A Semantic Web search tool for Scientific Grants
Presentation outline

Design and Implementation
Testing
Demonstration
Design

The research grant software tool uses a variety of software utilities to retrieve web data that is relevant to the researcher's profile. These tools may include:

- Informa (RSS news reader), etc.
- Lucene (Used to search the data for sections with relevancy.)
- Jena (Used to track ontology database information)
- Java Swing (Java User Interface).
- Protege (Ontology creator and editor.)
Design and Implementation

(For applications using Java Swing)
The constructor for the GUI object is called.

```java
private static AcademicSemanticWebTerm coreObject = null;
private static AcademicSemanticGUI singleton = null;

public static Mutex GUIMutex = new Mutex();

public AcademicSemanticGUI() {
    /*
     * I placed a mutex on this object because the interface
     * needs to finish loading before the worker thread can update the interface. The interface is updated
     * by the worker thread after the database has been downloaded from the web.
     */
    GUIMutex.lock();
    System.out.print("Running constructor.\n");
    System.out.print("Executing main.\n");
    init();
    DrawItems();
    singleton = this;
    GUIMutex.unlock();
}

public static AcademicSemanticGUI getInstance()
```

// Lock is acquired
Informa is dispatch on a separate Thread.
// Lock is dropped.
The RSSNewsFeeder function is executed in a separate thread.

The graphical widgets are displayed.

After the function has completed its task, the GUI object is notified and the rest of the GUI object is notified.

// Lock is acquired
// Lock is dropped.
Design and Implementation cont.

Code below: Shows how the GUI is notified that the database has completed loading.

```java
public class AcademicSemanticWebTermWorker extends Thread {

    AcademicSemanticWebTermWorker()
    {
        start();
    }

    public void run()
    {
        System.out.println("Entering run thread method.\n");
        AcademicSemanticWebTerm loader = new AcademicSemanticWebTerm();
        loader.RSSNewsFeeder();
        //loader.WebSphinxCrawler();
        System.out.println("Exiting run thread method.\n");
        AcademicSemanticGUI.GUIMutex.lock();
        AcademicSemanticGUI.getInstance().LoadingFinished();
        AcademicSemanticGUI.GUIMutex.unlock();
    }
}
```
Design and Implementation cont.

Flow chart that depicts the loading phase of the program.

WorkerThread:
Informa is retrieving and storing RSS news data in a database.
Database is loading

Graphical object's constructor
Has been called.

Ontology is loaded into the search tool using Jena.

Java Swing graphical widgets are displayed.

User creates a ontological profile while waiting for Informa to complete its task.

Loading has been completed, the user can search the database.
Design and Implementation cont.

Code below: Shows the ontological classes that the JenaResearchProfileOntologyLoader looks for when populating items in the Graphical interface with vocabulary from the ontology.

```java
public String[][] JenaResearchProfileOntologyLoader()
{
    String[] CategoryObjects = { "ResearchTopic",
        "RequiredFunding",
        "RequiredFacilities",
        "GrantSponsor",
        "ProposalDueDate"
    };
    String[] CategoryProperties = { "researchTopic",
        "fundingRequired",
        "facilitiesProvided",
        "organizationSponsor",
        "earliestProposalDueDate"
    };

    namespace = "http://www.semanticweb.org/ontologies/2011/9/Ontology1319588817251.owl#";

    String[][] categories = new String[5][1];
    int []index = { 1, 1, 1, 1, 1}; // keep track of the number of strings in each category.
    String[] temp = null;

    // Read the protege owl file.
    System.out.print("Enter Jena research ontology profile loader.\n");
    researchProfileModel = ModelFactory.createOntologyModel();
    InputStream inFoafInstance = null;
```
Design and Implementation cont.

Code below: Show the Java Swing api calls used to place the GUI items inside the Java applet window.

```java
public void DrawItems()
{
    String[] PropertyLabels = { "Research concentration",
                                "Required Funding",
                                "Facilities Provided",
                                "Funding Sponsor",
                                "Earliest due date of proposal"
    };

    System.out.print("Executing Draw Items.\n");

    JPanel MainTop = new JPanel();
    MainTop.setBounds(0, 0, 1500, 1000);
    MainTop.setLayout(new GridLayout(8, 1, 0, 2));
    MainTop.setBackground( new Color(0x00000000) );

    // Title
    JLabel Title = new JLabel("Science Research Grants Search Tool", JLabel.CENTER);
    Title.setSize( Title.getPreferredSize() );
    Title.setFont(new Font("Serif", Font.PLAIN, 36));
    Title.setForeground(new Color(0x00ffffff));
    MainTop.add(Title);
    JLabel Blank = new JLabel(""");
    MainTop.add(Blank);
```
Design and Implementation cont.

Flow chart the show the post loading phase of the program.

Loading has been completed, the user can search the database.

When the search button is clicked, all the ontological statements created by the user are loaded into Jena. (Button event handler is shown the code below)

```java
public void actionPerformed(ActionEvent e) {
    System.out.printf("Button pressed. \n", e.getActionCommand().toString());

    String[] resultsArray = null;
    String displayString = new String();

    if (!SearchResultsLabel.isVisible())
    {
    }
    else
    {
        // When performing the search, use the research ontology defined by the user.
        if (e.getActionCommand().contains("research profile"))
        {
            String itemString = null;
            String searchQuery = null;
            listItems1.clear();
            coreObject.clearAllOntologyProfileStatements();

            //coreObject.listStatementsInTheOntologyModel();

            for (int index2 = 0; index2 < OntologyInstanceSelection.length; index2++)
        
```
A **Lucene query string** is created based on the ontological statements created by the user. Code below constructs a query string.

```java
public String doSearchOnResearcherProfile()
{
    System.out.println("Entering doSearchOnResearcherProfile\n");

    Resource researcher = researchProfileModel.getResource(namespace + "ResearchScientist");
    StmtIterator iter = researcher.listProperties();

    StringBuffer searchStringForLucene = new StringBuffer(""");
    String retval = null;

    String subject = null;
    String predicate = null;
    String object = null;
    int numberOfFields = 0;

    String searchTemp[] = new String[5];

    while (iter.hasNext() && (numberOfFields < 5))
    {
        Statement statement = iter.nextStatement();
        //System.out.printf("Object %s\n", statement.toString());
        subject = statement.getSubject().getURI().toString().substring(statement.getSubject().getURI().toString().indexOf("/"));
        predicate = statement.getPredicate().getURI().toString().substring(statement.getPredicate().getURI().toString().indexOf("/"));
        object = statement.getObject().toString().substring(statement.getObject().toString().indexOf("/"));

        if (predicate.equals("fundingRequired".toString()))
        {
            if (object.equals("Any_Amount".toString()))
            {
```
The **Lucene query string** is passed into Lucene using Lucene APIs.

```java
public String[] LuceneSearch(String querystr)
{
    String[] retArray = null;

    System.out.println("Enter LuceneIndexer.\n");

    try {
        iwritter.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
    }

    try {
        System.out.printf("Total of %d documents in the index writer.\n", iwritter.numDocs());
        } catch (IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
    }

    // 2. query
    String querystr = "Funding";

    // the "title" arg specifies the default field to use
    // when no field is explicitly specified in the query.
    MultiFieldQueryParser q = null;
} /*
```
The search results are returned in a string array which is sent to the Java Swing GUI where the results are displayed. (The code below shows how the results are displayed on the GUI.)

```java
// Search the database
searchQuery = coreObject.doSearchOnResearcherProfile();
//searchQuery = "Funding AND $150,000";
// Pass the searchQuery into Lucene.
// Start search
resultsArray = coreObject.LuceneSearch(searchQuery);
System.out.println("SearchQuery = %s\n", searchQuery);
// Output search results on the Interface.

for (int outputIndex = 0; outputIndex < resultsArray.length; outputIndex++)
{
    //listItems1.addElement(resultsArray[outputIndex].toString());
    displayString = displayString + (resultsArray[outputIndex] + "\n");
}

if (resultsArray.length == 0)
{
    displayString = "No pages found."
}

System.out.println("String = %s\n", displayString);
resultsPanel.setText(displayString);
```

// End search, Output results.
//coreObject.listStatementsInTheOntologyModel();
// When performing a search, using the search string provided by the user.
else if (e.getActionCommand().contains("search query"))
{    String results[] = null;
```
Testing

The testing phase consists of multiple parts:

1. Verify that the ontological data in the user interface corresponds to the data displayed on the interface.

2. Check if the appropriated search string is generated based on the ontological statements. (Debug is shown on later slides)

3. Check the results of the search.
Testing

1. Verify that the ontological data in the user interface corresponds to the data displayed on the interface.
Testing

2. Check if the appropriated search string is generated based on the ontological statements. (Debug is shown below)
3. Check the results of the search.
NSF 12-018

Dear Colleague Letter: **Computational** and Data-Enabled Science and Engineering in Mathematical and Statistical Sciences (CDS&E-MSS)

Dear Colleagues:

The conduct of scientific research is being revolutionized by developments in the availability of computational resources and digital datasets. The last three decades have seen advances of roughly nine orders of magnitude in computing capability, together with deep advances in computational algorithms. These advances allow computational and multiscale simulations of unprecedented scope and accuracy. Simultaneous advances in digital data collection technology are proceeding at an even faster pace, with the result that enormous datasets are now generated routinely by scientific experiments and observations of natural phenomena. The result is a scientific revolution in the scope, use, and production of data.

As did digital computation itself, such data-intensive science is driving revolutionary advances in mathematics and statistics. How are features, let alone new laws of nature, to be found in the vast volumes of data being generated and collected? How can disparate data, from simulations and observations, different instruments and multiple communities, be combined to advance knowledge?

The Division of Mathematical Sciences and the Office of Cyberinfrastructure of the National Science Foundation recognize the importance of fundamental mathematical and statistical research in this field of computational and data-enabled science and engineering (CDS&E) and envision that the mathematical and statistical research communities will play a leading role in the future development of this emerging science. In partnership with the Office of Cyberinfrastructure, the CDS&E-MSS program in DMS supports fundamental mathematical and statistical research at the core of this emerging discipline. The goal of the program is to promote the creation, development, and application of the next generation of mathematical and statistical theories and tools that will be essential for addressing the challenges presented to the scientific and engineering communities by the ever-expanding role of computational modeling and simulation on the one hand, and the explosion in production of digital and observational data on the other. To this end, the program will support fundamental research in mathematics and statistics, including transition to practice, whose primary emphasis will be on meeting the aforementioned computational and data-related challenges. The program has strong interest in multidisciplinary collaboration and the training of next-generation mathematicians and statisticians firmly grounded in CDS&E.

Examples in which mathematical and statistical research enables advances in CDS&E include, but are not limited to:

- Sophisticated computational/statistical modeling for simulation, prediction, and assessment in computation-intensive and data-intensive scientific problems.
- State-of-the-art tools and theory in statistical inference and statistical learning for knowledge discovery from massive, complex, and dynamic data sets.
- General theory and algorithms for advancing large-scale modeling of problems that present particular computational difficulties, such as strong heterogeneities and anisotropies, multi-physics coupling, multiscale behavior, stochastic forcing, uncertain parameters or dynamic data, and long-time behavior.
Demonstration

In class or through a video.

Questions?

Farther information can be found at the following blogs:

- http://ebsemanticwebproject.wordpress.com/
- http://ebsemanticwebcourse.wordpress.com/