement.getElementsByTagName("pageno1");
Element pageno1Element = (Element)pageno1List.item(0);

NodeList textpageno1List = pageno1Element.getChildNodes();
System.out.println("Page No : " +
   ((Node)textpageno1List.item(0)).getNodeValue().trim());
 //---------

NodeList layout1List = firstPersonElement.getElementsByTagName("layout1");
Element layout1Element = (Element)layout1List.item(0);
```

```
NodeList textlayout1List = layout1Element.getChildNodes();
System.out.println("Type of Layout : " +
   ((Node)textlayout1List.item(0)).getNodeValue().trim());
 //---------

NodeList urgent1List = firstPersonElement.getElementsByTagName("urgent1");
Element urgent1Element = (Element)urgent1List.item(0);

NodeList texturgent1List = urgent1Element.getChildNodes();
System.out.println("Type of Delivery : " +
   ((Node)texturgent1List.item(0)).getNodeValue().trim());
 //---------

NodeList published1List = firstPersonElement.getElementsByTagName("published1");
Element published1Element = (Element)published1List.item(0);

NodeList textpublished1List = published1Element.getChildNodes();
System.out.println("Date of Publishing : " +
   ((Node)textpublished1List.item(0)).getNodeValue().trim());
//------------
//--------------------------------------------------------------

 System.out.println();
 System.out.println("***ARTICLE INFORMATION***");
 System.out.println("------------------------------");

 NodeList t1List = firstPersonElement.getElementsByTagName("t1");
 Element t1Element = (Element)t1List.item(0);

 NodeList textt1List = t1Element.getChildNodes();
 System.out.println("Text Type : " +
   ((Node)textt1List.item(0)).getNodeValue().trim());
```

```
//---------

NodeList pagenoList = firstPersonElement.getElementsByTagName("pageno");
Element pagenoElement = (Element)pagenoList.item(0);

NodeList textpagenoList = pagenoElement.getChildNodes();
System.out.println("Page No : " +
   ((Node)textpagenoList.item(0)).getNodeValue().trim());
 //---------

NodeList layoutList = firstPersonElement.getElementsByTagName("layout");
Element layoutElement = (Element)layoutList.item(0);

NodeList textlayoutList = layoutElement.getChildNodes();
System.out.println("Type of Layout : " +
   ((Node)textlayoutList.item(0)).getNodeValue().trim());
 //---------

NodeList urgentList = firstPersonElement.getElementsByTagName("urgent");
Element urgentElement = (Element)urgentList.item(0);

NodeList texturgentList = urgentElement.getChildNodes();
System.out.println("Type of Delivery : " +
   ((Node)texturgentList.item(0)).getNodeValue().trim());
 //---------

NodeList publishedList = firstPersonElement.getElementsByTagName("published");
Element publishedElement = (Element)publishedList.item(0);

NodeList textpublishedList = publishedElement.getChildNodes();
System.out.println("Date of Publishing : " +
   ((Node)textpublishedList.item(0)).getNodeValue().trim());
//-----------
```

```
//-----------



NodeList headlineList = firstPersonElement.getElementsByTagName("headline");
Element headlineElement = (Element)headlineList.item(0);

NodeList textheadlineList = headlineElement.getChildNodes();
System.out.println("Headline Type : " +
   ((Node)textheadlineList.item(0)).getNodeValue().trim());
//---------
NodeList levelList = firstPersonElement.getElementsByTagName("level");
Element levelElement = (Element)levelList.item(0);

NodeList textlevelList = levelElement.getChildNodes();
System.out.println("Level to Read : " +
   ((Node)textlevelList.item(0)).getNodeValue().trim());
//---------
NodeList atypeList = firstPersonElement.getElementsByTagName("atype");
Element atypeElement = (Element)atypeList.item(0);

NodeList textatypeList = atypeElement.getChildNodes();
System.out.println("Article Type: " +
   ((Node)textatypeList.item(0)).getNodeValue().trim());
//---------
NodeList keyList = firstPersonElement.getElementsByTagName("key");
Element keyElement = (Element)keyList.item(0);

NodeList textkeyList = keyElement.getChildNodes();
System.out.println("Key of the Newspaper : " +
   ((Node)textkeyList.item(0)).getNodeValue().trim());
//-----------
//------------------------------------------------------------
```

```
System.out.println();
System.out.println("***ORGANIZATION INFORMATION***");
System.out.println("-------------------------------");



NodeList elistList = firstPersonElement.getElementsByTagName("elist");
Element elistElement = (Element)elistList.item(0);

NodeList textelistList = elistElement.getChildNodes();
System.out.println("No of Employees who are working in news dept : " +
   ((Node)textelistList.item(0)).getNodeValue().trim());
//------------
NodeList elList = firstPersonElement.getElementsByTagName("el");
Element elElement = (Element)elList.item(0);

NodeList textelList = elElement.getChildNodes();
System.out.println("Chart Type : " +
   ((Node)textelList.item(0)).getNodeValue().trim());
//------------




NodeList ptypeList = firstPersonElement.getElementsByTagName("ptype");
Element ptypeElement = (Element)ptypeList.item(0);

NodeList textptypeList = ptypeElement.getChildNodes();
System.out.println("Prototype of the Newspaper : " +
   ((Node)textptypeList.item(0)).getNodeValue().trim());
//------------
 System.out.println();
```

```
      System.out.println("***************************");
      System.out.println("***************************");




      }



   }



  }
  catch (SAXParseException err)
  {
   System.out.println ("** Parsing error" + ", line "
    + err.getLineNumber () + ", uri " + err.getSystemId ());
   System.out.println(" " + err.getMessage ());

  }
  catch (SAXException e)
  {
  Exception x = e.getException ();
  ((x == null) ? e : x).printStackTrace ();

  }
  catch (Throwable t)
  {
  t.printStackTrace ();
  }


}
}
```

OUTPUT

```
REQUEST ANALYZER
================

Search Query Parameters Extraction
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Root element of the doc is newspaper
*********************************

Total no of Services : 1
**************************

***AUTHOR INFORMATION***
-----------------------
Name : saras
Age : 21
Desgination : Manager
Salary : 20,000
Address : Dharapuram
Email : suvanika2009@gmail.com
Phone no: 9790206892
Experience : 2 years

***NEWS SERVICE INFORMATION***
-----------------------------
Contact Person : Sathya
News Service No : 785
News Service Agency : Vijaya Agency
```

```
***COLUMNIST INFORMATION***
---------------------------
Name : Sanju
Salary : 16,000
Date Hired : 17.08.2010
Current Job Title : Clerk
Phone no: 9994206032

***EDITOR INFORMATION***
---------------------------
Name : kani
Salary : 20,000
Job Name : Editing
Section Name : Technical Section
Date Hired : 16.07.2007
Phone no: 9942277173

***REPORTER INFORMATION***
---------------------------
Name : santhanam
Salary : 12,000

***CONTENT INFORMATION***
---------------------------
Expiration Date : 17.08.2013
Page No : 123
Type of Layout : Content Layout
Type of Delivery : Content Delivery
Date of Publishing : 02.02.2011
```

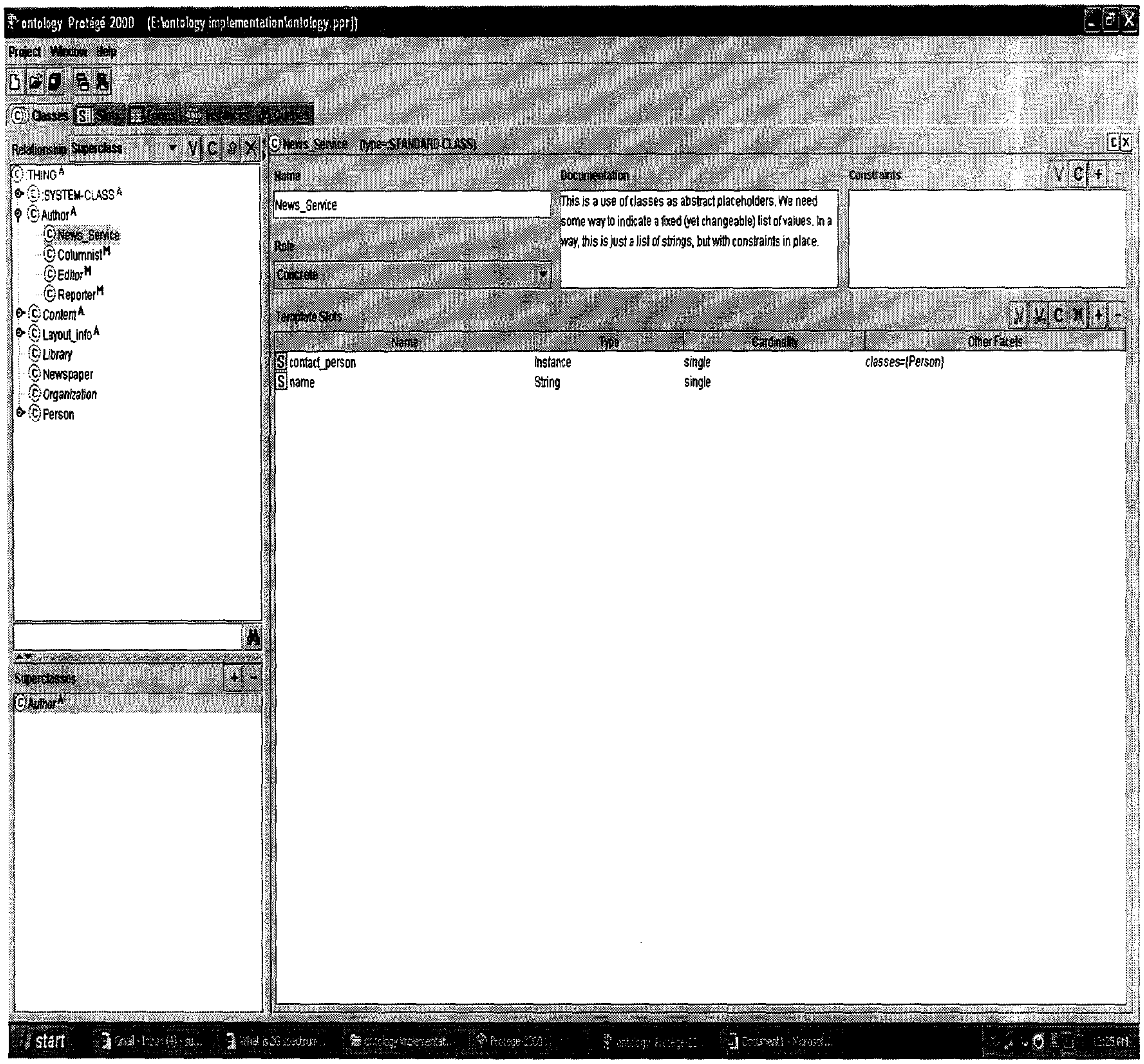
```
***ADVERTISEMENT INFORMATION***
------------------------------
Name : chithra
Expiration Date : 19.05.2014
Page No : 0089
Type of Layout : Content Layout
Type of Delivery : Content Delivery
Date of Publishing : 02.02.2011

***ARTICLE INFORMATION***
------------------------------
Text Type : Normal Text
Headline Type : Bold Letters
Level to Read : 15
Article Type: Technical,Normal,Biography
Key of the Newspaper : Information

***ORGANIZATION INFORMATION***
------------------------------
No of Employees who are working in news dept : 10,000
Chart Type : Pie Chart
Prototype of the Newspaper : General

***************************
***************************
```

SCREEN SHOT



News_Service (type=:STANDARD-CLASS)

Name: News_Service

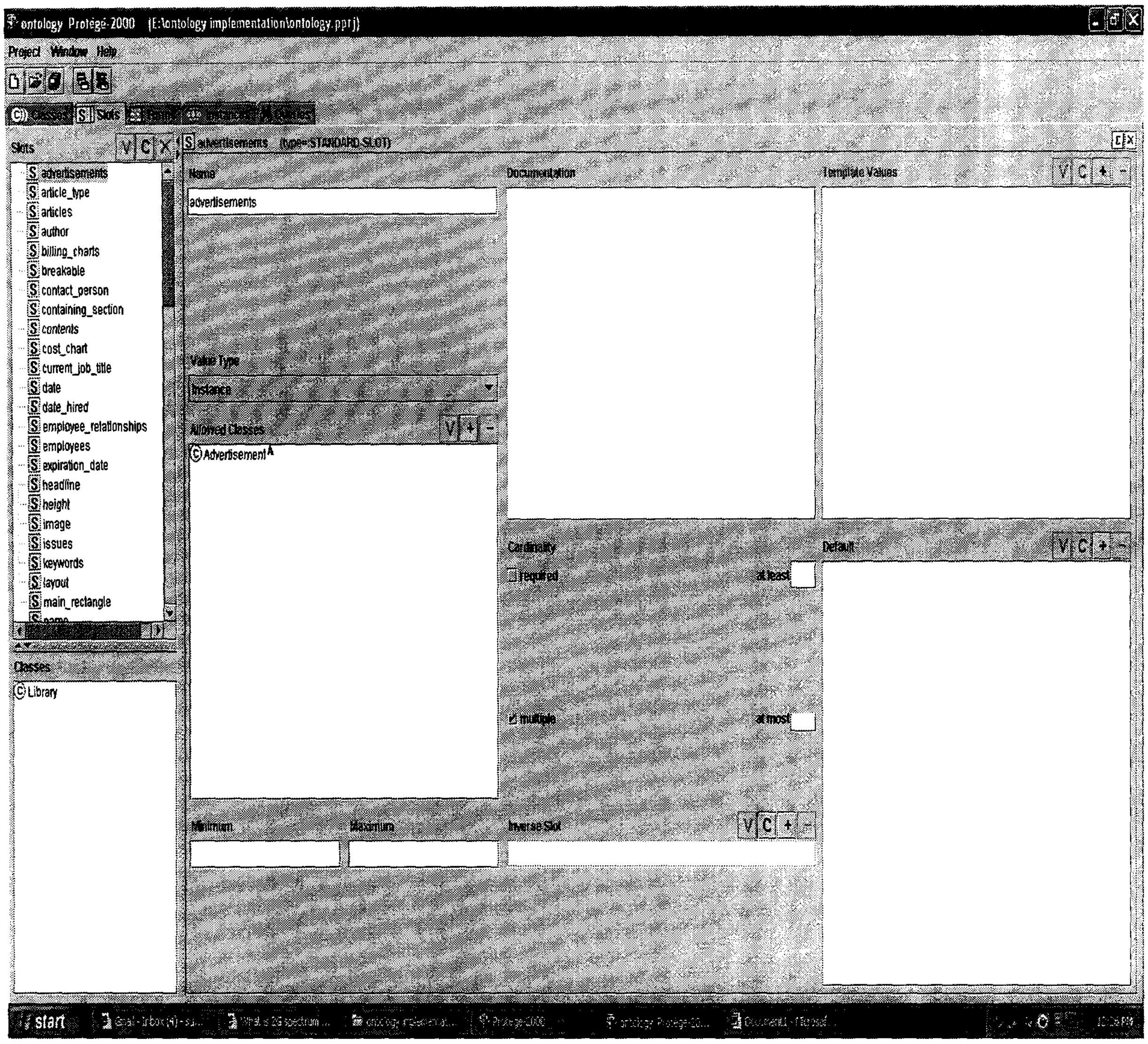
Role: Concrete

Documentation: This is a use of classes as abstract placeholders. We need some way to indicate a fixed (yet changeable) list of values. In a way, this is just a list of strings, but with constraints in place.

Constraints:

Template Slots

Name	Type	Cardinality	Other Facets
S contact_person	Instance	single	classes={Person}
S name	String	single	



ontology Protégé-2000 (E:\ontology implementation\ontology.pprj)

Slots
- advertisements
- article_type
- articles
- author
- billing_charts
- breakable
- contact_person
- containing_section
- contents
- cost_chart
- current_job_title
- date
- date_hired
- employee_relationships
- employees
- expiration_date
- headline
- height
- image
- issues
- keywords
- layout
- main_rectangle

Classes
- Library

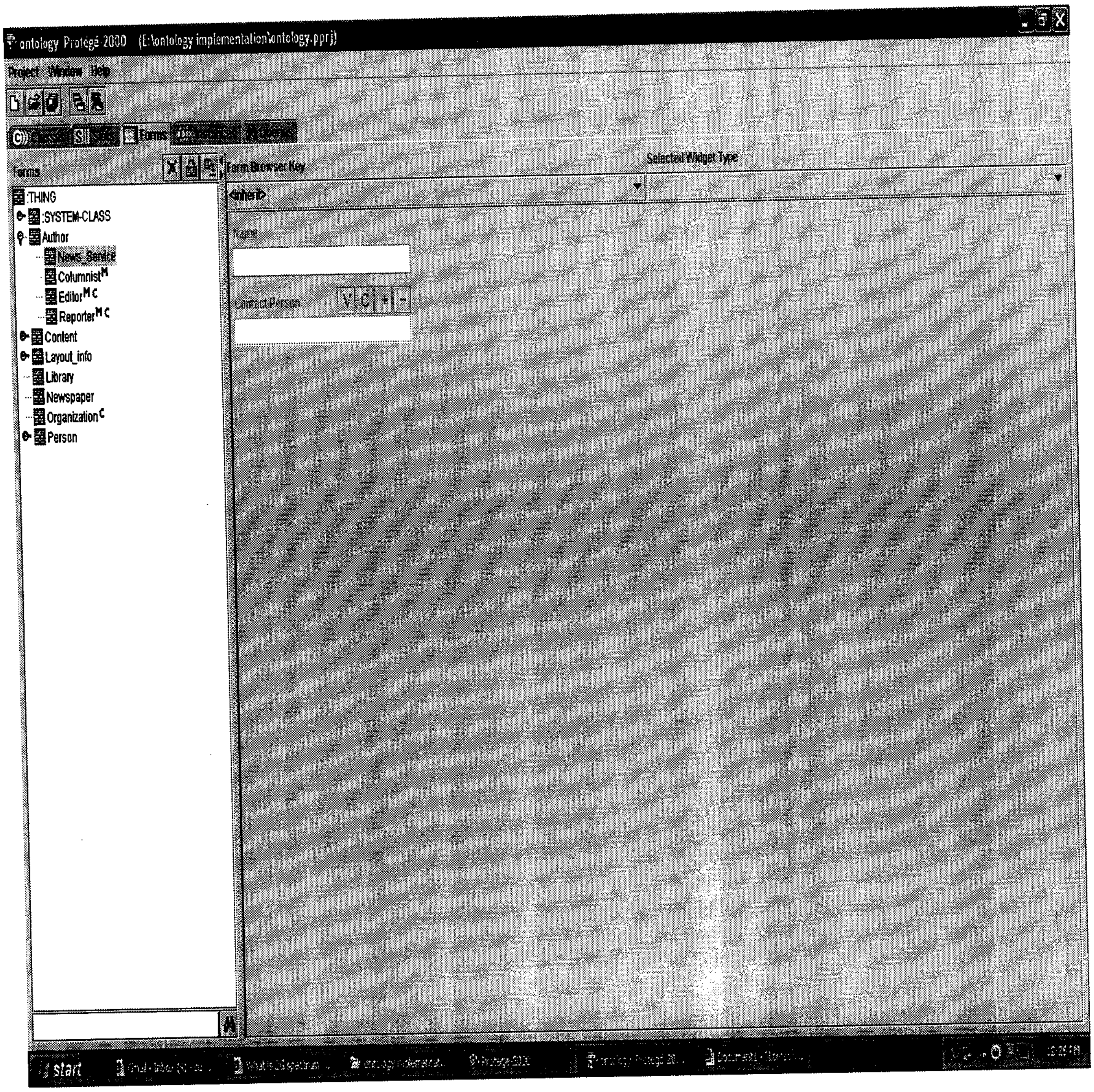
advertisements (type=:STANDARD-SLOT)

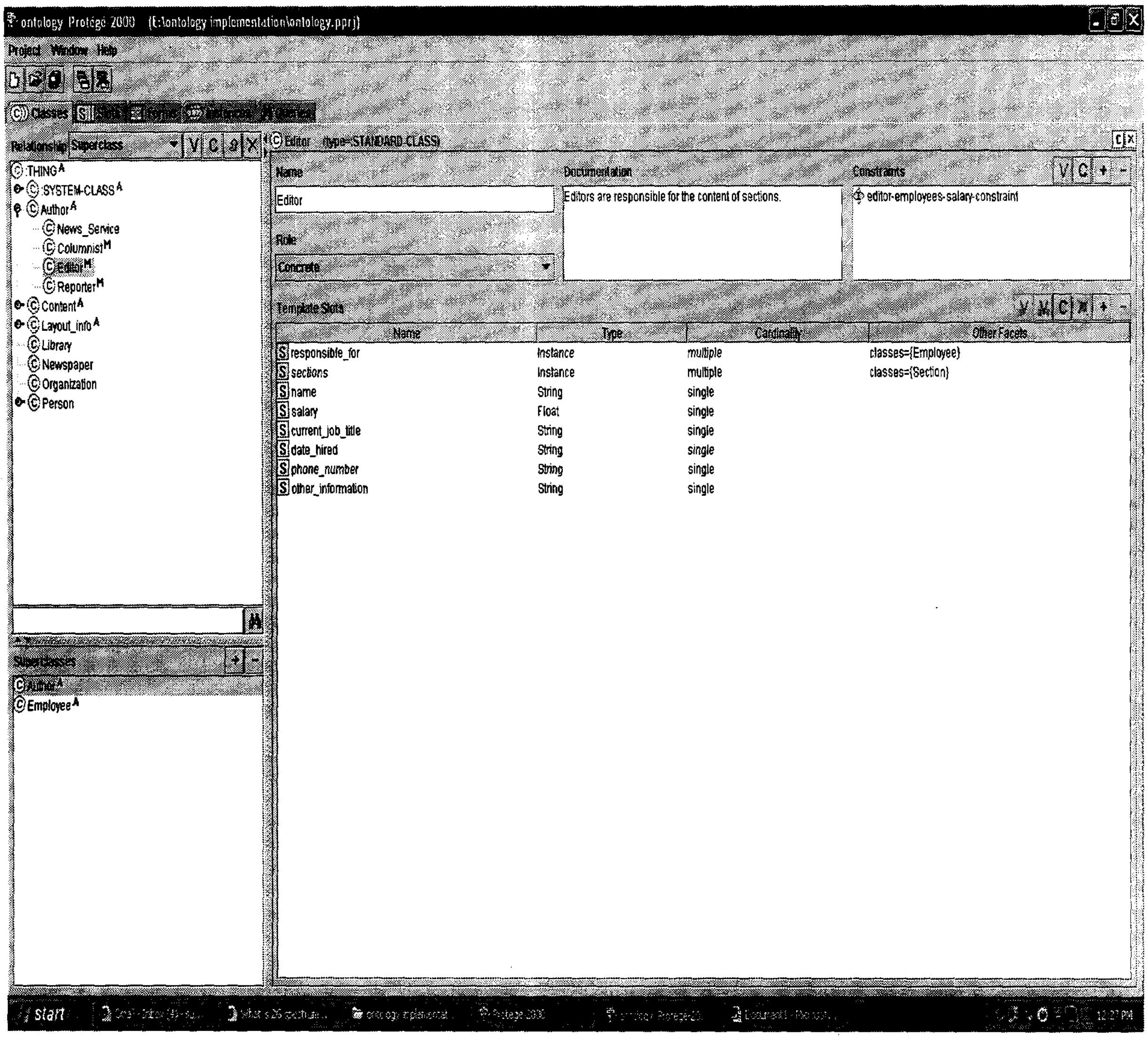
Name: advertisements

Value Type: Instance

Allowed Classes: Advertisement

Cardinality: ☐ required at least ☐ ☑ multiple at most


ontology Protégé 2000 (E:\ontology implementation\ontology.pprj)
Project Window Help
Classes
Slots
Forms
Instances
Queries
Forms
Form Browser Key
Selected Widget Type
<inherit>
:THING
:SYSTEM-CLASS
Author
News_Service
Columnist M
Editor M C
Reporter M C
Content
Layout_info
Library
Newspaper
Organization C
Person
Name
Contact Person
start



Editor (type=:STANDARD-CLASS)

Name: Editor

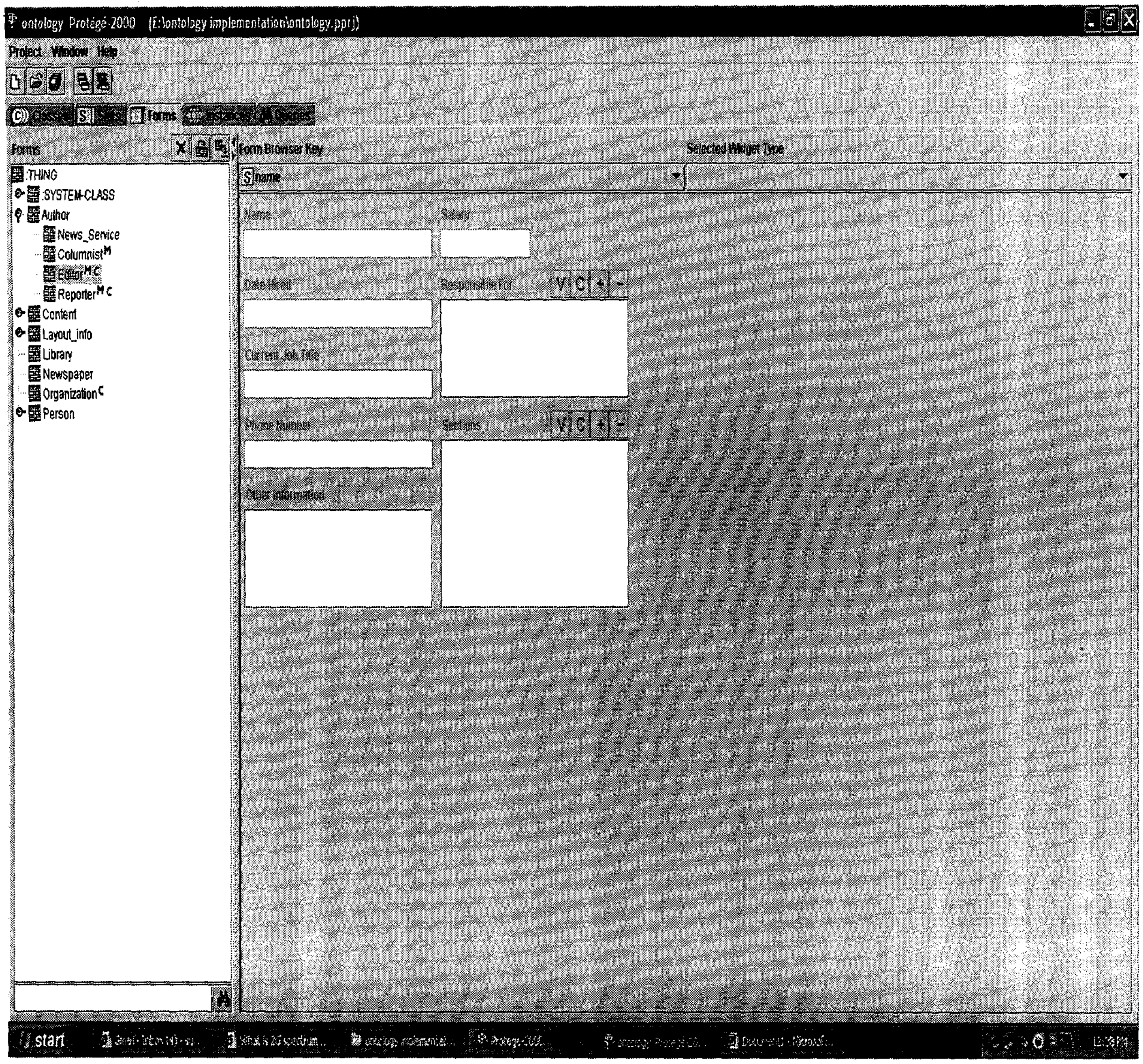
Role: Concrete

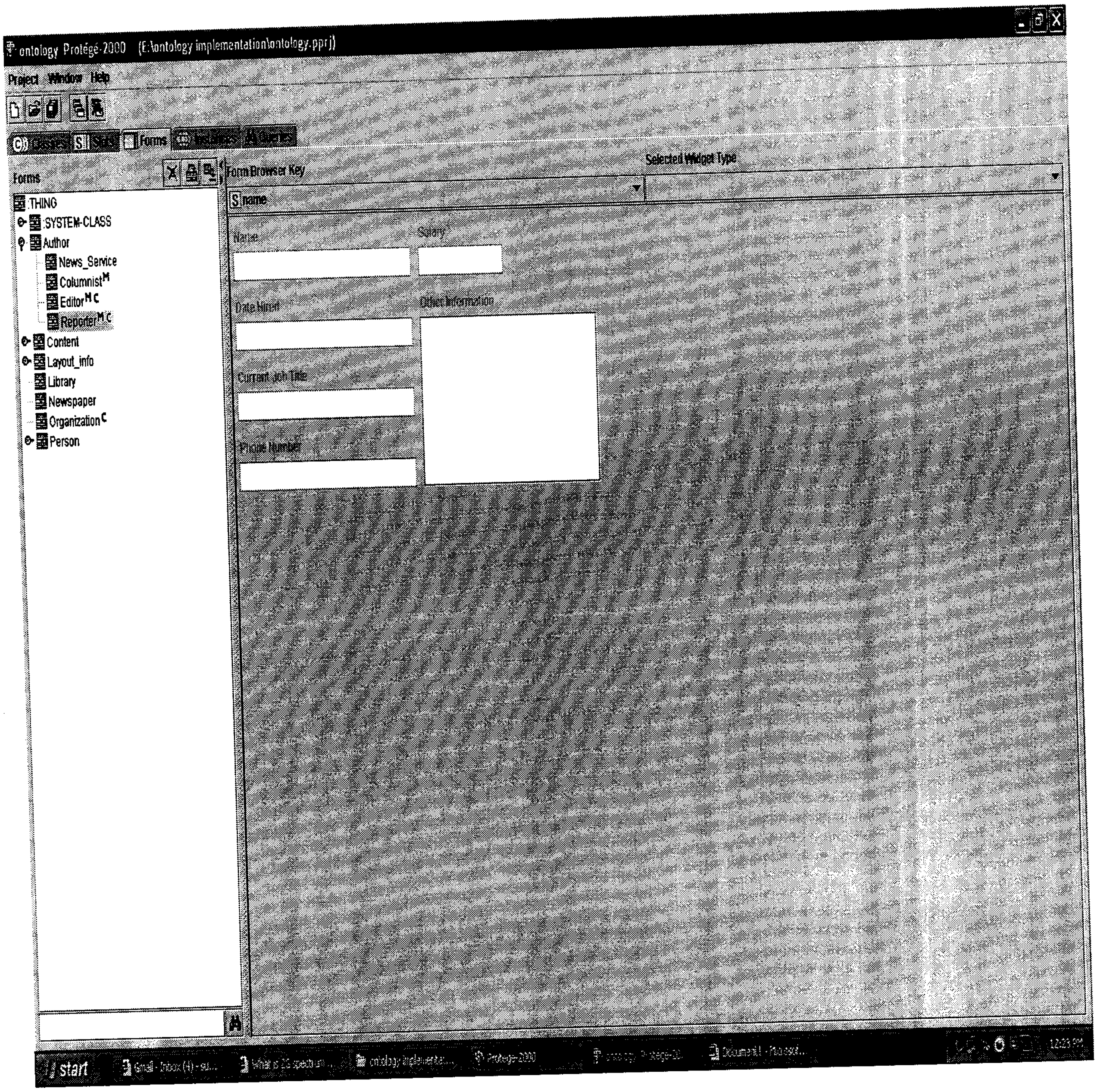
Documentation: Editors are responsible for the content of sections.

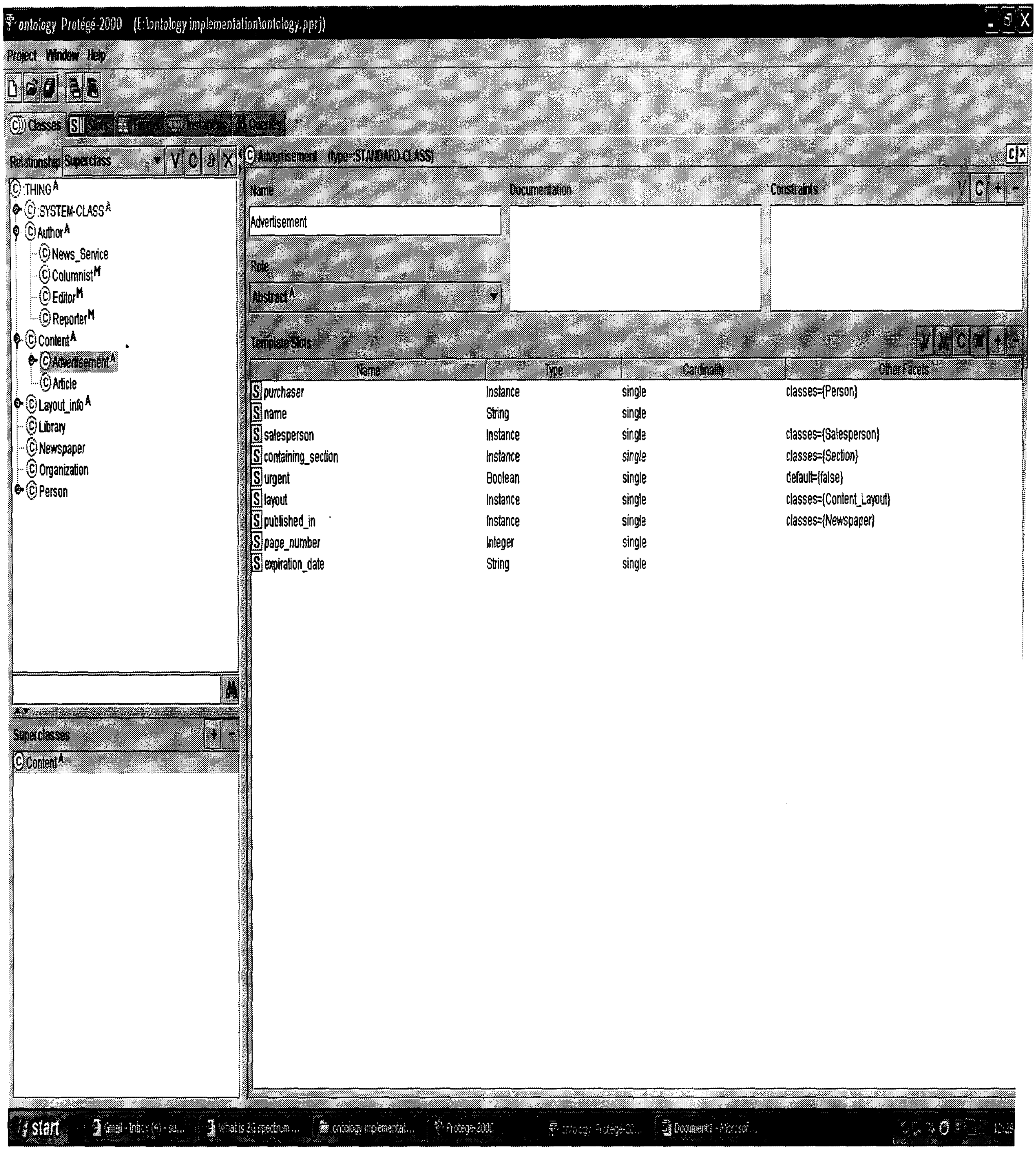
Constraints: editor-employees-salary-constraint

Template Slots

Name	Type	Cardinality	Other Facets
responsible_for	Instance	multiple	classes={Employee}
sections	Instance	multiple	classes={Section}
name	String	single	
salary	Float	single	
current_job_title	String	single	
date_hired	String	single	
phone_number	String	single	
other_information	String	single	


ontology Protégé-2000 (E:\ontology implementation\ontology.pprj)
Project Window Help
Classes Slots Forms Instances Queries
Forms
:THING
:SYSTEM-CLASS
Author
News_Service
Columnist M
Editor M C
Reporter M C
Content
Layout_info
Library
Newspaper
Organization C
Person
Form Browser Key
name
Selected Widget Type
Name
Salary
Date Hired
Responsible For
Current Job Title
Phone Number
Sections
Other Information
start


ontology Protégé-2000 (E:\ontology implementation\ontology.pprj)
Project Window Help
Classes
Slots
Forms
Instances
Queries
Forms
Form Browser Key
Selected Widget Type
S name
:THING
:SYSTEM-CLASS
Author
News_Service
Columnist M
Editor M C
Reporter M C
Content
Layout_info
Library
Newspaper
Organization C
Person
Name
Salary
Date hired
Other Information
Current Job Title
Phone Number
start



ontology Protégé-2000 (E:\ontology implementation\ontology.pprj)

C :THING A
C :SYSTEM-CLASS A
C Author A
C News_Service
C Columnist M
C Editor M
C Reporter M
C Content A
C Advertisement A
C Article
C Layout_info A
C Library
C Newspaper
C Organization
C Person

Superclasses: Content A

C Advertisement (type=:STANDARD-CLASS)

Name: Advertisement

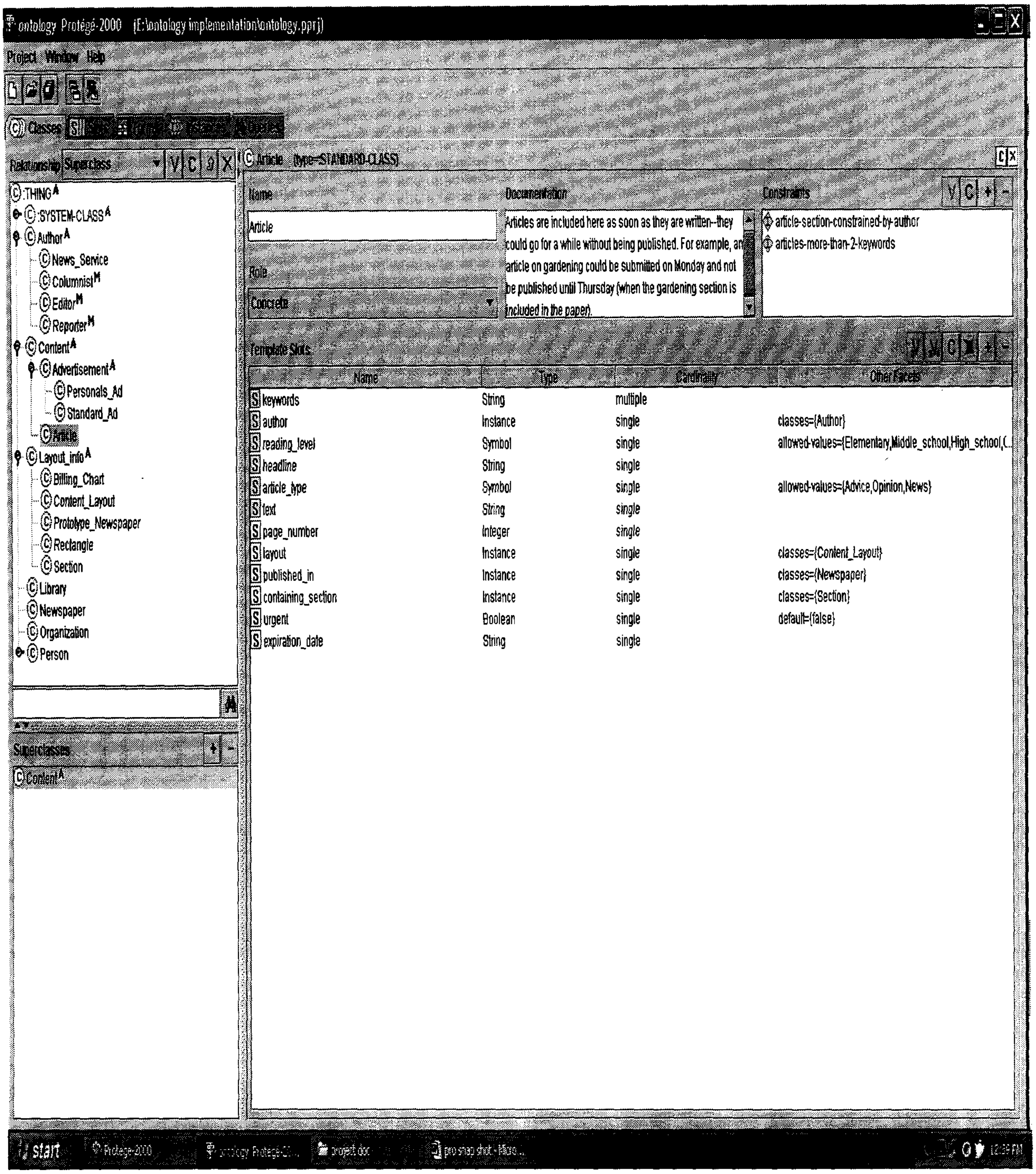
Role: Abstract A

Documentation

Constraints

Template Slots

Name	Type	Cardinality	Other Facets
purchaser	Instance	single	classes={Person}
name	String	single	
salesperson	Instance	single	classes={Salesperson}
containing_section	Instance	single	classes={Section}
urgent	Boolean	single	default={false}
layout	Instance	single	classes={Content_Layout}
published_in	Instance	single	classes={Newspaper}
page_number	Integer	single	
expiration_date	String	single	



:THING
- :SYSTEM-CLASS
- Author
  - News_Service
  - Columnist
  - Editor
  - Reporter
- Content
  - Advertisement
    - Personals_Ad
    - Standard_Ad
  - Article
- Layout_info
  - Billing_Chart
  - Content_Layout
  - Prototype_Newspaper
  - Rectangle
  - Section
- Library
- Newspaper
- Organization
- Person

Article (type=STANDARD-CLASS)

Name: Article

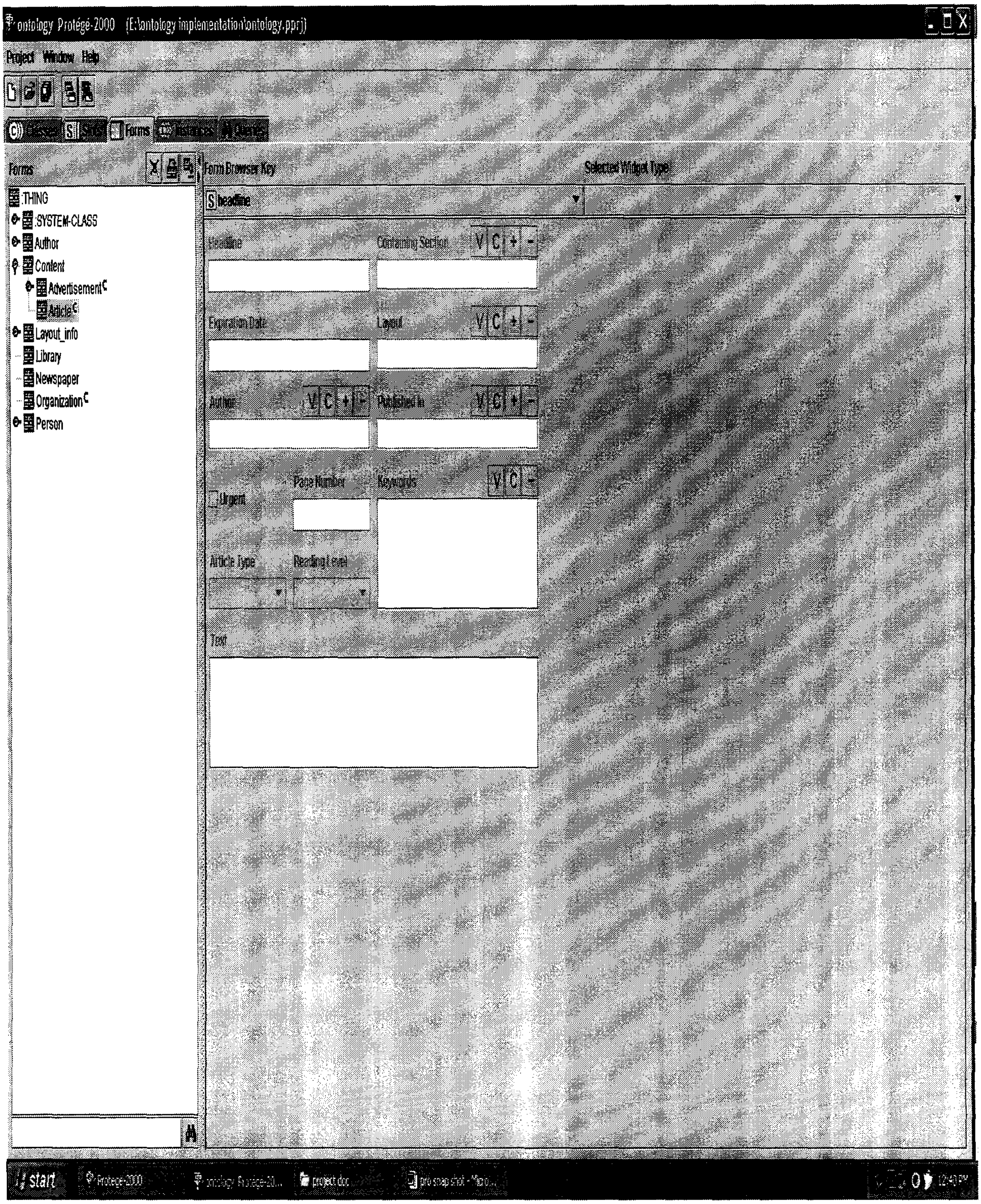
Role: Concrete

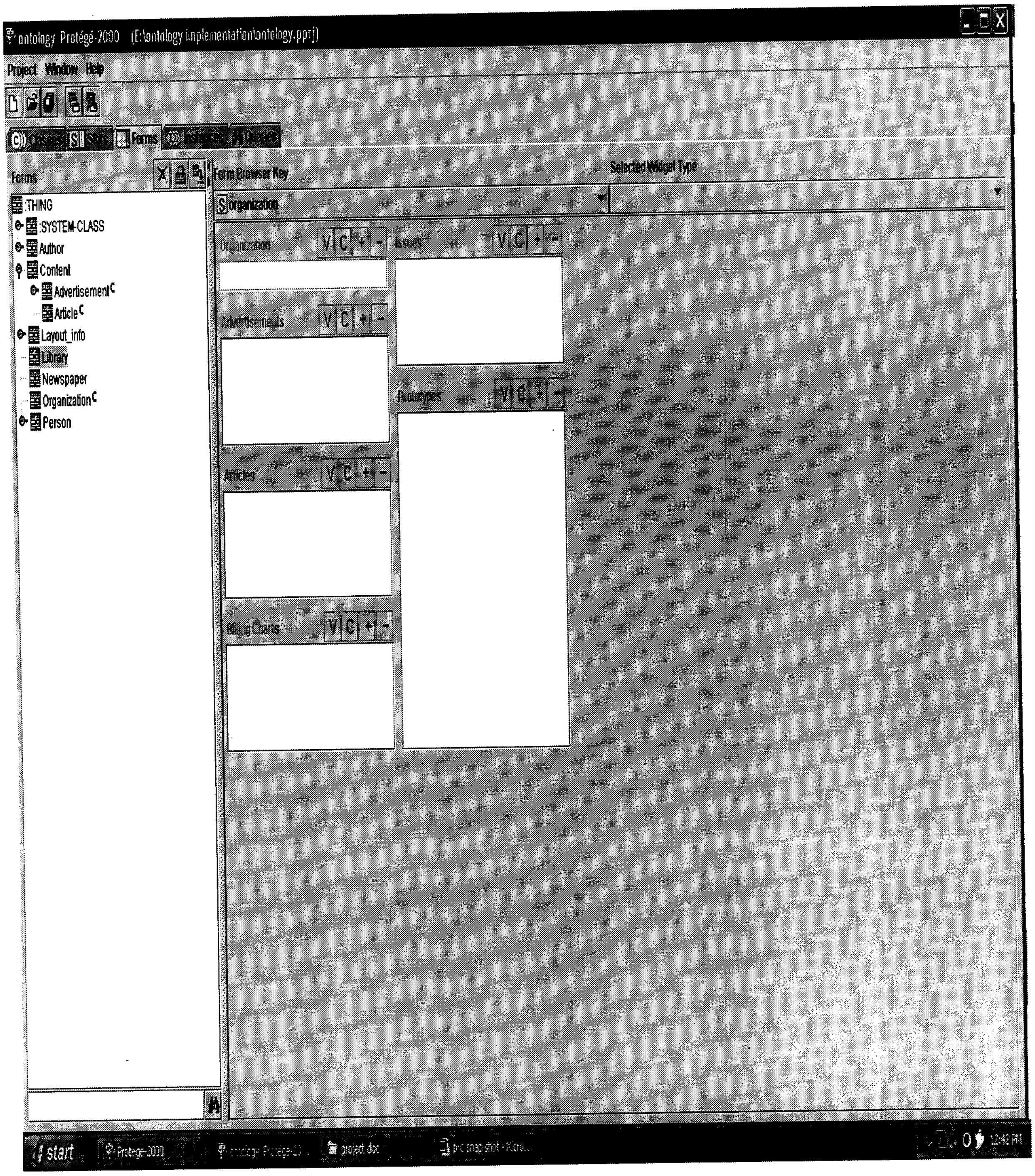
Documentation: Articles are included here as soon as they are written--they could go for a while without being published. For example, an article on gardening could be submitted on Monday and not be published until Thursday (when the gardening section is included in the paper).

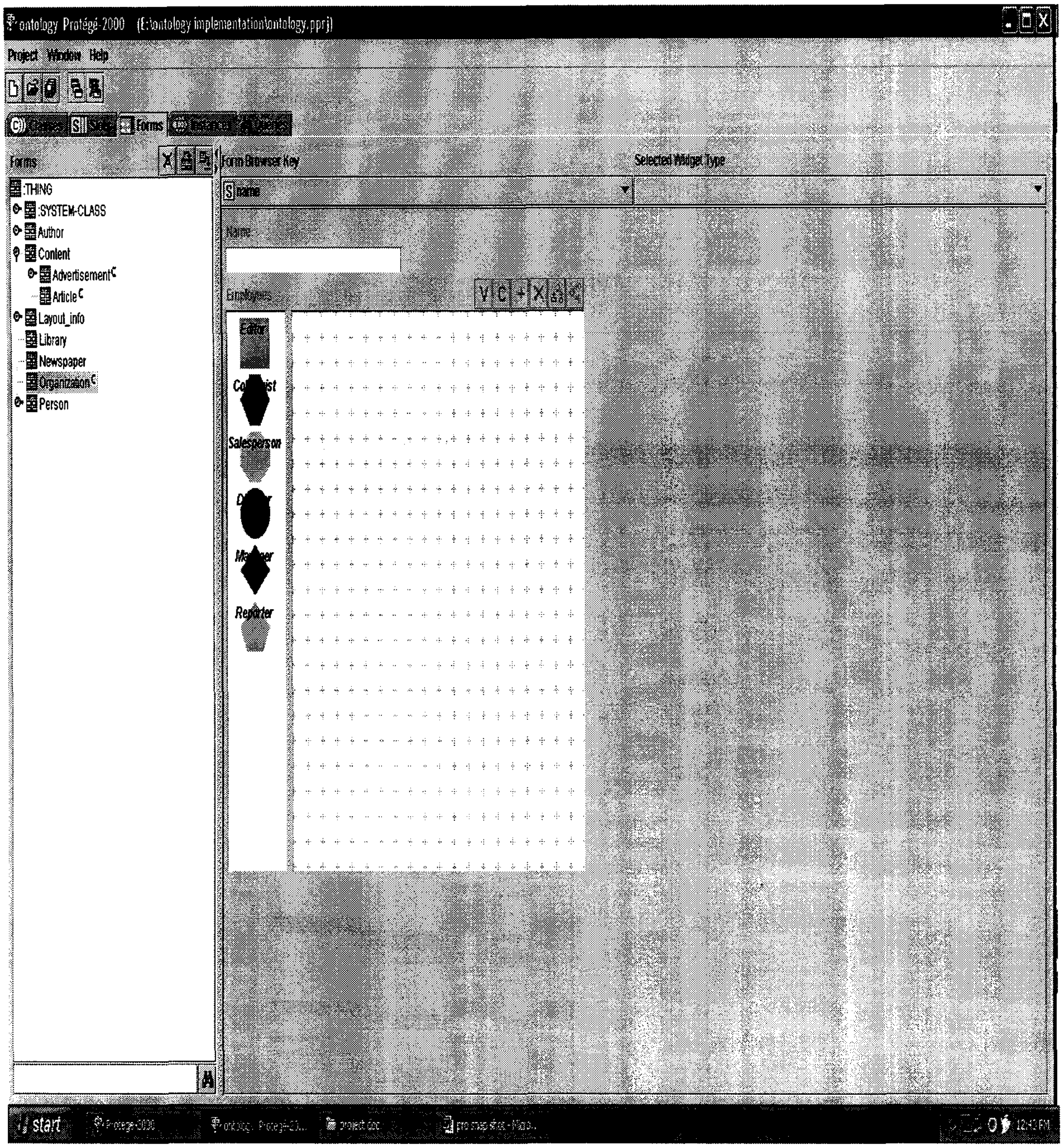
Constraints:
- article-section-constrained-by-author
- articles-more-than-2-keywords

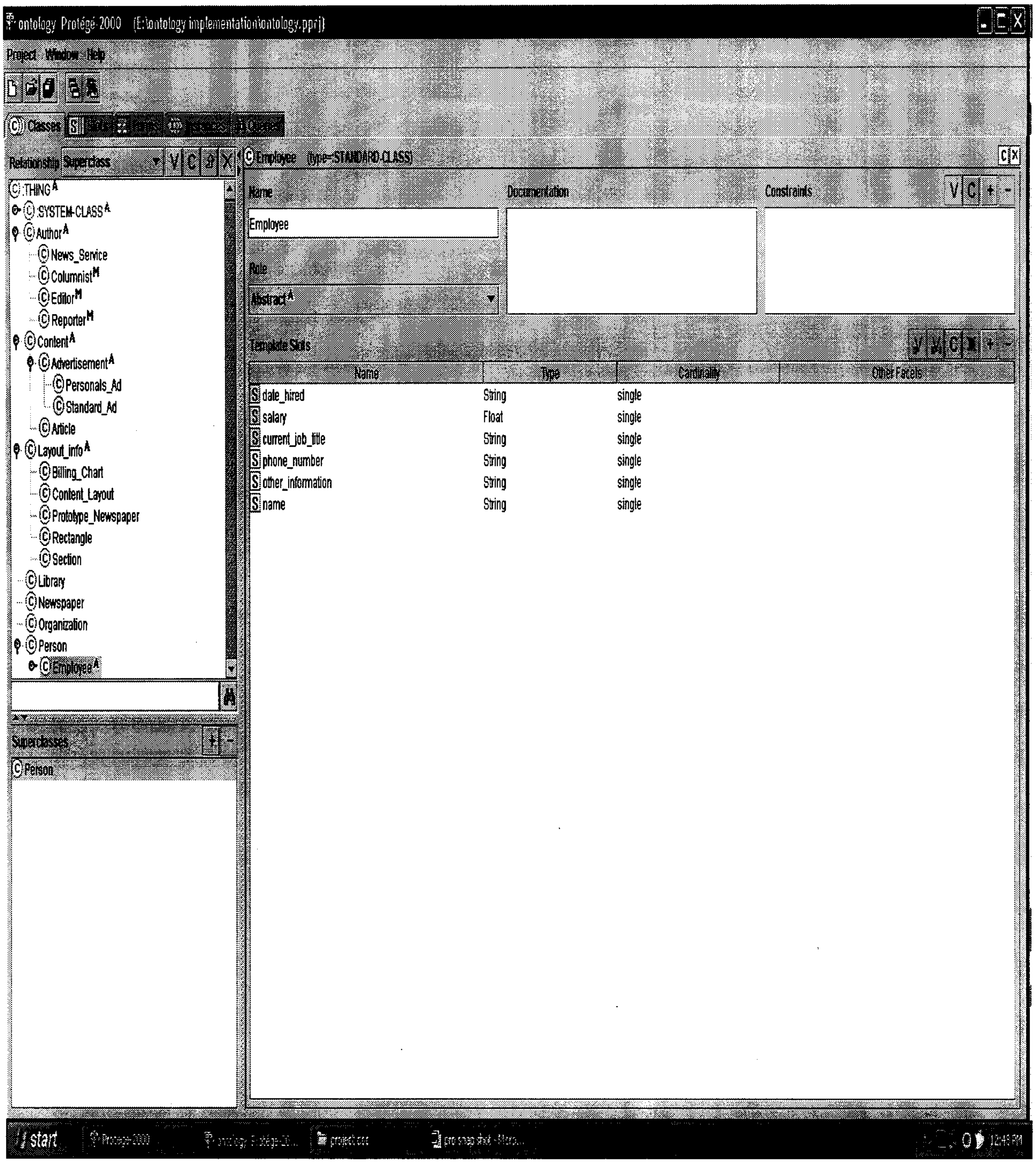
Template Slots

Name	Type	Cardinality	Other Facets
keywords	String	multiple	
author	Instance	single	classes={Author}
reading_level	Symbol	single	allowed-values={Elementary,Middle_school,High_school,(...
headline	String	single	
article_type	Symbol	single	allowed-values={Advice,Opinion,News}
text	String	single	
page_number	Integer	single	
layout	Instance	single	classes={Content_Layout}
published_in	Instance	single	classes={Newspaper}
containing_section	Instance	single	classes={Section}
urgent	Boolean	single	default={false}
expiration_date	String	single	


ontology Protégé-2000 (E:\ontology implementation\ontology.pprj)
Project Window Help
Classes
Slots
Forms
Instances
Queries
Forms
Form Browser Key
Selected Widget Type
headline
:THING
:SYSTEM-CLASS
Author
Content
Advertisement
Article
Layout_info
Library
Newspaper
Organization
Person
Headline
Containing Section
Expiration Date
Layout
Author
Published In
Urgent
Page Number
Keywords
Article Type
Reading Level
Text
start
Protégé-2000
ontology Protégé-20...
project.doc
pro snap shot - Micro...


ontology Protégé-2000 (E:\ontology implementation\ontology.pprj)
Project Window Help
Classes
Slots
Forms
Instances
Queries
Forms
Form Browser Key
Selected Widget Type
organization
:THING
:SYSTEM-CLASS
Author
Content
Advertisement
Article
Layout_info
Library
Newspaper
Organization
Person
Organization
Issues
Advertisements
Prototypes
Articles
Billing Charts
start
Protégé-2000
project.doc


ontology Protégé-2000 (E:\ontology implementation\ontology.pprj)
Project Window Help
Classes
Slots
Forms
Instances
Queries
Forms
Form Browser Key
Selected Widget Type
:THING
:SYSTEM-CLASS
Author
Content
Advertisement
Article
Layout_info
Library
Newspaper
Organization
Person
name
Name
Employees
Editor
Columnist
Salesperson
Director
Manager
Reporter
start
Protege-2000
ontology Protégé-20...
project.doc
12:43 PM



- THING
  - :SYSTEM-CLASS
  - Author
    - News_Service
    - Columnist
    - Editor
    - Reporter
  - Content
    - Advertisement
      - Personals_Ad
      - Standard_Ad
    - Article
  - Layout_info
    - Billing_Chart
    - Content_Layout
    - Prototype_Newspaper
    - Rectangle
    - Section
  - Library
  - Newspaper
  - Organization
  - Person
    - Employee

Superclasses: Person

Employee (type=:STANDARD-CLASS)

Name: Employee

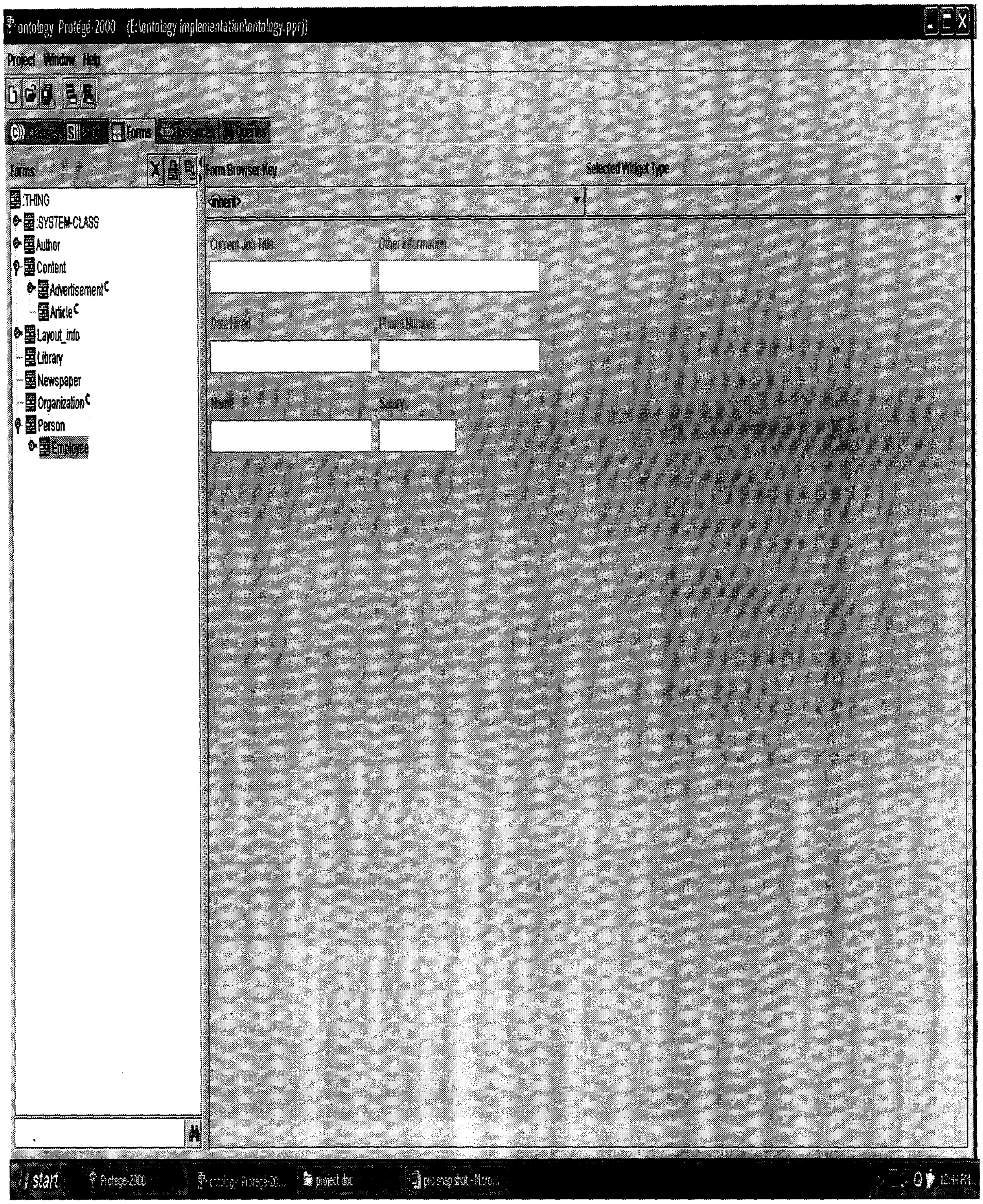
Role: Abstract

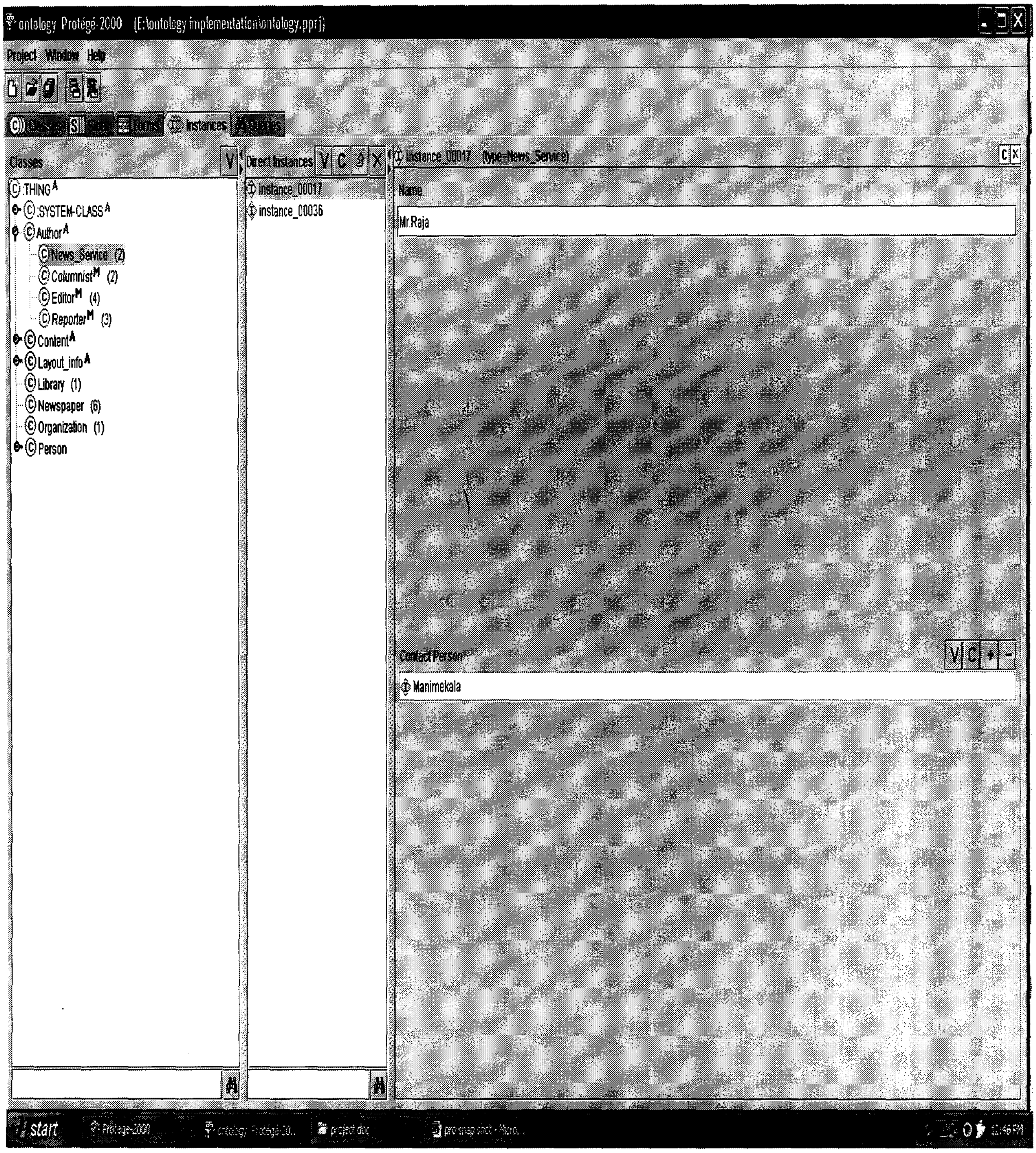
Documentation

Constraints

Template Slots

Name	Type	Cardinality	Other Facets
date_hired	String	single	
salary	Float	single	
current_job_title	String	single	
phone_number	String	single	
other_information	String	single	
name	String	single	


ontology Protégé-2000 (E:\ontology implementation\ontology.pprj)
Project Window Help
Classes
Slots
Forms
Instances
Queries
Forms
Form Browser Key
Selected Widget Type
<inherit>
:THING
:SYSTEM-CLASS
Author
Content
Advertisement
Article
Layout_info
Library
Newspaper
Organization
Person
Employee
Current Job Title
Other information
Date Hired
Phone Number
Name
Salary
start
Protege-2000
ontology Protégé-20...
project.doc
pro snap shot



Classes

- :THING
- :SYSTEM-CLASS
- Author
  - News_Service (2)
  - Columnist (2)
  - Editor (4)
  - Reporter (3)
- Content
- Layout_info
- Library (1)
- Newspaper (6)
- Organization (1)
- Person

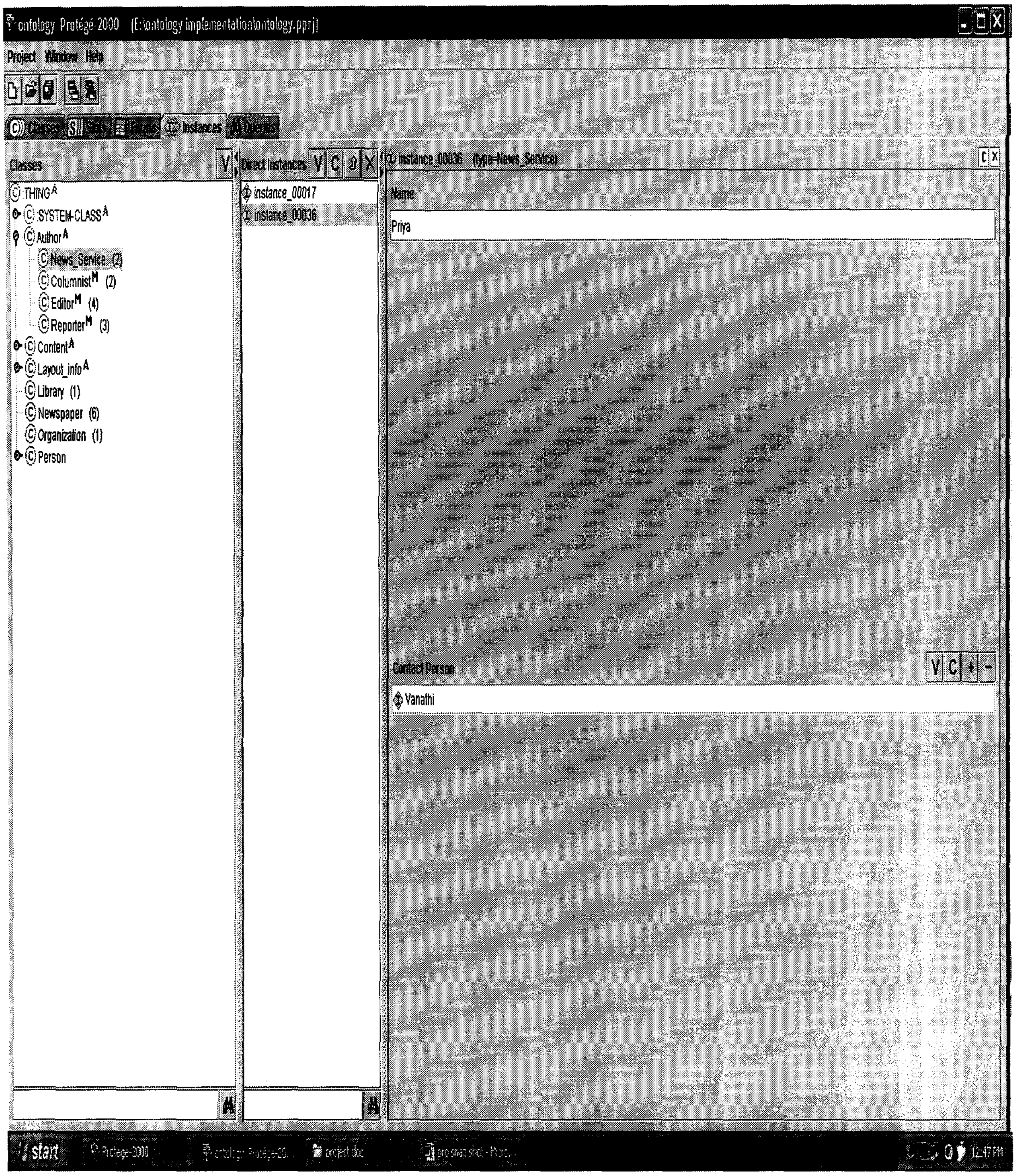
Direct Instances

- instance_00017
- instance_00036

instance_00017 (type=News_Service)

Name: Mr.Raja

Contact Person: Manimekala



ontology Protégé-2000 (E:\ontology implementation\ontology.pprj)

Classes

- THING^A
  - SYSTEM-CLASS^A
  - Author^A
    - News_Service (2)
    - Columnist^M (2)
    - Editor^M (4)
    - Reporter^M (3)
  - Content^A
  - Layout_info^A
  - Library (1)
  - Newspaper (6)
  - Organization (1)
  - Person

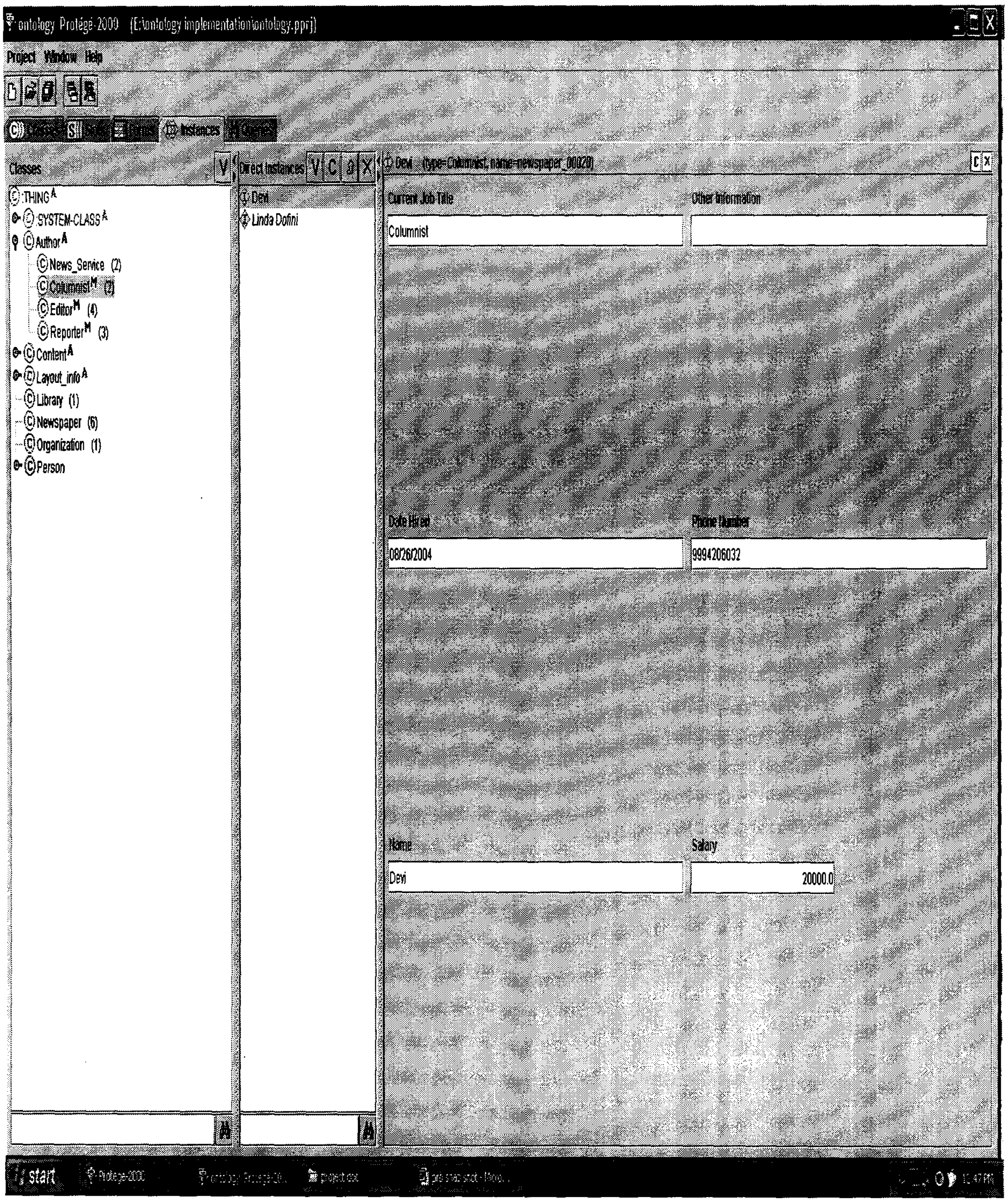
Direct Instances

- instance_00017
- instance_00036

instance_00036 (type=News_Service)

Name: Priya

Contact Person: Vanathi

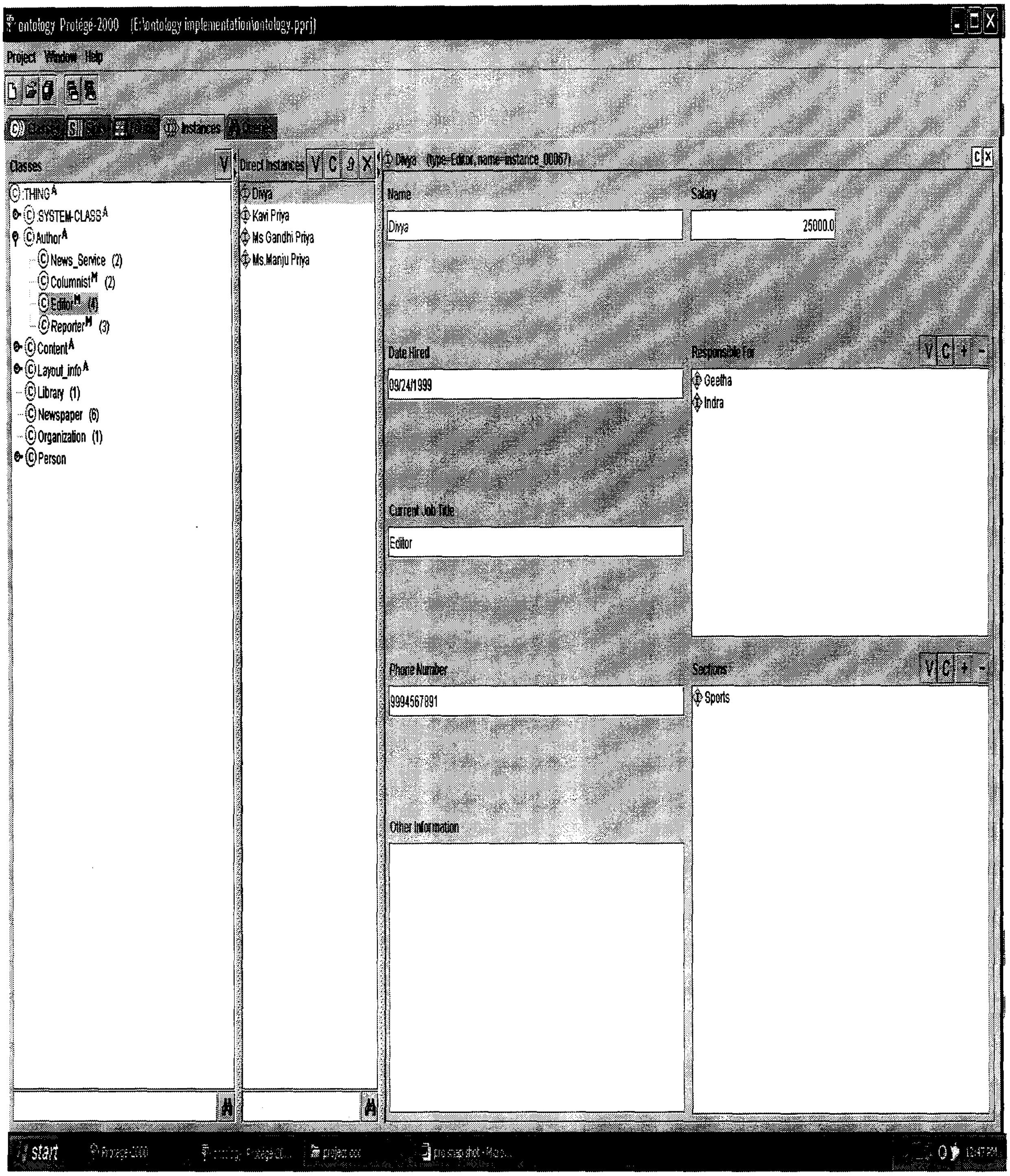


:THING
:SYSTEM-CLASS
Author
News_Service (2)
Columnist (2)
Editor (4)
Reporter (3)
Content
Layout_info
Library (1)
Newspaper (6)
Organization (1)
Person

Devi
Linda Dofini

Devi (type=Columnist, name=newspaper_00020)

Current Job Title	Other Information
Columnist	
Date Hired	Phone Number
08/26/2004	9994206032
Name	Salary
Devi	20000.0


ontology Protégé-2000 (E:\ontology implementation\ontology.pprj)
Project Window Help
Classes Slots Forms Instances Queries
start Protege-2000 ontology Protégé-20... project.ooc ...snapshot - Micro...

Classes

- :THING
  - :SYSTEM-CLASS
  - Author
    - News_Service (2)
    - Columnist (2)
    - Editor (4)
    - Reporter (3)
  - Content
  - Layout_info
  - Library (1)
  - Newspaper (6)
  - Organization (1)
  - Person

Direct Instances

- Divya
- Kavi Priya
- Ms Gandhi Priya
- Ms.Manju Priya

Divya (type=Editor, name=instance_00067)

Name: Divya

Salary: 25000.0

Date Hired: 09/24/1999

Responsible For: Geetha, Indra

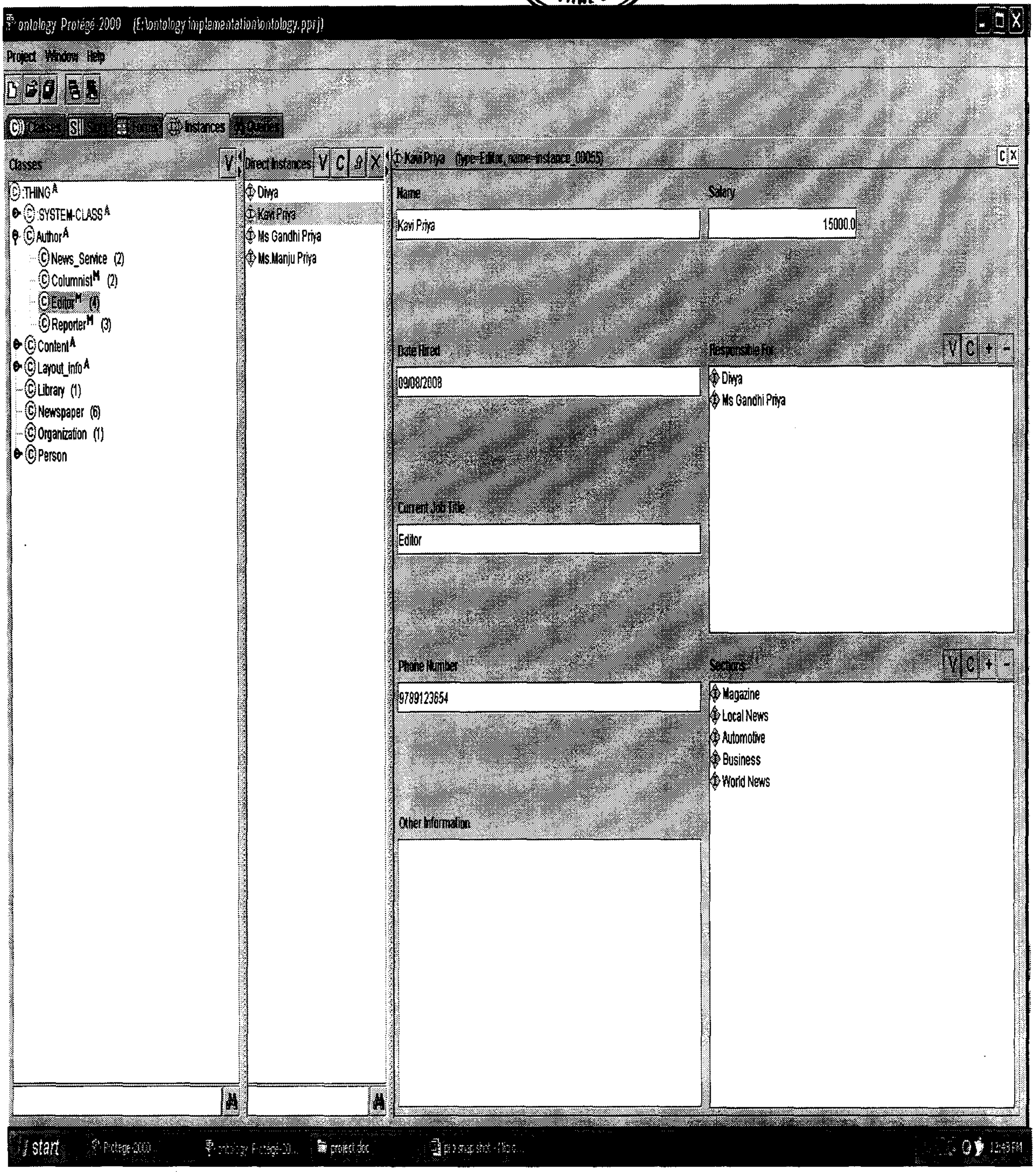
Current Job Title: Editor

Phone Number: 9994567891

Sections: Sports

Other Information:

P-3619



ontology Protégé 2000 (E:\ontology implementation\ontology.pprj)

Project Window Help

Classes | Slots | Forms | Instances | Queries

Classes
- :THING
  - :SYSTEM-CLASS
  - Author
    - News_Service (2)
    - Columnist (2)
    - Editor (4)
    - Reporter (3)
  - Content
  - Layout_info
  - Library (1)
  - Newspaper (6)
  - Organization (1)
  - Person

Direct Instances
- Diwya
- Kavi Priya
- Ms Gandhi Priya
- Ms.Manju Priya

Kavi Priya (type=Editor, name=instance_00055)

Name: Kavi Priya

Salary: 15000.0

Date Hired: 09/08/2008

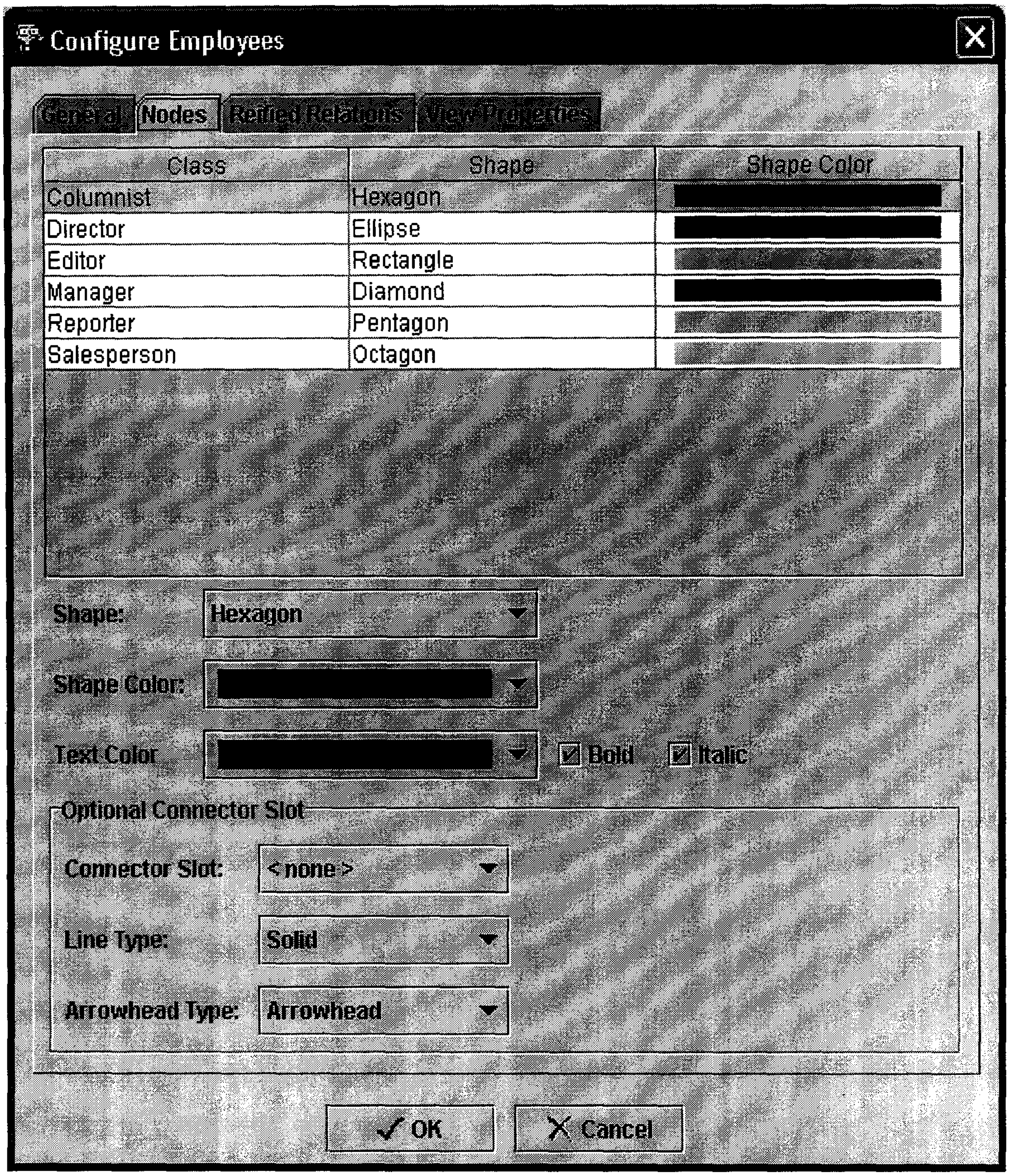
Responsible For:
- Diwya
- Ms Gandhi Priya

Current Job Title: Editor

Phone Number: 9789123654

Sections:
- Magazine
- Local News
- Automotive
- Business
- World News

Other Information:



Class	Shape	Shape Color
Columnist	Hexagon	■
Director	Ellipse	■
Editor	Rectangle	▨
Manager	Diamond	■
Reporter	Pentagon	▨
Salesperson	Octagon	▨

Shape: Hexagon

Shape Color: ■

Text Color: ■ [✓ Bold] [✓ Italic]

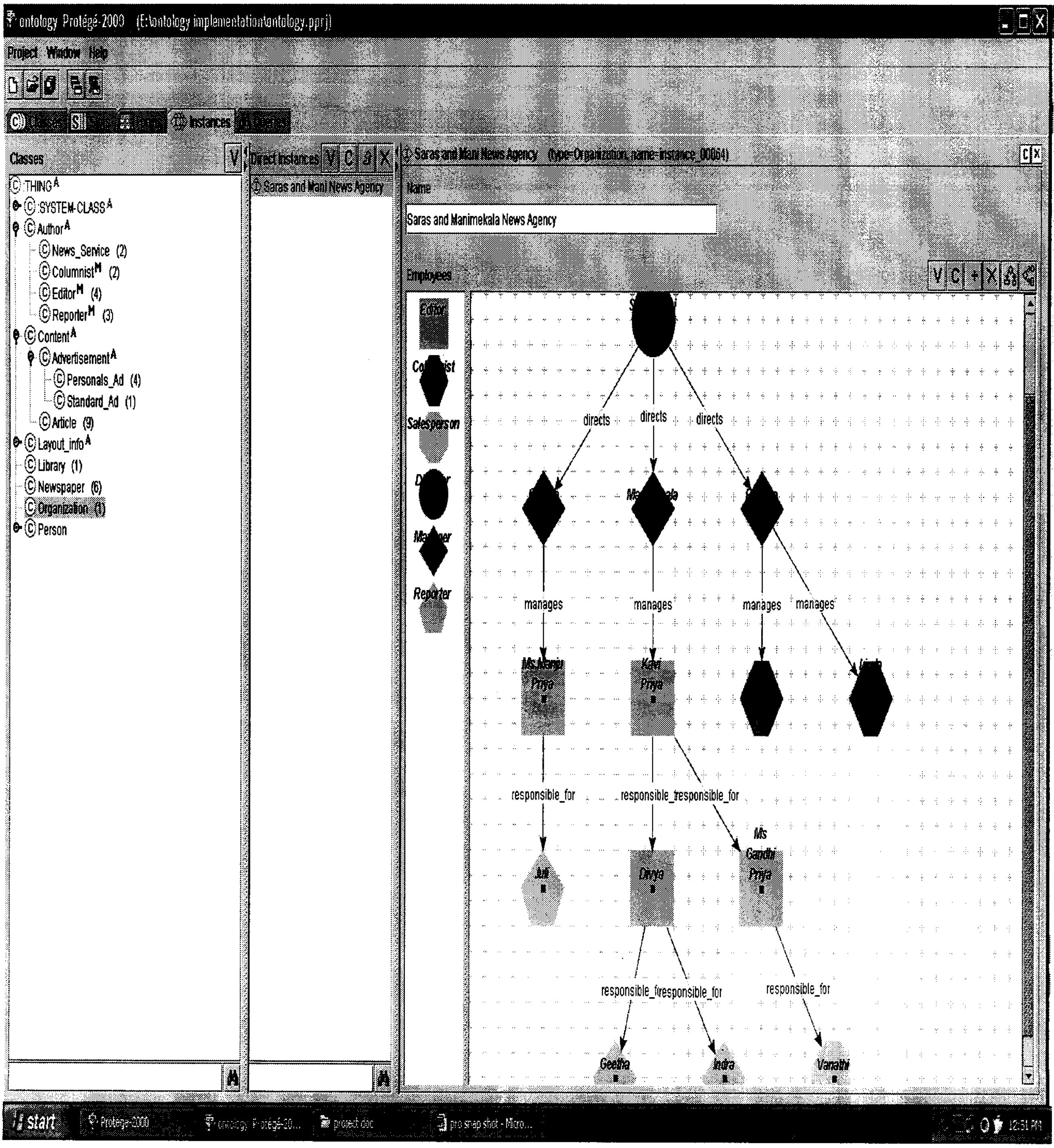
Optional Connector Slot

Connector Slot: < none >

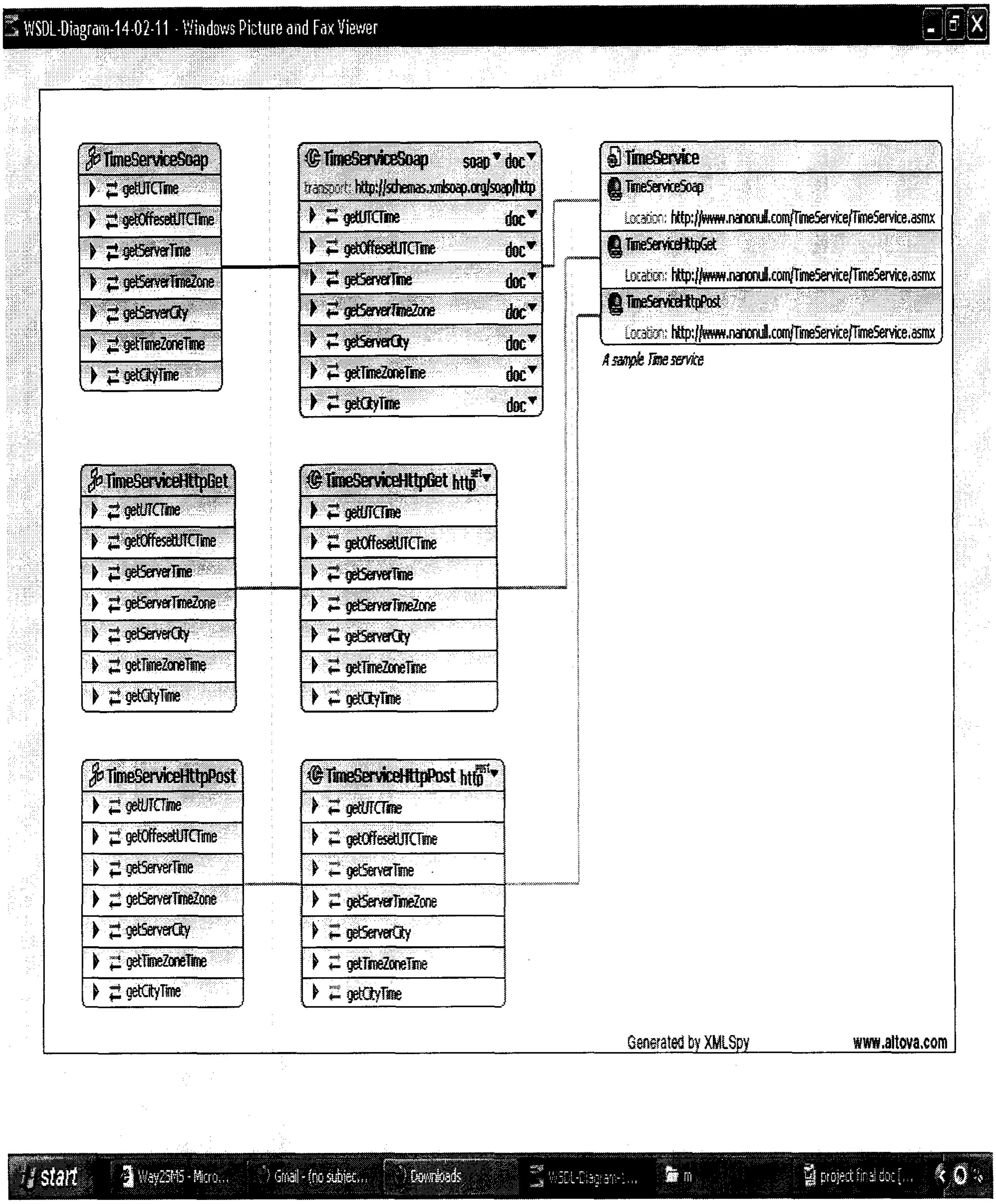
Line Type: Solid

Arrowhead Type: Arrowhead

Ontology Tree Structure



WSDL DIAGRAM



REFERENCES

1.Berners-Lee. T., Hendler J. and Lassila O., "The semantic Web. Scientific American", 2001, 284(5):34-43.

2. Buhler P. , Vidal J. M., " Semantic web services as agent behaviours", Agentcities: Challenges in Open Agent Environments, pages 25-31. Springer-Verlag, 2003.

3.Burstein M., Bussler C., Zaremba M., Finin T., Huhns M. N., Paolucci M., Sheth A. P., and Williams S., "A Semantic Web Services Architecture," IEEE INTERNET COMPUTING, vol. 9, pp. 72-81, 2005.

4. Choi, O., Han, S., Abraham, A.: Semantic Matchmaking Services Model for the intelligent Web Services. In: International Conference on Computational Science and Applications, pp. 146–148. IEE Press,UK ,2006.