# Automated Generation of SADI Semantic Web Services for Clinical Intelligence

Sadnan Al Manir[1] Alexandre Riazanov[3] Harold Boley[2]
Artjom Klein[3] Christopher J.O. Baker[1,3]

[1]Department of Computer Science
University of New Brunswick, Saint John, Canada

[2]Faculty of Computer Science
University of New Brunswick, Fredericton, Canada

[3]IPSNP Computing Inc., Canada

International Workshop on Semantic Big Data (SBD 2016)
San Francisco, USA
July 1, 2016

# Outline

# Outline

# Clinical Intelligence

- A research and engineering discipline
- Dedicated to the development of tools for data analysis for
  - clinical research
  - surveillance
  - effective health-care
- Goal: Self-service ad hoc querying of clinical data
- Issue: When data are schema-defined, in relational form, querying requires IT skills that not many clinicians have

# Motivation

Current practice in Hospital-Acquired Infection (HAI) Surveillance

- Infection data stored in Relational Databases (RDBs)
- Infection control specialists (i.e. domain experts)
  - need to access the RDBs
  - are familiar with the terminologies of their domain
  - typically lack IT expertise for
    - integrating information from RDBs
    - writing SQL queries
    - writing complex program code
  - have to rely on IT personnel for all these tasks

Consequences

- Decision-making about infections delayed
- Inefficient HAI surveillance
- Patient risk (52 percent of all hospital deaths related to HAI)

# Outline

# Research Approach

Proposed Solution: Semantic Querying (SQ) services over RDBs

- Hospital Acquired Infections – Knowledge in Use (HAIKU) applied RESTful Web services and Semantic Automated Discovery and Integration (SADI) design pattern
- HAI data: The Ottawa Hospital Data Warehouse (TOH DW)
- SADI supports ad-hoc, self-service, semantic querying over relational data in Clinical Intelligence
- SADI Semantic Web services used over Relational TOH DW
  - Similar to Relational-to-RDF translators (e.g. D2R) and Ontology-Based Data Access (e.g. ontop, MASTRO),
  - Service-based approach is more flexible, allowing access to both static data services and algorithmic resources

# Outline

# Our Contribution

Extending a prototype architecture [2] to a fully operational SADI service generation framework called **Valet SADI**:

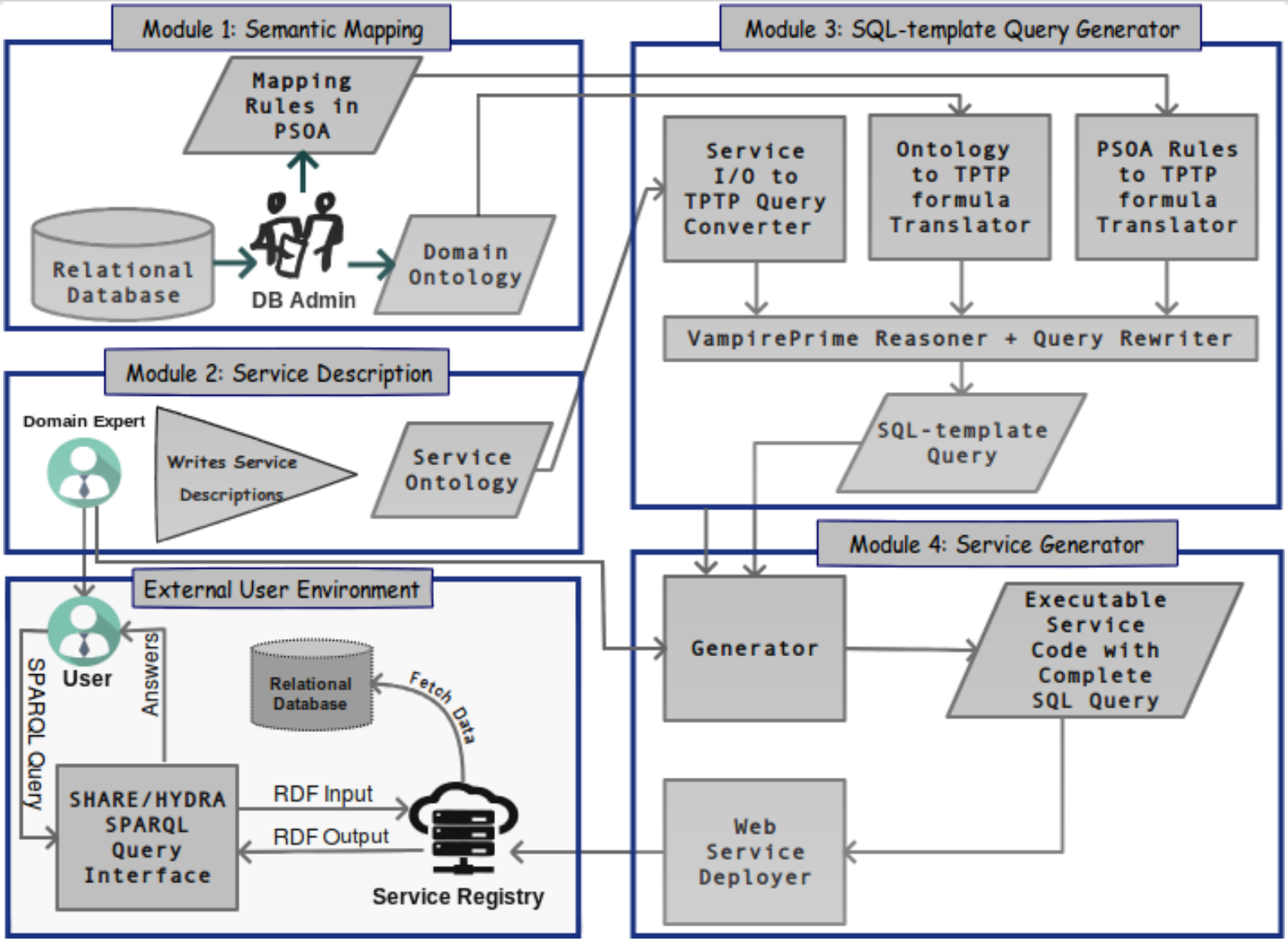
- Valet SADI based on semantic query rewriting
  - Mapping rules are specified manually between the domain ontologies and the RDBs (Quality Control)
- Valet SADI's Java implementation auto-generates SADI service Java code as part of a Maven Web application
  - Declarative I/O descriptions specified in OWL
  - Semantic mapping of source relational data specified in Positional-Slotted Object-Applicative (PSOA) RuleML [3]

# Our Contribution (Cont'd)

Benefits:

- Services can be created by non-IT users without knowledge of the Java programming language
- Users specify declarative mapping rules
  - No extra burden - same starting point for service creation
  - Less error-prone than Java-plus-SQL programming
- Executable services are generated:
  - Declarative I/O descriptions are rewritten into SQL queries
  - Java servlet code for the SADI services is generated
  - SQL queries are placed in an appropriate code block
- Implementation is domain-independent, given that mappings can be specified for each domain

# Architecture for Generating SADI Semantic Web Services



# Outline

# Example

- Data extract from clinical research DW of TOH
  - Tables representing patients and possible diagnoses
- Target is to trace how patients are linked to diagnoses
  - "Find ICD-10 diagnosis codes for a patient based on patient id"
- Performed by creating composition of two separate services
  - The first service takes a **patient id** as input and retrieves their **diagnosis id(s)** as output
  - The second service takes a **diagnosis id** as input and retrieves its **ICD-10 code** as output

# Outline

# Relevant Schema from TOH DW

- Table `Npatient` contains
  - basic information about all patients
- Table `NhrDiagnosis` contains
  - information about diagnoses
- Table `NhrAbstract` contains
  - general abstract information
- A complementary ICD-10-like chart is shown with the tables

| Npatient | | |
|---|---|---|
| patWID | patLastName | patFirstName |
| 1 | Doe | John |
| 2 | Lee | Mary |

| NhrDiagnosis | | |
|---|---|---|
| hdgWID | hdgHraWID | hdgCd |
| ... | ... | ... |
| 57 | 315 | A49 |

| NhrAbstract | |
|---|---|
| hraWID | hraPatWID |
| ... | ... |
| 315 | 1 |

| Chart: Diagnosis Code-Description | |
|---|---|
| hdgCd | Diagnosis Description |
| A49 | Bacterial infection |
| A91 | Dengue haemorrhagic fever |

# Domain Ontology (HAI.owl)

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix : <http://cbakerlab.unbsj.ca:8080/haitohdemo/HAI.owl#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@base <http://cbakerlab.unbsj.ca:8080/haitohdemo/HAI.owl> .

@prefix : <http://cbakerlab.unbsj.ca:8080/haitohdemo/HAI.owl#> .

:Person rdf:type owl:Class .
:Patient rdf:type owl:Class ;
         rdfs:subClassOf :Person .
:Diagnosis rdf:type owl:Class .
:abstractRecordForPatient rdf:type owl:ObjectProperty .
:has_abstract_record rdf:type owl:ObjectProperty .
:has_diagnosis rdf:type owl:ObjectProperty .
:is_diagnosed_for rdf:type owl:ObjectProperty ;
                  owl:inverseOf :has_diagnosis .
:has_diagnosis_code rdf:type owl:DatatypeProperty ;
                    rdfs:range xsd:string .
```

# Semantic Mapping using PSOA RuleML

```
1  Document (
2   Group (
3    Forall ?patWID (entityForPatientToPatWID(entityForPatient(?patWID)) = ?patWID)
4    Forall ?P (entityForPatient(entityForPatientToPatWID(?P)) = ?P)
5    Forall ?patWID ?patLastName ?patFirstName (
6     Patient(entityForPatient(?patWID)) :-
7      db_Npatient(?patWID ?patLastName ?patFirstName))
8   )
9   Group (
10   Forall ?hraWID (
11    entityForAbstractTohraWID(entityForAbstract(?hraWID)) = ?hraWID)
12   Forall ?P (
13    entityForAbstract(entityForAbstractTohraWID(?P)) = ?P)
14   Forall ?hraWID ?hraPatWID (
15    abstractRecordForPatient(entityForAbstract(?hraWID) entityForPatient(?hraPatWID)) :-
16     db_NhrAbstract(?hraWID ?hraPatWID))
17  )
18  Group (
19   Forall ?hdgWID (entityForDiagnosisTohdgWID(diagnosisEntity(?hdgWID)) = ?hdgWID)
20   Forall ?P (entityForDiagnosis(entityForDiagnosisTohdgWID(?P)) = ?P)
21   Forall ?hdgWID ?hdgHraWID ?hdgCd (
22    Diagnosis(entityForDiagnosis(?hdgWID)) :-
23     db_NhrDiagnosis(?hdgWID ?hdgHraWID ?hdgCd))
24   Forall ?hdgWID ?hdgHraWID ?hdgCd (
25    has_abstract_record(entityForDiagnosis(?hdgWID) entityForAbstract(?hdgHraWID)) :-
26     db_NhrDiagnosis(?hdgWID ?hdgHraWID ?hdgCd))
27   Forall ?hdgWID ?hdgHraWID ?hdgCd (
28    has_diagnosis_code(entityForDiagnosis(?hdgWID) ?hdgCd) :-
29     db_NhrDiagnosis(?hdgWID ?hdgHraWID ?hdgCd))
30   )
```

**Equations in lines 3-4, 10-13 and 19-20:**

Axiomatize auxiliary functions like

`entityForPatient`, `entityForPatientToPatWID`

# Semantic Mapping using PSOA RuleML (Cont'd)

```
31   Group (
      % HAI:is_diagnosed_for links HAI:Diagnosis to HAI:Patient directly
32    Forall ?Diag ?Abs ?Pat (
33     is_diagnosed_for(?Diag ?Pat) :-
34      And(has_abstract_record(?Diag ?Abs)
35          abstractRecordForPatient(?Abs ?Pat)))
36   )
37 )
```

- Rules in lines 5-7 and 21-23:
  - Classify entities as instances of corresponding classes in the virtual semantic model (RDFized relational tables)
- Rules in lines 14-16 and 24-29:
  - Populate the properties abstractRecordForPatient, has_abstract_record, and has_diagnosis_code from records in the tables
- Rule in lines 32-35:
  - Virtual table is_diagnosed_for as join of two stored tables Npatient and NhrDiagnosis

# Outline

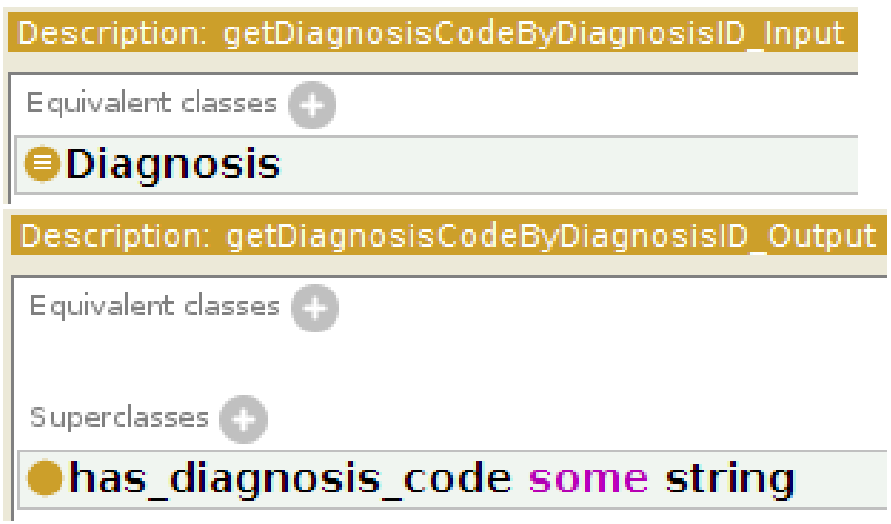
# Service I/O Descriptions

Declarative Input and Output Descriptions in OWL

1. `getDiagnosisIDByPatientID`: Retrieves diagnosis id (hdgWID) based on the patient id (patWID) and
2. `getDiagnosisCodeByDiagnosisID`: Retrieves diagnosis code (hdgCd) based on the diagnosis id (hdgWID)

**I/O for getDiagnosisIDByPatientID**


**I/O for getDiagnosisCodeByDiagnosisID**



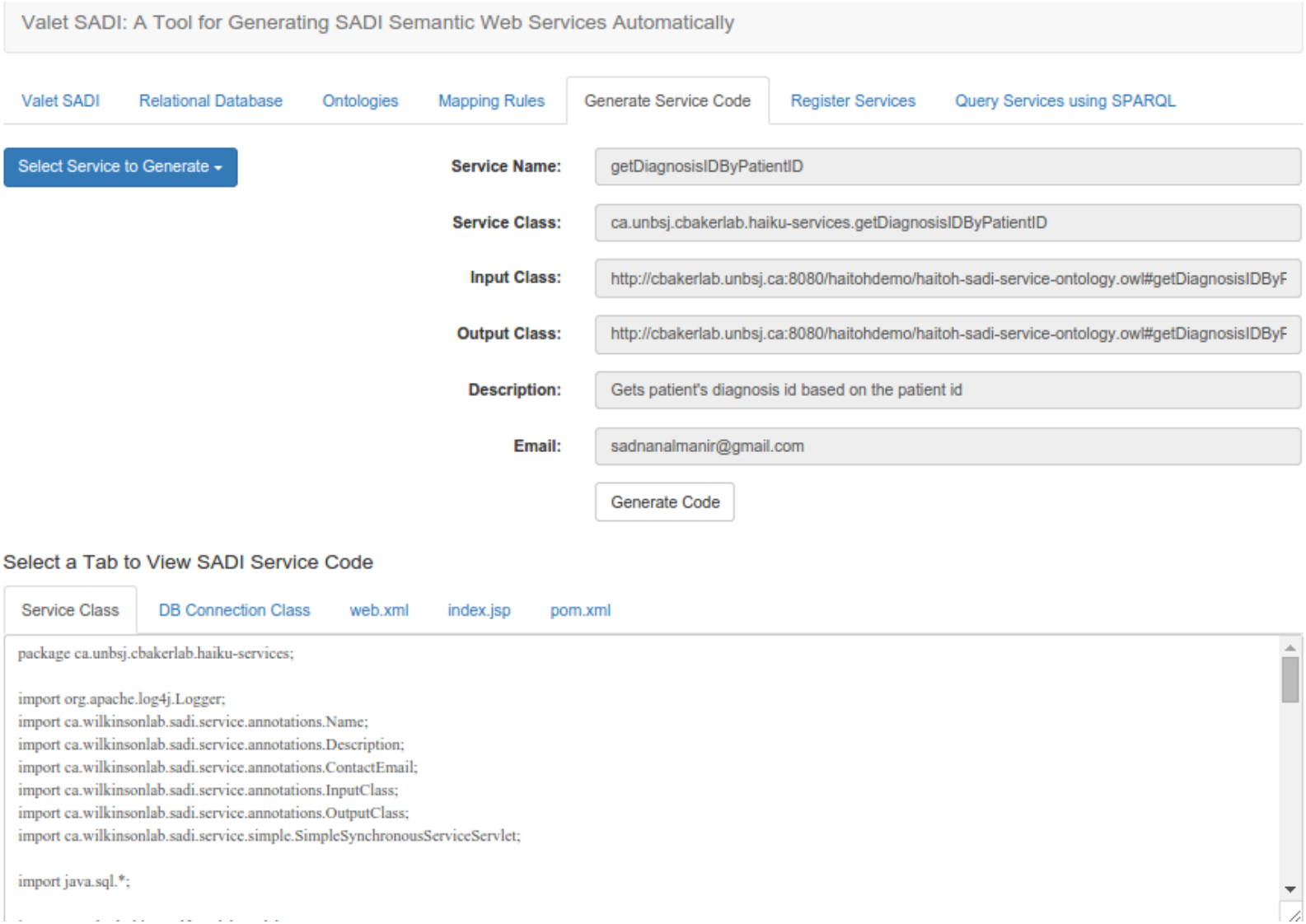
# Outline

# SQL-Template Query Generation

- Requires three inputs:
  - Declarative service descriptions in OWL
  - Domain ontology
  - Semantic mapping rules in PSOA for the DB schema
- These axioms are realized as specifications in TPTP (Thousands of Problems for Theorem Provers) and submitted to the VampirePrime reasoner, which
  - implements Incremental Query Rewriting (IQR)
  - uses auxiliary SQL-specific TPTP
  - produces SQL queries that are necessary and sufficient
- Each generated query is a template because
  - input is represented with formal parameters
  - instantiated every time the service is executed
  - input value substitutes the formal parameters

# Outline

# Generated Service Code for getDiagnosisIDByPatientID



# Preliminary Evaluation

- Valet SADI tested to perform the generation of two services:
  - getDiagnosisIDByPatientID
  - getDiagnosisCodeByDiagnosisID
- HYDRA [4] was used to test whether the generated services could be discovered, coordinated and invoked using SPARQL query execution where the end-user asks:
  "Find ICD-10 diagnosis codes for patient based on patient id"
- Additional services were generated from multi-table joins in our model database to asses whether this was achievable

# Preliminary Evaluation (Cont'd)

## SPARQL query executed on HYDRA

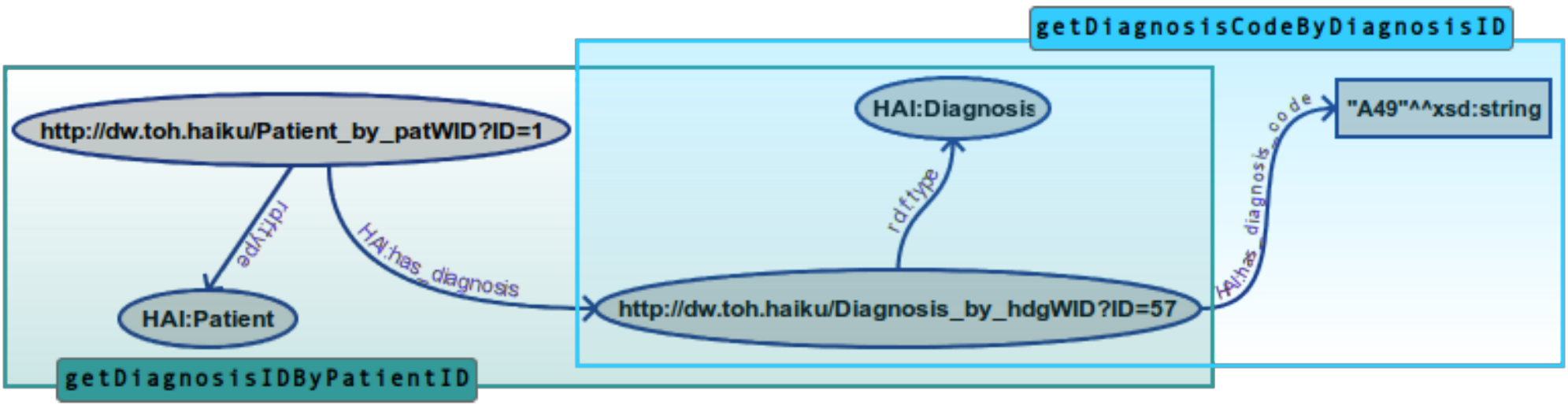
```
PREFIX HAIOnt: <http://.../HAI.owl#>
PREFIX rdf: <http://.../22-rdf-syntax-ns#>
SELECT ?patient ?diagnosis_code
FROM <http://.../input.rdf>
WHERE{
      ?patient rdf:type HAIOnt:Patient .
      ?patient hai:has_diagnosis ?diagnosis .
      ?diagnosis rdf:type hai:Diagnosis .
      ?diagnosis hai:has_diagnosis_code ?diagnosis_code .
}
```

## input.rdf

```
<rdf:RDF xmlns:servOnt="http://.../service-ontology.owl#"
         xmlns:HAIOnt="http://.../HAI.owl#"
         xmlns:rdf="http://.../22-rdf-syntax-ns#">
 <rdf:Description rdf:about="http://.../Patient_by_ID?ID=1">
  <rdf:type rdf:resource="http://.../ont.owl#getDiagnosisIDByPatientID_Input"/>
  <rdf:type rdf:resource="http://.../HAI.owl#Patient"/>
 </rdf:Description>
</rdf:RDF>
```

# Preliminary Evaluation (Cont'd)

Sample input and output:




HYDRA executes the SPARQL query to invoke, orchestrate, and serially execute the combined services

# Summary

- Using Valet SADI, domain experts need not be proficient in SQL or Java programming
- Mapping rules are specified by IT personnel before service generation (a one-time mapping for all services generated)
- Automation of service-code generation saves time and labor (requires approximately 10 seconds per service)
- The architecture is domain-independent (pluggable domain ontology)

# Future Work

- In ongoing work, we are experimenting with increasingly more complex service descriptions and HYDRA queries, and reviewing service generation performance speeds
- We are continuing with the generation of services for the Clinical Intelligence use case implemented in [1] and working with health-care organisations to accelerate HAI surveillance
- Additional trials of **Valet SADI** in non-clinical use cases are being initiated
- Future demonstrations will include a GUI for end-user querying instead of SPARQL

# References

[1] Riazanov A., Klein A., Shaban-Nejad A., Rose G. W., Forster A. J., Buckeridge D. L., and Baker C.J.O.
Semantic querying of relational data for clinical intelligence:a semantic web services-based approach.

[2] Al Manir M.S., Riazanov A., Boley H., and Baker C.J.O.
Generating Semantic Web Services from Declarative Descriptions.

[3] Harold Boley
PSOA RuleML: Integrated Object-Relational Data and Rules.

[4] SPARQL engine for querying SADI Semantic Web services
http://ipsnp.com/hydra/