Using RESTFul API and SHACL to Invoke Executable Semantics In the Context of Core Software Ontology

Author： Xianming Zhang

Presenter: Xianming Zhang ,forzxm@163.com

Aviation Industry Development and Research Center of China

Xianming Zhang is an engineer and researcher of Aviation Industry Development and Research Center of China located in Beijing, China. He received his Degree Master in Information Science from China Aviation Establishment, Degree Bachelor in Computer Science from Tianjin University .

Research Interests

- Applied and Domain Ontology
- Sematic Web
- Semantic Technology
- Knowledge Engineering

content

- Part1 Purpose of This Paper
- Part2 Introduction to Framework: Core Software Ontology
- Part3 Introduction to Framework: SHACL and RESTFul API
- Part4 A Use Case as the Motivation
- Part5 Applying the Framework to the Use Case

Part1 Purpose of This Paper(1)

- Core Software Ontology (CSO) based on the well-known DOLCE Ontology is designed to describe computation domain, including semantic aspects of computing objects(software ,data and their realizations) and execution ,which constitute the executable semantics.
- The initial hope of building software is to launch a computing/execution and return a value, and it is clear that current state of CSO fails to achieve that as the execution situation still cannot be used to invoke the included computing objects.

Part1 Purpose of This Paper(2)

- A typical ontology is based on RDF, and the RDF provides a format basis in which URLs are encapsulated to represent multiple entities. These URLs pointing to static contents can be simply accessed.
- RESTFul APIs often focus on computing, which means that their URLs should not be simply accessed
- It is feasible that fitting RESTFul APIs as URLs into the context of CSO, but executable semantics including these RESTFul APIs is not to be invoked to launch computing in the context of CSO.

Part1 Purpose of This Paper(3)

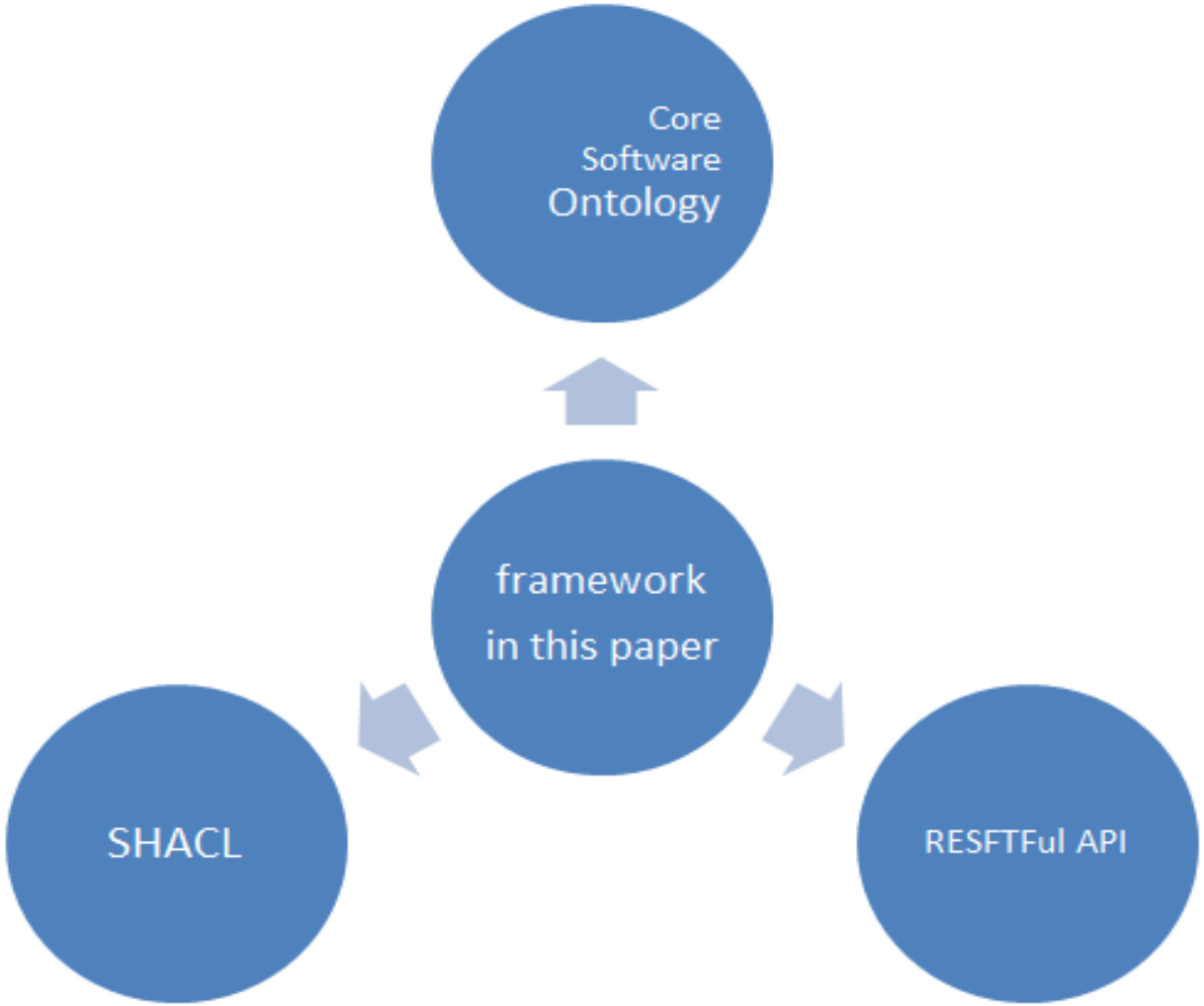
- Shapes Constraint Language (SHACL) provides SHACL-JS engine that can access and invoke JavaScript code on web by its URL. From this perspective, SHACL can help to invoke executable semantic in the context of CSO.
- But JavaScript apparently cannot access to databases and cannot support too complex algorithms such as Matrix or Calculus needed by engineering computing.

Part1 Purpose of This Paper(4)

The purpose of this paper is to present a framework.

- a. Fitting RESTFul API into the context of CSO
- b. Breaking the hurdle between SHACL-JS engine and REST API to enable former to invoke latter to get result,
- c. With a , b and SHACL-SPARQL Construct, REST API can fit into the context of CSO very well and the constructed executable semantics can be invoked for desired purpose.

Part1 Purpose of This Paper(5)



Part2 Introduction to Framework: Core Software Ontology(1)

- This paper puts forward a framework working as basis for invoking executable semantics, in which following ones are cooperating with each other to achieve the goal: Core Software Ontology (CSO), Shapes Constraint Language (SHACL) and RESTFul API.
- A lightweight, easy-to-apply foundational ontology known as DOLCE+DnS Ultralite (DUL)is used as basis for CSO.

Part2 Introduction to Framework: Core Software Ontology(2)

Some concepts from DUL Ontology for this paper are introduced.

- DUL:InformationObject
- DUL: InformationRealization
- DUL: Activity
- DUL: Task
- DUL:Role
- DUL:Plan
- DUL:Situation
- DUL:isRoleOf
- DUL:satisfies
- DUL:executesTask

Part2 Introduction to Framework: Core Software Ontology(3)

Some concepts of CSO.

- CSO:ComputationalObject
- CSO:Data
- CSO:Software
- CSO: ComputationalActivity
- CSO:ComputationalTask
- CSO:Input
- CSO:Output
- CSO:inputFor
- CSO:outputFor

Part2 Introduction to Framework: Core Software Ontology(4)

From DUL Ontology to CSO

CSO	DUL Ontology
CSO:ComputationalObject	DUL: InformationRealization
CSO:Data	DUL:InformationObject
CSO:Software	DUL:InformationObject
CSO:ComputationalActivity	DUL: Activity
CSO:ComputationalTask	DUL: Task
CSO:Input, CSO:Output	DUL:Role
CSO:inputFor,CSO:outputFor	DUL:isRoleOf

Part2 Introduction to Framework: Core Software Ontology(5)

CSO:executes is also introduced to formalize that a CSO:Software is used to complete a CSO:ComputationalTask. The figure below shows the definition(D1)

$$\begin{aligned}&\text{CSO:executes}(x, y) =_{def} \text{CSO:Software}(x) \wedge \text{CSO:ComputationalTask}(y) \wedge \\ &\exists co,ca,p(\text{CSO:ComputationalObject}(co) \wedge \text{CSO:ComputationalActivity}(ca) \wedge \text{DUL:Plan}(p) \wedge \\ &\text{DUL:realizes}(co,x) \wedge \text{DUL:expresses}(x, p) \wedge \text{DUL:defines}(p, y) \wedge \text{DUL: executesTask}(ca, y) \\ &\wedge \text{DUL: hasParticipant}(ca, co)) \quad (D1)\end{aligned}$$

Part 3 Introduction to Framework: SHACL and RESTFul API(1)

- The Shapes Constraint Language (SHACL) is a W3C recommendation.
- SHACL-SPARQL is one extension mechanism for SHACL to express constraints in SPARQL, allowing shape definitions to point at executable SPARQL queries to perform the validations .
- SHACL-JS engine is an advanced extension mechanism for SHACL, allowing users to express SHACL constraints and the advanced features of custom targets, functions and rules with the help of JavaScript.

Part 3 Introduction to Framework: SHACL and RESTFul API(2)

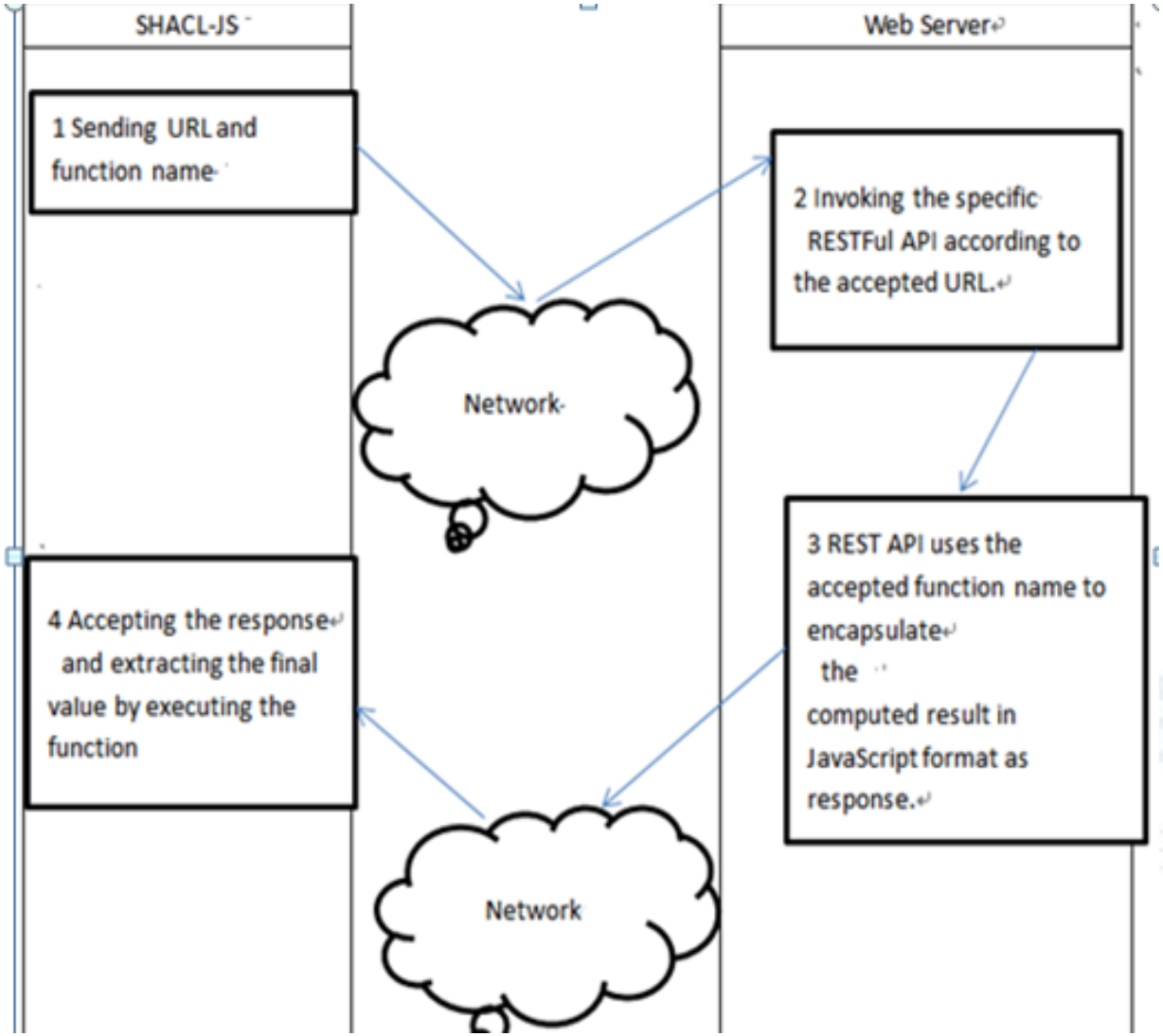
- A RESTFul API is an application program interface (API) that uses existing HTTP methodologies defined by the RFC 2616 protocol, which allows two software programs to communicate with each other.

Part 3 Introduction to Framework: SHACL and RESTFul API(3)

A computer with SHACL-JS is called as client and a computer with RESTFul API running on it is called as Web Server.

- A user operating the client submits data in RDF that contain URL of a RESTFul API and function name into SHACL-JS engine, SHACL-JS engine sends the URL to Web Server after parsing.
- Web Server runs the received URL and then gets the returned value.
- Web Server encapsulates the value as the string in JavaScript code format ,and then feedbacks to the client.
- The client simply runs the string ,and returns the value to user.

Part 3 Introduction to Framework: SHACL and RESTFul API(4)

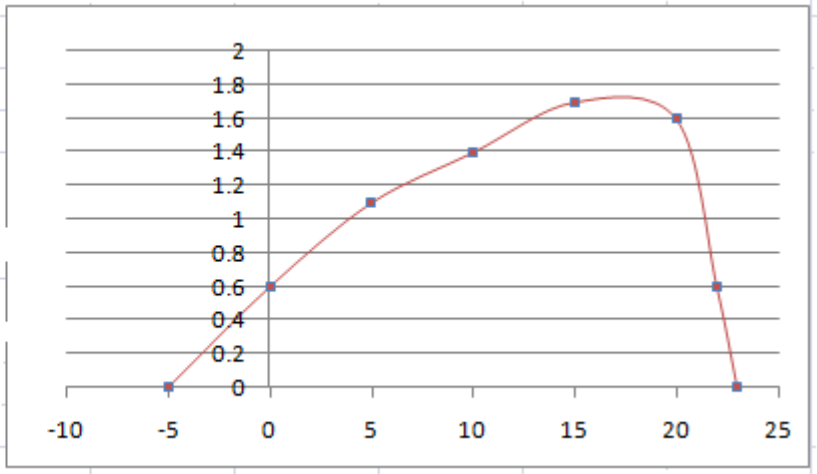


Part4 A Use Case as the Motivation(1)

- This use case is to computing lift coefficient of airfoil; the lift-coefficient formula is as follows.

  Lift-coefficient=f(attack-angle)

Attack Angle	Lift Coefficient
-5	0
0	0.6
5	1.1
10	1.4
15	1.7
20	1.6
22	0.6
23	0



Part4 A Use Case as the Motivation(2)

- There is no explicit formula (or calculation script) to accurately calculate lift coefficient from attack angle. Typically, lift-coefficient is a list of data through a limited number of experiments that record the data under different attack angles.
- A dedicated RESTFul API can be developed and deployed on the web server. The RESTFul API implements numerical approximation, such as least square and interpolating to allow users to query for any given attack-angle.

Part5 Applying the Framework to the Use Case (1)

- The framework is applied to the use case, and we can freely get returned value of lift coefficient for a given attack angle. In this paper, a temporary ontology is created in the context of CSO for this use case, known as S-C Ontology.

Part5 Applying the Framework to the Use Case (2)

S-C Ontology(1)

A few of ground entities can be extracted from this use case and be aligned to predefined concepts in CSO.

S-C Ontology	CSO	memo
s-c: attack-angle	CSO:Data	
s-c:lift-coefficient	CSO:Data	
s-c:computation1	CSO:Software	
s-c:computation1 -computational-object	CSO: ComputationalObject	DUL:realizes s-c:computation1 and rdfs:seeAlso “http://server:8080/lift-coefficient?attack-angle={value}”

Part5 Applying the Framework to the Use Case (3)

S-C Ontology(2)

S-C Ontology	CSO/DUL Ontology	memo
s-c:activity1	CSO:ComputationalActivity	The execution of a certain CSO:ComputationalObject/CSO:Software
s-c:run	DUL:hasParticipan	Linking a CSO: ComputationalActivity with a CSO: ComputationalObject ,which means giving rise to execution of software.
s-c:in/ s-c:out	DUL:hasParticipant	Linking a CSO:ComputationalActivity with a CSO:Data as input/output for computing.

Part5 Applying the Framework to the Use Case (4)

S-C Ontology(3)

Modeling Plan and Situation in S-C ontology

S-C Ontology	CSO/DUL Ontology	memo
s-c:computation-plan	DUL:Plan	the design of computing configuration
s-c:input-attack-angle	CSO:Input	
s-c:output-lift-angle	CSO:Output	
s-c:task1	CSO:ComputationalTask	
s-c:computation-situation	DUL:Situation	The computing configuration

Part5 Applying the Framework to the Use Case (5)

S-C Ontology(4)

The triples in s-c:computation-plan

s-c:computation-plan
s-c:input-attack-angle CSO:inputFor s-c:task1 s-c:output-lift-angle CSO:inputFor s-c:task1

Part5 Applying the Framework to the Use Case (6)

S-C Ontology(5)

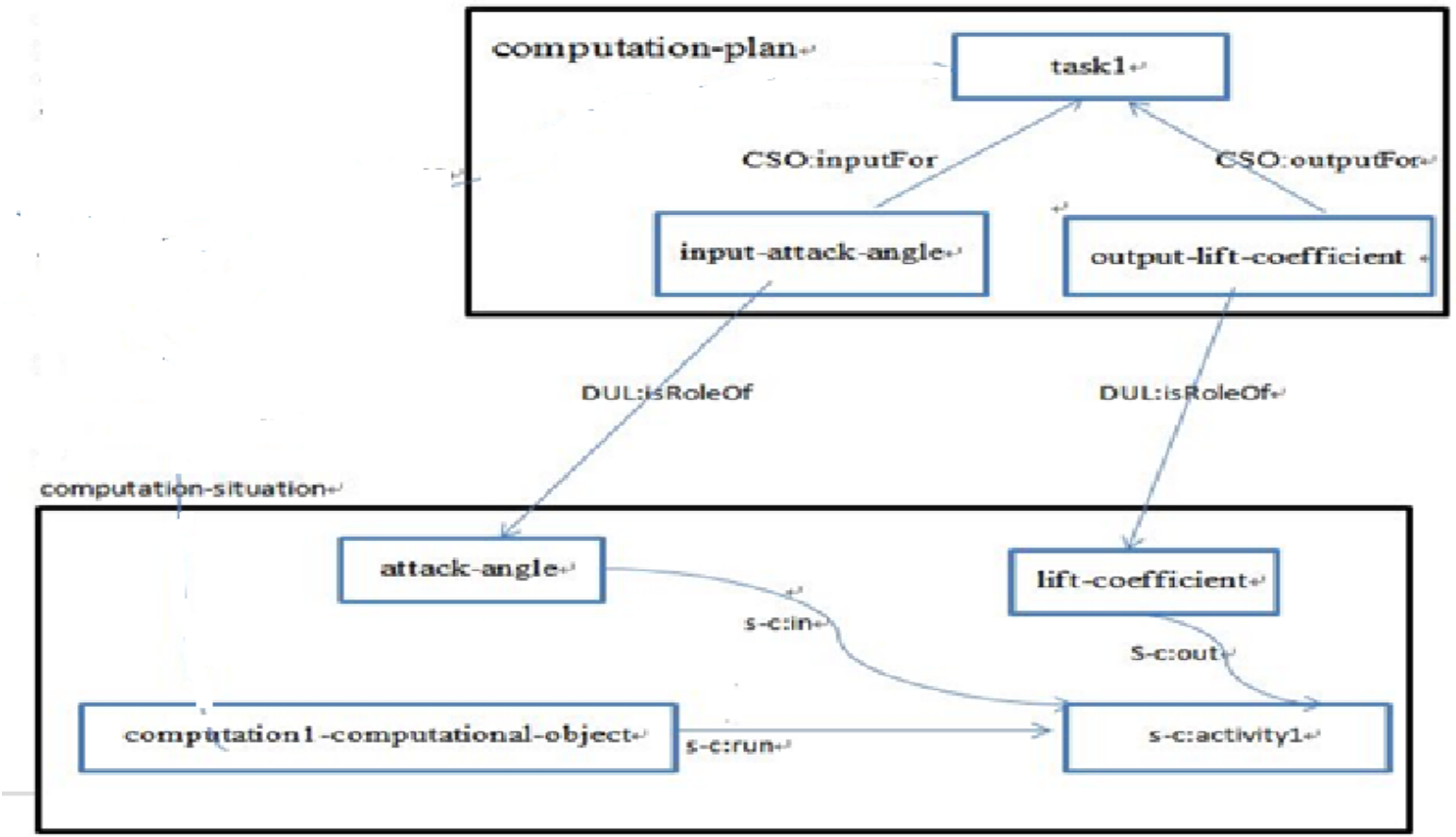
The triples in s-c:computation-situation

s-c:computation-situation
s-c:attack-angle s-c:in s-c:activity1 s-c:lift-coefficient s-c:out s-c:activity1 s-c:computation1-computational-object s-c:run s-c:activity1

Part5 Applying the Framework to the Use Case (7)

S-C Ontology(6)

The associations between s-c:computation-plan and s-c:computation-situation.



Part5 Applying the Framework to the Use Case (8)

To Inference Situation from Plan(1)

In this paper ,the CONSTRUCT clause of SPARQL Update is used to construct s-c:computation-situation from s-c:computation-plan with associated CSO:Data and CSO:Software.

Part5 Applying the Framework to the Use Case (9)

To Inference Situation from Plan(2)

```
construct {
  ?software_computational_object s-c:run s-c:activity1.
  ?inData s-c:in  s-c:activity1.
  ?inData DUL:hasDataValue ?value.
  ?outData s-c:out s-c:activity1.
  ?software_computational_object rdfs:seeAlso ?url.
 }
where {
 GRAPH <http://semantic-computing/#computation-plan>{
          ?inRole cso:input-for  s-c:task1.
                    ?outRole cso:output-for  s-c:task1.
}
 GRAPH <http://semantic-computing/#software-plan>{
   ?software cso:executes s-c:task1.
   ?inData DUL:hasRole  ?inRole.
          ?outData DUL:hasRole ?outRole.}
  GRAPH <http://semantic-computing/#software-data>{
   ?software_computational_object DUL:realizes ?software.
          ?software_computational_object rdfs:seeAlso ?url.
   ?inData DUL:hasDataValue ?value. }
 }
```

Part5 Applying the Framework to the Use Case (10)

- To Inference Situation from Plan(3)

There are several graphs in this statement: <http://semantic-computing/#computation-plan> contains triples of s-c:computation-plan; <http://semantic-computing/#software-plan> contains triples representing associations between CSO:Data/CSO:Software and entities in s-c:computation-plan; <http://semantic-computing/#software-data> contains triples representing associations among CSO:Data/CSO:software. The constructed result is below, it is noted the s-c:computation1-computational-object(an instance of CSO: ComputationalObject) links a RESTFul API via rdfs:seeAlso .

Part5 Applying the Framework to the Use Case (11)

To Inference Situation from Plan(4)

The constructed situation :

s-c:computation1-computational-object s-c:run s-c:activity1.

s-c:attack-angle s-c:in s-c:activity1.

s-c:attack-angle DUL:hasDataValue 13.0.

s-c:lift-coefficient s-c:out s-c:activity1.

s-c:computation1-computational-object rdfs:seeAlso <http://ip/lift-coefficient?attack-angle=>.

Part5 Applying the Framework to the Use Case (12)

Using SHACL to Invoke Executable Semantics(1)

- The s-c: computation-situation forms executable semantics, and now its goal is to create a triple of " s-c:lift-coefficient DUL:hasDataValue ?value",
- Using the triples of s-c:computation-situation to create a model named as Shape-Function, triples of which meet standard of SHACL-JS to form a function that will be further used to invoke RESTFul API.

Part5 Applying the Framework to the Use Case (13)

Using SHACL to Invoke Executable Semantics(2)

The SPARQL Construct statement to use

```
prefix sh: <http://www.w3.org/ns/shacl#>
construct {
s-c:dynamicFunc sh:jsFunctionName "dynamicFunc".
s-c:dynamicFunc sh:jsLibrary <http://jsLibrary/temp>.
s-c:dynamicFunc sh:returnType xsd:double.
s-c:dynamicFunc sh:parameter <http://parameter/temp>.
s-c:dynamicFunc rdf:type sh:JSFunction.
<http://parameter/temp> sh:datatype xsd:double.
<http://parameter/temp> sh:path s-c:number.
<http://jsLibrary/temp> sh:jsLibraryURL ?dynamicFuncURL.
}
where{
  ?software_computational_object s-c:run s-c:activity1.
  ?inData s-c:in  s-c:activity1.
  ?inData DUL:hasDataValue ?value.
  ?software_computational_object rdfs:seeAlso ?url.
  BIND(('http://IP/RESTTemplate/access?url='+STR(?url)+
'?value='+STR(?value)) as ?dynamicFuncURL).

}
```

Part5 Applying the Framework to the Use Case (14)

Using SHACL to Invoke Executable Semantics(3)

The constructed model named as Shape-Function, also can be regarded as a shape according to SHACL. It must be noted that there are two RESTFul APIs should be discussed here, they are 'http://ip/RESTTemplate/access?url=" and "http://ip/lift-coefficient?attack-angle=", the function of former is to invoke the latter and encapsulate the return value in JavaScript format with "dynamicFunc" as function name, below is the concise code.

Part5 Applying the Framework to the Use Case (15)

Using SHACL to Invoke Executable Semantics(4)

```
s-c:dynamicFunc sh:jsFunctionName "dynamicFunc".
s-c:dynamicFunc sh:jsLibrary <http://jsLibrary/temp>.
s-c:dynamicFunc sh:returnType xsd:double.
s-c:dynamicFunc sh:parameter <http://parameter/temp>.
s-c:dynamicFunc rdf:type sh:JSFunction.
<http://parameter/temp> sh:datatype xsd:double.
<http://parameter/temp> sh:path s-c:number.

<http://jsLibrary/temp>
sh:jsLibraryURL 'http://ip/RESTTemplate/access?url=http://ip/lift-coefficient?attack-angle=13'.
```

Part5 Applying the Framework to the Use Case (16)

Using SHACL to Invoke Executable Semantics(5)

In addition to the Shape-Function complied with SHACL-JS , another model named as Shape –Construct complied with SHACL-SPARQL is needed to work with the Model-Function the achieve the goal.

Part5 Applying the Framework to the Use Case (17)

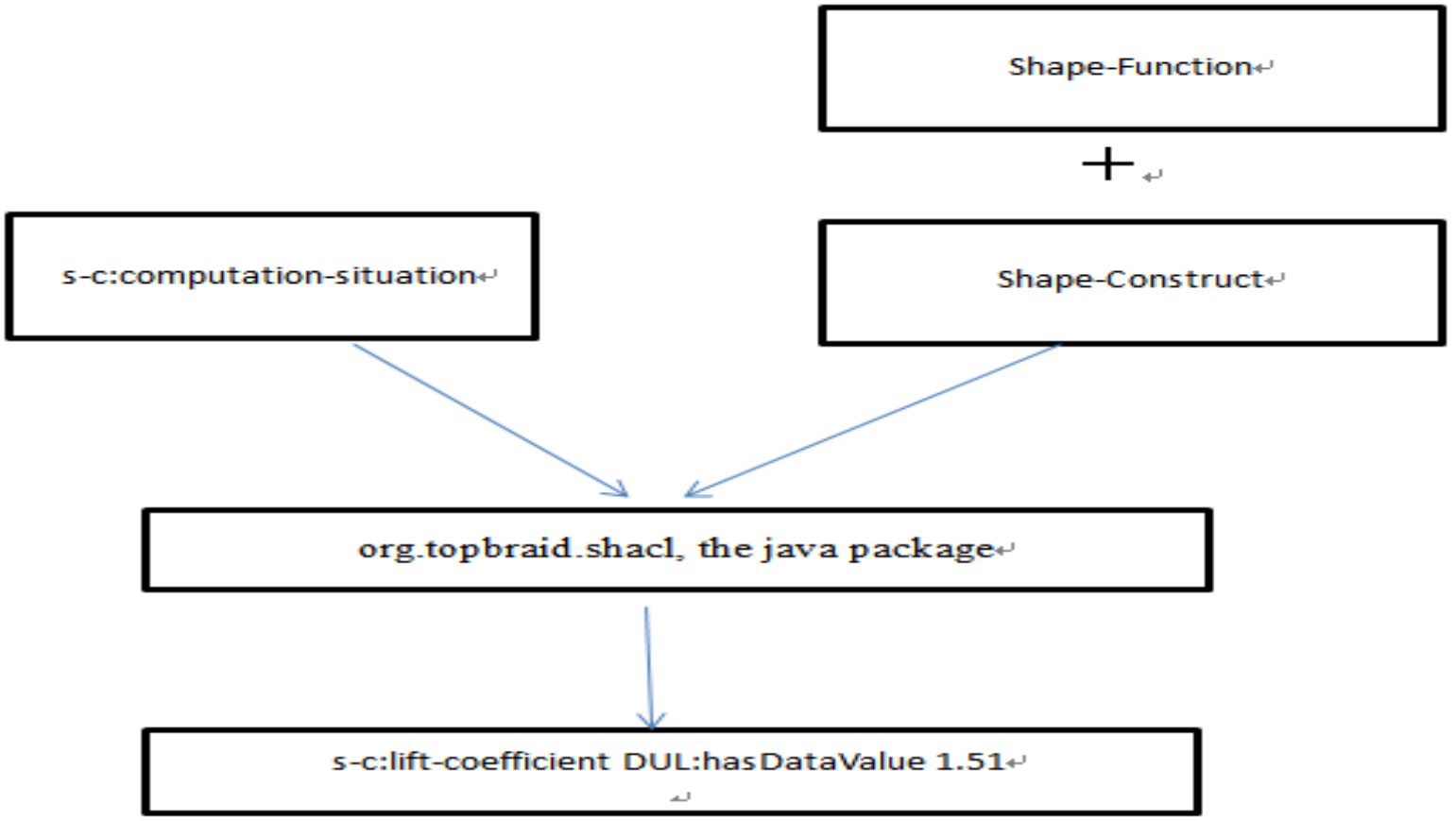
Using SHACL to Invoke Executable Semantics(6)

```
@prefix s-c:<http://semantic-computing/#>.
s-c:rule1
  a rdfs:Class, sh:NodeShape ;
  sh:targetNode s-c:activity1 ;
  rdfs:label "to run sc:activity1" ;
  sh:rule[
      a sh:SPARQLRule ;
          sh:construct """
                        CONSTRUCT {
                        ?outdata DUL:hasDataValue  ?value.
          }
                        WHERE {
                          ?outdata  s-c:out ?this.
                    BIND(<http://semantic-computing/#dynamicFunc>()
AS ?value).
                        }
                        """ ;
  ] .
```

Part5 Applying the Framework to the Use Case (18)

Using SHACL to Invoke Executable Semantics(7)

The process and method of creating the new triple



Thank you !!!