A Technical(ish) Introduction to the SADI Semantic Web Service Framework

Alexandre Riazanov

C-BRASS project

University of New Brunswick, Saint John

Montreal trip, Aug-Sep 2010

Part I: SADI in General

Part II is about a case study

SADI Framework Outline

- **S**emantic **A**utomated **D**iscovery and **I**ntegration
- SADI is a **set of conventions** for creating Semantic Web Services that can be **automatically discovered and orchestrated**.
- SADI does not create new technologies or message formats. It relies on well-established standards: RDF, OWL and HTTP.
- A SADI service consumes an RDF graph with a designated node and produces an RDF graph about the same node with some **new properties attached**.
- Declaration of the new property predicates describes the semantics of the service and makes it *discoverable*.

SADI Service Examples

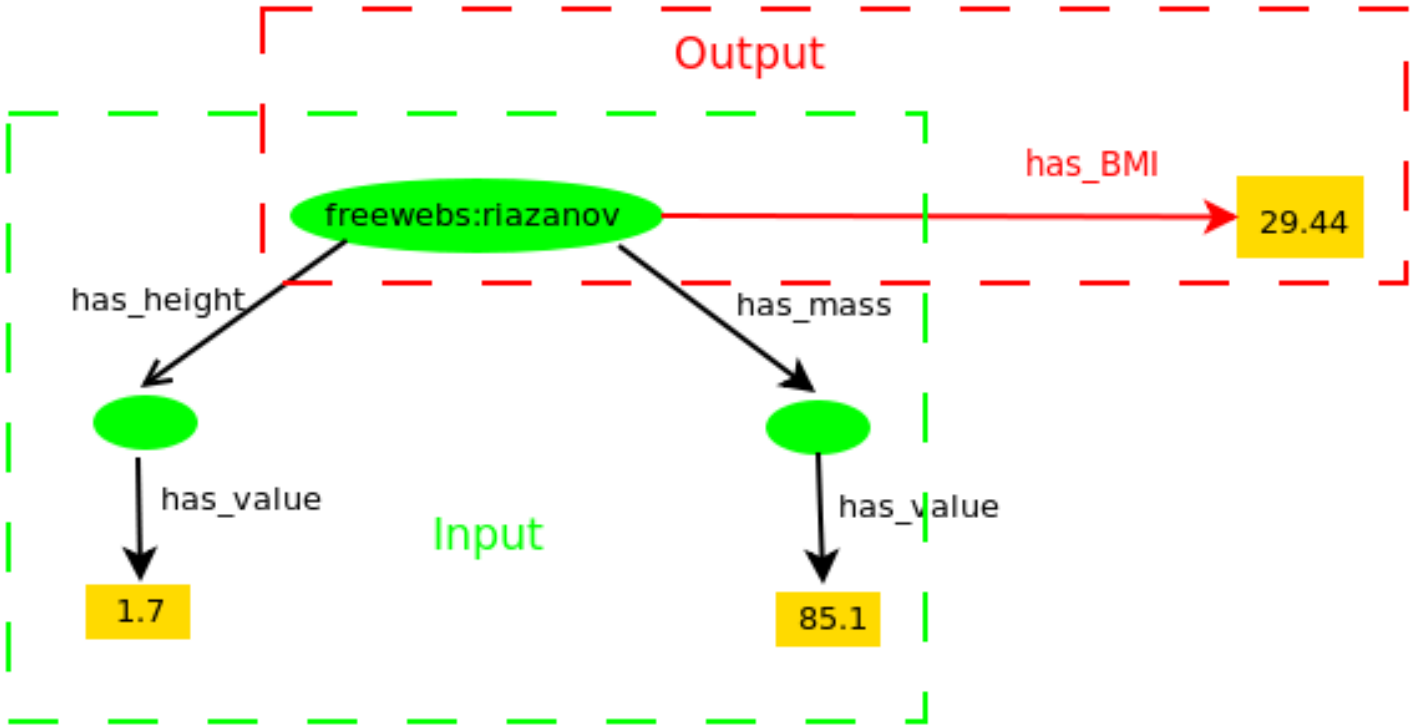
SADI services can serve information both from DBs and by running analytical resources.

- Gene Ontology record ID $\longrightarrow$ corresponding GO term
- EntrezGene ID $\longrightarrow$ symbolic name of the gene and its textual description
- KEGG gene record $\longrightarrow$ UniProt equivalent
- weight and height of a person $\longrightarrow$ Body Mass Index
- patient record data (age + some test results) $\longrightarrow$ various risk scores, such as coronary desease risk score

More interesting services will be discussed in Part II – our own case study.

SADI Service Input/Output

- RDF $\longrightarrow$ RDF
- One main input node $\longrightarrow$ same node in the output + new properties
- Output = annotated input



Input Class

- The OWL class of the main input node, specifying what the service expects to see in the input.
- Can be just an OWL class name from a domain ontology.
- Typically, a bunch of existential property restrictions: *which properties on the main input node the service expects.*
- If the input class is defined as

```
(has_height some (has_value some xsd:float))
   and
(has_mass some (has_value some xsd:float))
```

the service expects something like this in the input:

```
<http://www.freewebs.com/riazanov> has_height [has_value float"1.7"] .
<http://www.freewebs.com/riazanov> has_mass [has_value float"85.1"] .
```

Output Class

- The OWL class of the main output node specifying what the service promises to produce as the output.
- Should imply some existential property restrictions. *The corresponding predicates semantically define what the service does.*
- The service computing BMI has this output class:

  ```
  (has_BMI exactly 1 xsd:float)
  ```

  The service advertises that it can compute the Body Mass Index.
- We say that the service *annotates* its input with the predicate has_BMI.

  Typical output:

  ```
  <http://www.freewebs.com/riazanov> has_BMI float"29.44"
  ```

Making Use of Semantic Service Declarations: Automatic Service Discovery

- Services are registered in a registry and indexed by the predicates they provide.
- If a client program needs to compute the BMI of a person, it will
  - look for a service providing has_BMI
  - check, possibly with the help of a reasoner, that the client's data match the input class of a found services
- We have data and declaratively (semantically) specify what we would like to compute based on this data. A program finds for us exactly the tools that can do the job.
- *This is fully automatic discovery!*

Automatic Composition

- Often, what the client wants cannot be achieved with just one service, but can be achieved with a sequence of calls to different services.
- If we only have a person's URI, but there is a service decorating the URI with *has_height* and *has_mass*, the client program can figure out that it should call this service prior to calling the has_BMI service.
- *This is automatic service composition!*
- Service calls can be interleaved with arbitrary reasoning steps deriving more facts logically, using an OWL TBox or a rule set

$\Rightarrow$ *Very intelligent computing!*

Automatic discovery and composition with SHARE

- SHARE is a prototypical client for SADI services.
- For the user, it is
  - SPARQL querying over a virtual RDF graph build from
  - some initial RDF graph
  - by adding information that can be computed by calls to SADI services
  - and derivations made by an OWL reasoner
- The user just submits a SPARQL query and gets the results – *the service discovery and composition is completely transparent.*

SHARE snapshot

SPARQL query:

```
PREFIX bmi: <http://sadiframework.org/examples/bmi.owl#>
SELECT ?patient ?bmi
FROM <http://biordf.net/cardioSHARE/patients.rdf>
WHERE {
   ?patient rdf:type patients:AtRiskPatient .
   ?patient bmi:BMI ?bmi
}
```

View results as RDF. There were warnings executing the query. Click for details.

Submit

Query results

bmi	patient
26.32017237098755	http://biordf.net/cardioSHARE/patients.r

What is so cool about it?

- We compute **regular SPARQL queries**. There is nothing specific to SADI here
- But the queries are computed over an RDF database that is **generated dynamically** by adding different service call results
- Some of the **data does not exist** at the time of issuing the query in any form – it is only created as a result of service executions
- There is literally **no effort** by the user to integrate the data – SHARE does it for him
- It's a bit like AI, isn't it?

Other End-User Tools

- SADI plugin for Taverna
  - Taverna is a Workflow Management system – allows to quickly create complex scientific workflows by combining various components: Web services, scripts, subworkflows, etc
  - The composition of services is manual, but with a lot of help from the system: mapping of data to the input of a service is automatic, and the system also suggests services that can apply to a piece of data
- SADI plugin for Sentient Knowledge Explorer
  - Sentient KE: commercial tool from IO Informatics
  - Essentially, an RDF graph browser
  - SADI plugin allows to add virtual RDF subgraphs corresponding to available services, to the browsing space
  - The user can select some data and see what predicates can be added to the data with available SADI services

Writing SADI Services is Easy!

- There is a Java library that isolates the business logic of a service from the environment: HTTP POST → Jena Model (RDF graph) → . . . business logic . . . → Jena Model (RDF graph) → HTTP response
- Service provider implements all the business logic in

```
void processInput(Resource input,Resource output)
```

- Whole project skeleton is generated: provider only needs to supply basic parameters, like the service name. Or use a skeleton Eclipse project.
- Similar things for Perl exist.
- The hardest part is knowledge engineering: defining what the service does ontologically

Typical Resource Publishing Scenario

- **Identify a source of data or a piece of software you want to publish with SADI**. For example, if you have a DB about some substances with literature reference, you can have a service linking your substances to PubMed IDs
- **Model your data semantically:** find ontologies describing your domains and decide how you data will be expressed in the terms of these ontologies. *You may need to write your own ontology*, although it may be a really simple one, say, doable in 1-2 days
- **Model your services I/O semantically:** decide how to describe the operation of your services in the terms of the domain ontologies. Typically, amounts in writing a small ontology
- **Code the business logic of your services in Java or Perl.** If a service wraps a DB, convert the input RDF into a query and the query results back to RDF. The effort is typically negligent, compared to the modelling

SADI Work in UNB Saint John

- General direction: **more bioinformatics resources as SADI services**
- Mutation Impact text mining: text → mutation, protein, impact data with literature references
- Mutation Impact DB: mutations, proteins, impacts → related entities with literature reference
- Access to PubMed Central and LipidMaps DBs (simple wraps for their Web services)
- Work in progress: text mining for lipids
- Work in progress: lipid classification service. Lipid description in terms of functional groups → lipid class. A service at Dumontier Lab in Ottawa will create descriptions from chemical structure descriptions.

Collaboration Model

- We are actively looking for bioinformatics resources to publish them as SADI services
- We can help you to increase the value of your resources by *integrating them in a broader biological and biochemical context* $\Rightarrow$ new users, new application scenarios, new research, bigger impact
- Option 1: give us your code and/or access to your data, and possibly ontologies, and let us to wrap it as SADI services. Modeling the data may require co-operation. We can host the resulting services
- Option 2: you adopt the SADI technology with our help after a few pilot efforts with your data/software
- Research and publishing opportunities: a lot of work in the data integration domain, new intelligent tools/applications become possible

Part II: Case Study with Mutation Impact Mining Pipeline and DB

Part I is a general intro to SADI

Mutation Impact Pipeline

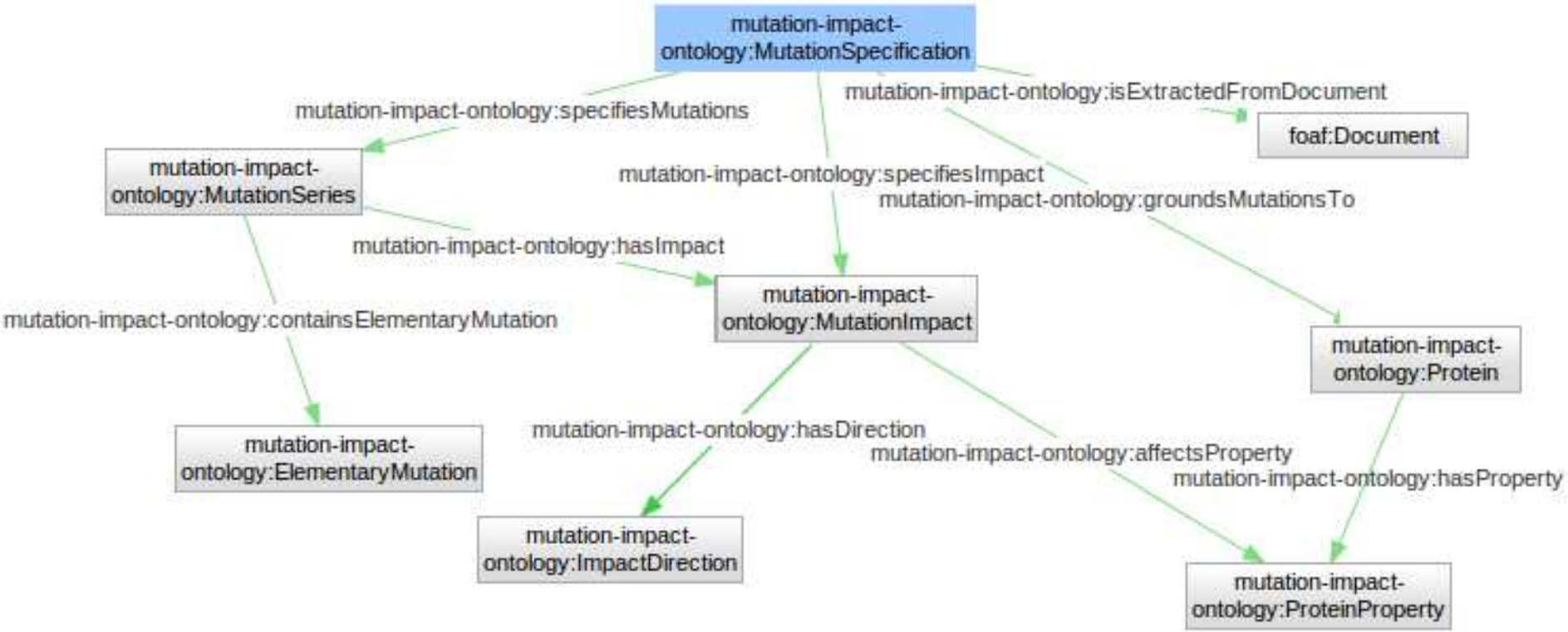
The haloalkane dehalogenase from the nitrogen-fixing hydrogen bacterium Xanthobacter autotrophicus GJ10 (DhlA) prefers 1,2-dichloroethane (DCE) as substrate and converts it to 2-chloroethanol and chloride ... DhlA shows only a small decrease in activity when Trp-125 is replaced with phenylalanine

$\Longrightarrow$

- "haloalkane dehalogenase" is a protein
- its UniProt Id is P22643
- "Trp-125 is replaced with phenylalanine" is the point mutation W125F
- "activity" is the protein property being affected, GO_00188786 in Gene Ontology
- "decrease" means the impact is negative

UNB-Concordia joint work. GATE pipeline wrapped as a Java library. Outputs Java objects representing mutation specifications: statements about mutations and their impacts on protein properties.

Mutation Impact Ontology



Main Service: Text → Mutation Specifications

- Just wraps the Mutation Impact mining pipeline as a SADI service
- Identifies input text in the service input, submits it to the pipeline and converts the results to an RDF graph

Input:

```
<http://example.com/text1>   rss:link   anyURI"http://example.com/text1"
```

Output:

```
<http://example.com/text1>  foaf:topic              midb:mutationSpec243
miodb:mutationSpec243       mio:groundsMutationsTo  miodb:protein528
miodb:protein528            mio:hasSwissProtId      "P22643"
miodb:mutationSpec243       mio:specifiesImpact     miodb:mutationImpact624
miodb:mutationImpact624     mio:hasDirection        mio:Positive
miodb:mutationImpact624     mio:affectsProperty     miodb:proteinProperty326
miodb:proteinProperty326    mio:hasType             GO:GO_0018786
                    . . . . . .
```

Services based on Mutation Impact DB

- Wildtype protein $\longrightarrow$ mutation specifications (complete description of)
- Mutant protein $\longrightarrow$ mutation specifications
- Specific property of a specific protein $\longrightarrow$ known mutation impacts
- Mutation impact (on a specific property of a specific protein) $\longrightarrow$ mutation specifications
- Bio entity type (e.g., mio:Protein or mio:PointMutation) $\longrightarrow$ known instances
- Set of elementary mutations $\longrightarrow$ subsets described in the literature (with links to the corresponding mio:MutationSpecification)

Use Case 1

- A protein engineer is looking for mutations that can improve catalitic activity of an enzyme
- Query: find all mutations and the structure images of wild type proteins that were mutated, where the impact of the mutation is an enhanced haloalkane dehalogenase activity (GO_0018786)
- Predicates from our ontology take us from GO_0018786 to mutations and proteins: $proteinPropertyHasType + affectsProperty + hasDirection + specifiesImpact + containsElementaryMutation + groundMutationsTo$
- Two external SADI services provide $has3DStructure$ to link proteins to their structure descriptions, and $hasJmol3DStructureVisualization$ to link the structure to a 3D image file

Use Case 1 SHARE Query (simplified)

```
SELECT DISTINCT ?StructureImage ?NormalizedMutation
WHERE {
   # Property type --> property
   ?Property mioe:proteinPropertyHasType go:GO_0018786 .

   # property --> positive impact
   ?Impact mio:affectProperty ?Property .
   ?Impact mio:hasDirection mio:Positive .

   # impact --> mutation spec --> mutation series --> elementary mut.
   ?MutationSpecification mio:specifiesImpact ?Impact .
   ?MutationSeries mio:mutationSeriesIsSpecifiedBy ?MutationSpecification .
   ?MutationSeries mio:containsElementaryMutation ?Mutation .
   ?Mutation mio:hasNormalizedForm ?NormalizedMutation .

   # wildtype protein
   ?MutationSpecification mio:groundMutationsTo ?Protein .

   # protein --> structure --> image file
   ?Protein pred:has3DStructure ?Structure .
   ?Structure objects:hasJmol3DStructureVisualization ?StructureImage . }
```

Use Case 2

- Systems biologist is seeking to understand the likely impact of reported mutations on signalling or metabolic pathways in which the mutated protein participates
- Query: find pathways, together with their images, that might have been altered by a mutation of the protein P22607
- Our predicates take us from P22607 to its mutations with positive or negative impact on some property:
  $groundsMutationsTo + specifiesImpact + hasDirection$
- External predicate $isEncodedBy$ links P22607 to the corresponding gene
- $isParticipantIn$ links the gene to pathways
- $visualizedByPathwayDiagram$ links a pathway to its image

Use Case 2 SHARE Query (simplified)

```
SELECT DISTINCT ?Pathway ?PathwayDiagram
WHERE {
   # protein --> mutation specification
   ?MutationSpecification mio:groundsMutationsTo uniprot:P22607 .

   # mutation spec --> impact
   ?MutationSpecification mio:specifiesImpact ?Impact .

   # impact cannot be neutral
   {?Impact mio:hasDirection mio:Positive}
      UNION {?Impact mio:hasDirection mio:Negative} .

   # protein --> gene
   uniprot:P22607 pred:isEncodedBy ?Gene .

   # gene --> pathway
   ?Gene ont:isParticipantIn ?Pathway .

   # pathway --> image
   ?Pathway pred:visualizedByPathwayDiagram ?PathwayDiagram
}
```

Use Case 3

- A researcher in drug discovery is looking for existing drugs targeting a new disease condition
- Query: find all drugs related to mutated proteins, together with their interaction partners, where the mutation impact is a increased carbonic anhydrase activity (GO_0008270)
- Our predicates link GO_0008270 to proteins via the instances of this protein properties, positive impacts and mutation specifications
- External ontologies facilitate the linking of proteins to the IDs of drugs affecting them
- Another external predicate, $hasMolecularInteractionWith$, links the proteins to proteins they interact with

Use Case 3 SHARE Query (simplified)

```
SELECT ?DrugName ?InteractingProtein
WHERE {
   # protein property type --> instance
   ?Property mioe:proteinPropertyHasType go:GO_0008270 .

   # protein property --> positive impact
   ?Impact mio:affectProperty ?Property .
   ?Impact mio:hasDirection mio:Positive .

   # impact --> mutation spec
   ?MutationSpecification mio:specifiesImpact ?Impact .

   # mutation spec --> protein
   ?MutationSpecification mio:groundMutationsTo ?Protein .

   # protein --> drug id --> drug name
   ?Protein objects:isTargetOfDrug ?Drug .
   ?Drug objects:hasDrugGenericName ?DrugName .

   # protein --> interaction partner
   ?Protein pred:hasMolecularInteractionWith ?InteractingProtein }
```

Use Case 4

- A researcher in genomics asks for all known mutations reported in the literature for a protein containing a non-synonymous SNP
- Query: find all reported mutations of the protein with the nsSNP rs2305178
- External predicate *correspondsToEntrezGene* finds the Entrez ID of the gene corresponding to rs2305178
- Predicates *correspondsToEntrezGene*, *hasRefSeqTranscript* and *isEncodedBy* link the Entrez gene ID to the sequence, to the KEGG gene ID, and then to the protein
- Complication: there are services that provide *correspondsToEntrezGene* and *hasRefSeqTranscript*, but no services yet providing their inverses: KEGG ID $\rightarrow$ sequence $\rightarrow$ Entrez ID, but not Entrez ID $\rightarrow$ sequence $\rightarrow$ KEGG ID
- Temporary solution: enumerate proteins from Mutation Impact DB, and then link them to other entities

Use Case 4 SHARE Query (simplified)

```
SELECT DISTINCT ?NormalizedMutation ?Document
WHERE {
   # enumerate known proteins
   ?Protein mioe:biologicalEntityHasType mio:Protein .

   # protein --> KEGG gene ID --> sequence --> Entrez gene ID
   ?Protein pred:isEncodedBy ?KeggGene .
   ?KeggGene objects:hasRefSeqTranscript ?RefSeq .
   ?RefSeq objects:correspondsToEntrezGene ?EzGene .

   # SNP --> Entrez gene ID (and join)
   dbsnp:rs2305178 objects:correspondsToEntrezGene ?EzGene .

   # protein --> mutation spec --> mutation and document
   ?MutationSpecification mio:groundMutationsTo ?Protein .
   ?MutationSeries mio:mutationSeriesIsSpecifiedBy ?MutationSpecification .
   ?MutationSeries mio:containsElementaryMutation ?Mutation .
   ?Mutation mio:hasNormalizedForm ?NormalizedMutation .
   ?Document foaf:topic ?MutationSpecification }
```