Operationalizing the Semantic Web: A Prototype Effort using XML and Semantic Web Technologies for Counter-Terrorism

September 9, 2004

M. Personick*, B. Bebee*, B. Thompson
SAIC/Advanced Systems & Concepts
(bebeeb@saic.com)

B. Parsia
The University of Maryland, College Park
Maryland Information and Network Dynamics Lab
Semantic Web Agents Project

C. Soechtig
Object Sciences Corporation

Advanced Systems & Concepts (ASC)

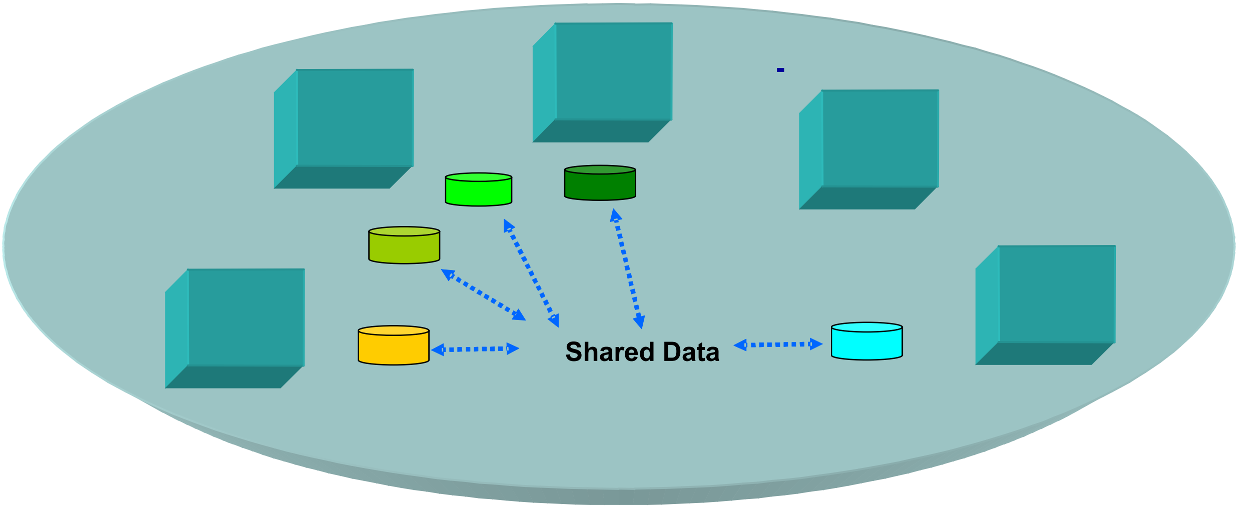
*Presenter

Our challenge is your challenge…

- **A key challenge in Counter-Terrorism efforts is discerning and revealing group structure from diverse information resources with varying quality**
  - Understanding Groups and Actors
  - Identifying links and relationships between Groups and Actors
- **Increasingly sophisticated information and tool sharing across organizations is a ubiquitous subtext**
- **Create an environment where culturally diverse and physically distributed actors can:**
  1. Expose resources in a controllable manner
  2. Participate in collaborative analytical processes
  3. Federate insights and knowledge about aggregate resources

Isn't that the goal of the Web?…

An Experimental Network enables the opportunity



- **Prototyping on experimental network:**
  - Multi-Agency Participation and Data Sharing
  - Operational Data
  - R&D Technologies
  - Access to users
- **Challenge of quickly bringing new repositories onto the network**
  - Poorly described information
  - Unique Schemas
  - Favored Tools

Three Distinct Challenges Encountered by the Team

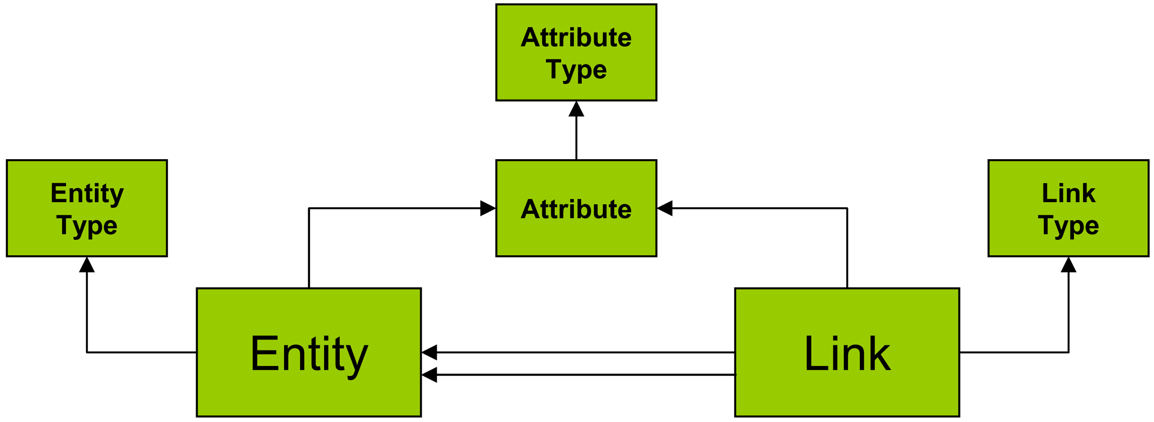
- **Lack of repository schema descriptions**
  - Often little or no description is available
- **Tight coupling between analytical tools and specific schemas**
  - Sharing information between tools is complicated
  - Contributing insights back into repositories is nearly impossible
- **Robustness to rapidly changing domains**
  - Nature of domain precludes *a priori* knowledge of information to be stored or the analyses performed

Before the current efforts involving web services and Semantic Web languages, the team applied traditional relational approaches.

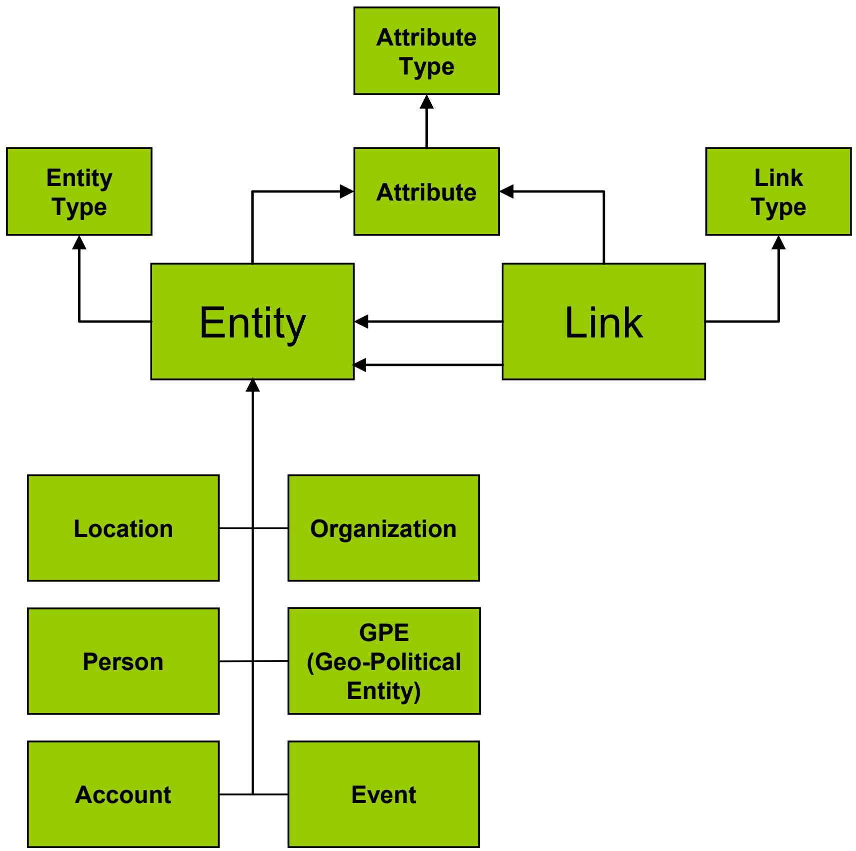
Research Efforts to Address the Challenges

- **Initiated prototyping efforts combining the use of XML Specifications and Semantic Web Languages**
  - Holistic application of Web Architecture principles in conjunction with Semantic Web languages
  - Use the principles to enable a loose-coupling between application and persistence layer
- **Apply the results of the prototyping efforts within the Experimental Network**
  - Initial repository description using RDF/OWL
  - The ability to make semantic queries of large distributed repositories using XPointer
  - An application using these technologies to visualize a link-chart
- **Lay the foundation for future efforts focused on federation of resources described with Semantic Web languages**

- Initial strategy was to consolidate information into common schema: Evidence Database (EDB)
- Relational schema designed for maximum flexibility to accommodate information sharing amongst different user communities



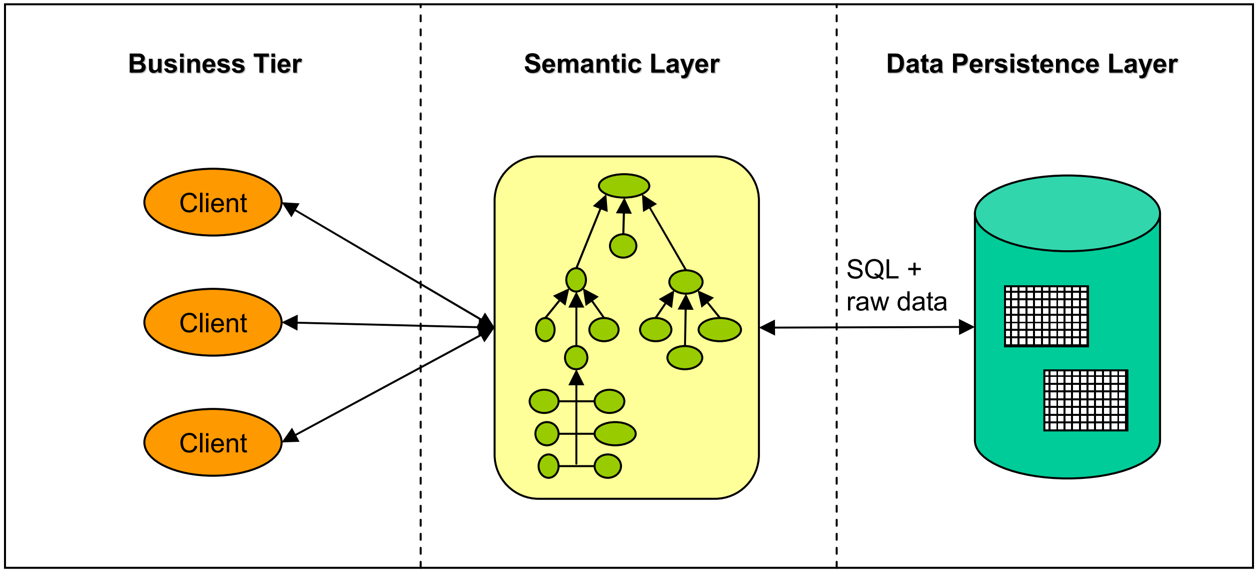
Technical Background: Relational Strategy



- **Demands for stronger typing led to six Entity subclasses**
- **Now information about entities and their attributes stored in 8 tables instead of 2**
- **Performance problems necessitated changes to the database schema**
  - Tight coupling between application code and the database made changing the relational schema very difficult.
- **An abstraction layer in between the application and persistence layer was required**

Three primary concerns:

1. Promote separation of concerns between application layer and data persistence layer.
2. Dramatically improve query performance.
3. Maximize overall system scalability.

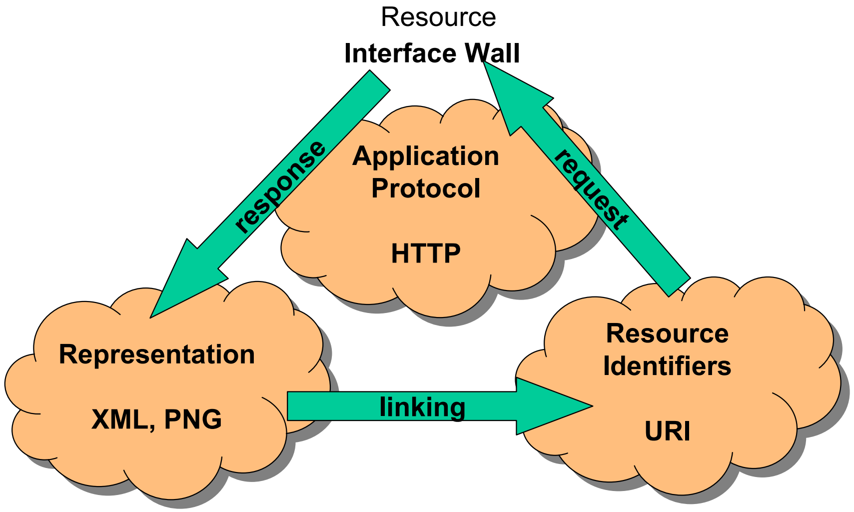


- **Web Services approach for accessing resources**
  – REpresentational State Transfer (REST architectural style)
- **Data model to represent semantics of resources**
  – Resource Description Framework (RDF)
- **Serialization format for resource transmission**
  – Extensible Markup Language for RDF (RDF/XML)
- **Rich mechanism to enable querying through Web Service layer**
  – Server-side XPointer
- **Expressive semantic query language to retrieve sub-resources**
  – RDF Data Query Language (RDQL)

Technology Sidebar: REST Uniform Interface Constraints

1. Resource is the unit of identification
   - **Universal document identifiers.**
2. Resource state is manipulated through the exchange of representations
   - **Document interchange.**
3. Generic interaction semantics.
   - **Create, update, read & delete documents.**
4. Self-descriptive messaging.
   - **Intermediaries, e.g., caches and security firewalls.**
5. Hypermedia is the engine of application state
   - **Hyperlink traversal plus form-based data submission.**

Generic Interaction Semantics of *REST Rotary Engine*



2xx - Success
3xx - Redirect
4xx - Client Error
5xx - Server Error

OPTIONS
HEAD
GET
POST
PUT
DELETE

Resource Description Framework (RDF)

- Resource Description Framework (RDF) language and Topic Maps considered for data model
- RDF chosen due to ease of use and wider acceptance
- EDB's structure mapped to an RDF vocabulary using RDF Schema and OWL
- Model objects (sub-graphs) will be serialized as RDF/XML during transport

Solution Architecture: REST-ful Query Mechanism Server-Side XPointer

- **With semantic data model defined, repository can be treated as a semantic store**
- **However queries were previously performed in a fashion that was tightly coupled to relational model**
  - Java & JDBC as query mechanism
  - SQL as query language
- **Solution architecture needs a query mechanism suitable for decoupled interaction with the semantic data model through a REST-ful Web Service layer**
  - Taking a resource-centric view, entire contents of an EDB repository instance looks like a very large graph
  - This EDB graph is addressable by a URI provided by the Web Service and can be represented as RDF/XML
  - However the <u>graph is generally too large to even load into memory</u>, much less transmit to clients
  - Need a request mechanism so that <u>only subgraphs are transmitted</u>
- **This sounds somewhat similar in concept to URI fragment identifiers**

Technology Sidebar: XPointer

- **W3C XPointer framework provides extensible processing model for URI fragment identifiers**
  - XPointer is an XML linking technology
  - Not to be confused with XPath, which is tied to an XML syntax
- **Highly extensible by way of XPointer schemes defined in namespaces:**
  - `#xmlns(x=http://www.myorg.org/scheme1)x:xpath(//title)`
  - `#xmlns(r=http://www.myorg.org/scheme2)r:rdf-query(…)`
  - `#xmlns(q=http://www.myorg.org/scheme3)q:tm-query(…)`
- **However keep in mind that URI fragment identifiers are not passed with a normal HTTP request**
  - Therefore client must GET entire representation and then apply XPointer processor
  - Extensible, but not scalable

Server-side XPointer integrates XPointer with HTTP

- **The HTTP/1.1 protocol defines an extensible request header named "Range"**
  - The client specifies a "range-unit", e.g.,
    "xpointer"
    and a "range-value", e.g.,
    "xmlns(x:http://mindswap.org)x:rss(…)"
- **The server sends back only the identified sub-resources for the negotiated content type (or a status code indicating an appropriate error)**
- **Provides scalable retrieval and update of XML sub-resources**
- **Can be used anywhere URIs are used**
- **Can use both syntactic and logical addressing schemes**

Solution Architecture: REST-ful Query Mechanism
Server-Side XPointer

http://www.myorg.org/mydoc

```
GET /mydoc HTTP/1.1
Host: www.myorg.org
Accept: text/xml
```

```
HTTP/1.1 200 Ok
Content-Type: text/xml
<!DOCTYPE foo [
<!ELEMENT foo (bar*)>
<!ELEMENT bar (#PCDATA)>
<!ATTLIST bar id ID #IMPLIED>
]>
<foo>
   <bar id="a12">Hello</bar>
   <bar id="a13">World</bar>
</foo>
```

http://www.myorg.org/mydoc#a13

```
GET /mydoc HTTP/1.1
Host: www.myorg.org
Accept: text/xml
Range-Unit: xpointer
Range: xpointer=a13
```

```
HTTP/1.1 206 Partial Content
Mime-Version=1.0
Accept-Range=xpointer
Content-Range=xpointer=a13
Content-Type=multipart/mixed

-------=_Part_
Content-Type: text/xml
Content-Length: 25

<bar id="a13">World</bar>
-------=_Part_--
```

RDF Data Query Language (RDQL)

- **Now that we've defined the query mechanism (XPointer) , we need to define an addressing scheme (query language)**
- **Don't want to lock ourselves into a syntax**
  - Again, XPointer does not necessarily imply XPath
  - Syntax lock-in and tight coupling lead to system fragility and evolutionary dead-ends
- **Need to define meaningful views of the data through a logical addressing scheme**
- **RDF Data Query Language allows for logical addressing at the data model and ontology level, avoiding serialization syntax**

Technology Sidebar: RDQL

```
GET /edb HTTP/1.1
Host: www.myorg.org
Accept: application/rdf+xml
Range-Unit: xpointer
Range: xpointer=xmlns(ms=http://mindswap.org)
       ms:rdql(SELECT ?e WHERE (?e <edb:hasEntityType> ?t)
                               (?t <edb:hasDescription> "Person")
               USING edb FOR <http://www.myorg.org/edb-schema/1.0#>)
_________________________________________________________________________

HTTP/1.1 206 Partial Content
Content-Type: application/rdf+xml
<!- Only the selected sub-graph is transmitted to the client. -->
<rdf:RDF … />
```

- **RDQL is a W3C RDF data query language based on SquishQL**
- **Designed to extract information from RDF graphs**
- **Query consists of a graph pattern, expressed in triples, that is matched against an RDF graph**
  - Each triple pattern comprised of named variables and RDF values (URIs and literals)
  - Query can additionally have constraints on values of those variables returned in answer set
  - Query also specifies which variables are Matching sub-graphs are returned in triple form
  - Above query selects all entities and returns RDF/XML serialized representation of all Person sub-resources
- **Several implementations exists, notably JENA**

- **Outcome of efforts described will be applied on an experimental network with real users and data in the counter-terrorism domain**
- **Migration to the solution architecture is an ongoing process expected to be demonstrated December 2004**
- **To date progress has been made in all key areas**
  - RDF vocabulary for data defined
  - RDF/XML applied for instance data
  - Server-Side XPointer processor in progress
  - RDQL query scheme in progress
  - REST-ful Web Service layer still in design stage

Solution Architecture: Current Status & Reflections

- **The application of REST, RDF, and server-side XPointer will greatly enhance information sharing capabilities**
  - The semantic data model can remain fixed while the underlying persistent store changes, allowing for separation of concerns
  - This separation of concerns will allow the database team to optimize the relational tables as needed
  - Loosely coupled design promotes flexibility and scalability
- **Deploying Semantic Web languages in the context of overall Web Architecture is critical**
- **Could we have achieved these goals without semantic web technologies?**
  - No! Tight coupling constrains flexibility and scalability

Questions