Tools API Breakout

The number of Semantic Web tools is growing very fast. When building Semantic Web applications, we would like to be able to assemble a set of tools, choosing the "best-of-breed" for each function: the best editor, the best visualization tool, the best inference engine, the best storage and retrieval capability, etc. Use of RDF and OWL as standard languages significantly eases the task of combining various Web tools. However, putting together tools derived from multiple sources is still a difficult task, in large part because there is no standardization at the API level.

Relational databases have SQL with attendant components as a standard query language, but more importantly they have JDBC/ODBC as a standard API. JDBC/ODBC is important for tool vendors, because it means they can build a tool independent of choice of RDBMS. (Not quite, for each vendor's SQL may be slightly different, but JDBC/ODBC helps a lot.) This same API also insulates product providers from version creep—as new versions of a database product are released, the tools that use it won't break if they limit their communications to JDBC/ODBC. Also, JDBC/ODBC handles the problem of remote communications. The Semantic Web is going to need the same kind of standards, for precisely the same reasons.

An alternative place to look for standard API's for the Semantic Web is the XML community. Some Semantic Web folks are pushing very hard for this approach.

This brings up an interesting question: “Should Semantic Web tools be more XML-like, or more database like?” In some cases, the database analogy is much better than the XML analogy. In others, the reverse is true. Each viewpoint brings lessons learned on API design and use that might apply to the Semantic Web.

Its probably too early (maturity-wise) to think about setting up a standards committee for a Semantic Web API, but its not too early to begin comparing the requirements and capabilities of various systems and tools, and to begin to understand what kinds of features such an API might have.

Objectives of APIs

- Language and OS neutral
  - Bindings for specific languages, esp. Java, .net, ???
- Open, vendor neutral
- Scalable
- Incremental implementation complexity with minimal mandatory subset (cf. DOM1, 2, 3)
- Distributed representation
- Common security features
- Implementable by semantic web services
- Leverage existing standards and approaches, especially work being done by DAWG
  - SPARQL - query language, protocols? Rec. by Apr. 05
- Support transactional sessions (some?)

Function classes needing Standard APIs

- Triple store (query, store)
- Reasoner
- Editor (include graphical) by language (OWL lite, … , SWRL, …)
- Parser
- Validator
- Translator and mapper
- “Semantinator”

Example APIs to Consider*

- Models from database world
  - SQL, JDBC/ODBC
- Models from XML world
  - DOM, Sax
- RDF/OWL APIs
  - Parsers: ARP, OWL API (U Manchester)
  - Reasoners: Jena, DIG
  - Editors: Protégé, SWeDE, Eclipse-based
  - Triple-stores: Sesame, Kowari, Tucana

**not complete listing*

Methodology

- Define and prioritize function classes
- For each function class:
  - Specify requirements and use cases
  - Prioritize and stage requirements
  - Collect existing options
  - Propose new approaches
  - Evaluate and choose
  - Design/specify API and develop metrics & testing approach
  - Build reference implementation(s)
  - Test & evaluate
  - Next stage

DAWG doing some of these tasks!! Coord.

Deliverables

- Description of function classes
- For each function class:
  - API requirements & use cases
  - Minimal API specification
  - Metrics & testing approach
  - Reference implementation & evaluation results
  - Next stage API specification, etc.

Schedule

- Description of function classes – +3 months
- For each function class (*some easier than others*):
  - API requirements & use cases – +6 months
  - Minimal API specification – +9 months
  - Metrics & testing approach – +9 months
  - Reference implementation & evaluation results – +15
  - Next stage API specification, etc. – next year(s)
- Submit for standardization (W3C?? ISO??) after release and use for ?? months (years?)

Faster if DAWG efforts line up