

USC/ISI DAML Homework 2

DARPA/ISO DAML Project
2000-12-15

Martin Frank (with Pedro Szekely and Bob Neches)

<http://www.isi.edu/webscripiter>

USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292 USA

1 DAML sources used

Start with the collected results of homework assignment 1, augmented, if desired, by hypothetical DAML repositories of your own choosing.

The running example in this paper uses two ontologies - the USC/ISI ontologies of Software Project To-Do Items and of Milestones (<http://www.isi.edu/webscripiter/todo.o.daml> and <http://www.isi.edu/webscripiter/milestone.o.daml>)

It uses three DAML data sources: (1) to-do items at <http://www.isi.edu/webscripiter/todo.gen.d.daml> (which are generated from an XML database at <http://www.isi.edu/~frank/webscripiter/mrfstatus.xml>), (2) milestones at <http://www.isi.edu/webscripiter/mrfmilestones.d.daml>, and (3) an annotation file that relates to-do items to milestones at <http://www.isi.edu/webscripiter/mrftodoaddenda.d.daml>. We intentionally produced the former two independent of each other and then related them via a third file to make sure that WebScripiter can handle that situation.

(We have also produced WebScripiter reports based on DAML data produced by SRI and Stanford for the first homework, and based on the other seven USC/ISI ontologies but will not refer to them in this paper.)

2 Challenge Queries

Propose 5 queries of roughly increasing complexity that might be performed on any or all these data sets. Queries can be expressed in English, predicate calculus, and/or other suitable representations.

1. Give me the entry dates, names, and descriptions of all open milestones for a given project, ordered by entry date, plus the names and entry-dates of the open to-do items that relate to them.

This is just about the most ambitious type of query that WebScripiter can currently already handle (you can view the result at <http://www.isi.edu/~frank/snapshots/20001215webscripiterreportforsnap.html>). One complication was that it involves having multiple to-do items in a single table cell (WebScripiter tables may not be in "first normal form" in "relational database speak").

2. For all the projects that I am involved in, give me (a) what my colleagues are working on right now, (b) the list of open to-do items per project, (c) a history of accomplishments per project.

A close cousin of that output can be seen at <http://www.isi.edu/~frank/webscripiter/todo.html> - but that was produced by a highly complex XSL script from XML database files, and cannot currently be produced by WebScripiter. From a query execution standpoint, the main problem is that the desired output is not a single table but rather a *collection* of *nested* tables; however, we will address queries of that complexity during the course of the project.

3. Alert me whenever someone creates or modifies a to-do item that lists myself as a performer, and whenever someone finishes a to-do item that lists myself as a requester.

WebScripter runs as a batch process that is initiated by hand. We will have to create a batch-running and a "diffing" capability against archived previous state to support this functionality.

4. Give me all of the personnel associated with the DAML effort (based on an exact list of concepts and attributes to look for in the various ontologies).

We currently do not support mixing data from multiple ontologies in the same column. The first solution to the above query is that the user explicitly lists the desired contents of a report (every stanford:Student and stanford:Faculty and isi:Employee and ... should become a row in the report, in the first column I want the stanford:lastName or isi:familyName or ...). That is, WebScripter will produce nice-looking reports from heterogeneous data sources - but the burden is on the user to specify exactly (by String match) what does and does not go into the report.

5. Refresh a list of people (with their phone numbers, email addresses, and home pages) associated with Semantic Web projects and highlight those that are new and those that have revised their software.

Requires ontology triangulation on the fly to support cleanly. WebScripter would be given a statement such as "look for a concept called person or staff with a last name or family name ..., and for concepts that have a similar structure", with the degree of required similarity defined by the user.

3 Query Execution Architecture

Describe how you would expect these queries to be implemented. Identify the major DAML software components and sketch (in text, HTML, PowerPoint, GIF, or JPEG) the control and data flow among them. Your solution may address some or all of the following topics, query language, dynamic retrieval, crawling, cacheing, translation, inference, scalability, consistency, security, ...

The final, interactive version of WebScripter needs access to both the ontologies (pre-tripelization) to guide the user in *constructing* the query, as well as access to data (post-tripelization) to efficiently *executing* the query.

As far as DAML **ontologies** are concerned we propose:

```
DAML source pages on the Web
  |
  AltaVista-like crawled index of ontology DAML pages (not triples!)
  |
  WebScripter
```

As far as DAML **data** is concerned we propose:

```
DAML data sources on the Web
  |
  on-demand puller or pre-run crawler
  |
  database that can handle millions of triples
  |
  WebScripter
```

4 Security Issues

Consider how this could be accomplished if some of the DAML content was sensitive information stored on multiple WWW sites protected by passwords and/or certificates. You may or may not have access to all of the data.

What we envision is that there is a per-user collection of their passwords and certificates. WebScripiter queries then run against the combination of the (probably crawled) public database and the (possibly crawled) per-user database of protected information.

WebScripiter currently already tags every triple it reads with the URL it came from (by running a loop of `rdffapi` calls on a per-input-URL basis). It could similarly tag the security status of every triple so that the DAML version of the WebScripiter report output can carry through the appropriate security tags (or omit secure output altogether).

We have a very real security issue with our current WebScripiter data - our project milestones, to-do items, and status are essentially publicly available to the world on a real-time basis right now (the only thing that protects them is "security by obscurity"). We would like to be able to share all of our data with certain parties (such as the subcontractors for our ANTS project), and just selected information with the rest of the world (e.g. our project descriptions and their rough status without the detailed internal discussions and to-do items). However, unless otherwise directed, we will not address these security issues on our own and instead focus on our core interactive-WebScripiter-reports effort.

5 Our Software

Extra Credit: Develop software to implement any or all of 3.

We have fully implemented the first query as stated above (URL for the output also above).

The software we have is a report instantiator that takes an XML definition of a report definition and fills it with data. It operates on top of Melnik's RDFAPI and works exclusively on the triplezied data. The content of every column to the right is computed from the content to the left (the first column that seeds the rows in the table is a special case - it specifies the type of instance that creates a row).

We believed we could put up a servlet that takes a WebScripiter report definition and fills it with data by today (2000-12-15) but did not make it, but it should be up before the end of the year (we will announce it separately). It will offer a library of WebScripiter reports that can be run as they are, or can be downloaded, modified, put on an arbitrary Web server, and then run from there (by posting the URL of the modified report definition to an HTML form on our Web site).

6 Tools Wishlist

Based on your architectural sketch in 3, propose any new entries to be added to the Tools Wishlist.

We don't believe anything on our wish list is novel, but there is probably some value in tabulating what every group is asking for, so here it is:

A Database that does very fast pattern matching against triples, and scales to a number of triples that would not fit into main memory.

A Crawler that munges all DAML content known to mankind and deposits it into the above database. WebScripiter report definitions currently explicitly list every DAML source URL that should be loaded - practical for our internal use of WebScripiter for project management but obviously impractical for searching the Web at large for content.

A DAML ontology validator that takes an ontology file and data files (separately) and reports on what in the latter is lacking with respect to the former. We have defined nine DAML ontologies so far, but they are all rather "flat" and rarely specify required cardinalities and data types as doing so is not very gratifying without being able to use them for validation.

7 Lessons Learned

Discuss any lessons learned or insights gained from this assignment.

One experience over the last weeks is that the most error-prone part of specifying WebScripiter report definitions is in mistyping a character or two in strings that should match a subject, predicate, or object of a triple. The most practical solution (for hackers) in working with RDF source files is to redirect all of the triples to a file and then copying and pasting the right strings to the WebScripiter report definition. We are scheduled to make WebScripiter reports be interactively producible in the second semester of our effort.

8 Where to Find Our Homework 2 Results

Post the homework results on your public project page, and send the URL to webmaster@daml.org.

This document is written in GNU texinfo format. The official location of its most readable format is <http://www.isi.edu/webscripiter/isidamlhw2.gen.html>.

A plain text version suitable for emailing is at <http://www.isi.edu/webscripiter/isidamlhw2.gen.txt>

An Adobe Portable Document Format version suitable for printing is at <http://www.isi.edu/webscripiter/isidamlhw2.gen.pdf>.