Semantic Web for Extending and Linking Formalisms

Jin Song Dong Jing Sun Hai Wang

School of Computing,
National University of Singapore,
`dongjs,sunjing,wangh@comp.nus.edu.sg`

Abstract. The diversity of various formal specification techniques and the need for their effective combinations requires an extensible and integrated supporting environment. The Web provides infrastructure for such an environment for formal specification and design because it allows sharing of various design models and provides hyper textual links among the models. Recently the Semantic Web Activity proposed the idea of having data on the web defined and linked in a way that it can be used for automation, extension and integration. The success of the Semantic Web may have profound impact on the web environment for formal specifications, especially for extending and integrating different formalisms. This paper demonstrates how RDF and DAML can be used to build a Semantic Web environment for supporting, extending and integrating various formal specification languages. Furthermore, the paper illustrates how RDF query techniques can facilitate specification comprehension.

keywords: specification environment, Semantic Web

1 Introduction

Many formal specification techniques exist for modeling different aspects of software systems and it's difficult to find a single notation that can model all functionalities of a complex system [21, 34]. For instance, B/VDM/Z are designed for modeling system data and states, while CSP/CCS/π-calculus are designed for modeling system behaviour and interactions. Various formal notations are often extended and combined for modeling large and complex systems. In recent years, *formal methods integration* has been a popular research topic [1, 13]. In the context of combining state-based and event-based formalisms, a number of proposals have been presented [8, 11, 12, 19, 26, 28, 29, 32]. Our general observations on these works are that

> Various formal notations can be used in an effective combination if the semantic links between those notations can be clearly established. The semantic/syntax integration of those languages would be a consequence

when the semantic links are precisely defined. Due to different motivations, there are possible different semantic links between two formalisms, which lead to different integrations between the two.

Unlike UML,an industrial effort for standardising diagrammatic notations, a single dominating integrated formal method may not exist in the near future. The reason may be partially due to the fact that there are many different well established individual schools, e.g., VDM forum, Z/B users, CSP group, CCS/π-calculus family and etc. Another reason may be due to the open nature of the research community, i.e. FME (www.fmeurope.org), which is different from the industrial 'globalisation' community, i.e. OMG (www.omg.org).

Regardless of whether there will be or there should be an ultimate integrated formal method (like UML), *diversity* seems to be the current reality for formal methods and their integrations. Such a diversity may have an advantage, that is, different formal methods and their combinations may be effective for developing various kinds of complex systems[1]. The best way to support and popularise formal methods and their effective combinations is to build a widely accessible, extensible and integrated environment.

The World Wide Web provides an important infrastructure for a promising environment for various formal specification and design activities because it allows sharing of various design models and provides hyper textual links among the models. Recently the Semantic Web Activity [2] proposed the idea of having data on the web defined and linked in a way that it can be used for automation and integration. The success of the Semantic Web may have profound impact on the web environment for formal specifications, especially for extending and integrating different formal notations. This paper demonstrates an approach on how RDF [18] and DAML [30] can be used to build a Semantic Web environment for supporting, checking, extending and integrating various formal specification languages. Furthermore, based on this Semantic Web environment, specification comprehension (queries for review/understanding purpose) can be supported.

The reminder of the paper is organised as follows. Section 2 firstly introduces RDF/DAML and demonstrates how DAML environment can be built for formal specification languages such as Z and CSP[2]. Then it illustrates how the DAML environments for Z and CSP can be extended for Object-Z and TCSP. Section 3 demonstrates how various integration approaches for combining Object-Z and (T)CSP can be supported by the Semantic Web environment. Section 4 illustrates how specification comprehension can be supported by RDF queries. Section 5 discusses related work and concludes the paper.

---

[1] In fact, one of the difficult tasks of OMG is to resist many good new proposals for extending UML — a clear consequence and drawback of pushing a single language for modeling all software systems.

[2] Specific formal notations used in this paper are mainly for demonstrating the ideas, other formalisms can also be supported.

2 Semantic Web for formal specifications

Following the success of XML [31], W3C's primary focus is on Semantic Web. Currently, one of the major Semantic Web activities at W3C is the work on Resource Description Framework (RDF) [18] and the DARPA Agent Markup Language (DAML) [30].

RDF is a foundation for processing metadata; it provides interoperability between applications that exchange machine-understandable information on the Web. RDF uses XML to exchange descriptions of Web resources and emphasises facilities to enable automated processing. In fact, the RDF descriptions provide a simple ontology system to support the exchange of knowledge and semantic information on the Web. RDF Schema [7] provide the basic vocabulary to describe RDF vocabularies. RDF Schema can be used to define properties and types of the web resources. Similar to XML Schema which give specific constraints on the structure of an XML document, RDF Schema provide information about the interpretation of the RDF statements. DAML is a semantic markup language based on RDF and XML for Web resources. DAML currently combines Ontology Interchange Language (OIL) and features from other ontology systems.

In this section we use Z [33] and CSP [15] as examples to demonstrate how a Semantic Web environment for formal specification languages can be developed. These environments can be further extended and integrated.

2.1 Semantic Web environment — RDFS/DAML for Z

Firstly, a customised RDFS/DAML definition for Z language is developed according to its syntax and static semantics. This definition (a DAML ontology itself) provides information about the interpretation of the statements given in a Z-RDF instant data model. Part of the RDFS definitions (for constructing a Z schema) is as follows:

```
<rdf:RDF
  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd = "http://www.w3.org/2000/10/XMLSchema#"
  xmlns:daml = "http://www.daml.org/2001/03/daml+oil#"
  xmlns:z = "http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z#">

<!-- some definition omitted -->

  <rdfs:Class rdf:ID="Schemadef">
    <rdfs:label>Schemadef</rdfs:label>
   </rdfs:Class>
  <rdfs:Class rdf:ID="Schemabox">
    <rdfs:label>Schemabox</rdfs:label>
      <rdfs:subClassOf rdf:resource="#Schemadef"/>
      <rdfs:subClassOf>
```

```
      <daml:Restriction daml:cardinalityQ="1">
        <daml:onProperty rdf:resource="#name"/>
      </daml:Restriction> </rdfs:subClassOf>
    <rdfs:subClassOf>
      <daml:Restriction daml:minCardinality="0">
        <daml:onProperty rdf:resource="#del"/>
        <daml:toClass rdf:resource="#Schemadef"/>
      </daml:Restriction> </rdfs:subClassOf>

    <!-- some definition omitted -->
    <rdfs:subClassOf>
      <daml:Restriction daml:minCardinality="0">
        <daml:onProperty rdf:resource="#decl"/>
      </daml:Restriction> </rdfs:subClassOf>
    <rdfs:subClassOf>
      <daml:Restriction daml:minCardinality="0">
        <daml:onProperty rdf:resource="#predicate"/>
      </daml:Restriction>
    </rdfs:subClassOf>
</rdfs:Class>
```

(note that `xmlns` stands for XML name space)

The DAML class `Schemadef` represents the Z schemas. The class `Schemabox`, a subclasses of `Schemadef`, represents the Z schemas defined in schema box form. The class `Schemabox` models a type whose instance may consist of a `name`, a number of declarations `decl` and some `predicate` definitions. In addition, a `Schemabox` instance may also have zero or more properties `del` whose value must be another `Schemadef` instance (for capturing the Z Δ-convention). As the paper focuses on demonstrating the approach, other Semantic Web environments for Z constructs are left out but can be found at:

`http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z.daml`

Under the Semantic Web environment for the Z language, Z specifications as RDF instant files can be edited (by any XML editing tool).

The Z notation contains a rich set of mathematical symbols. Those symbols can be presented directly in Unicode which is supported by RDF (XML). A set of entity declarations is defined to map those Z symbols to their Unicode correspondents (with a Z LaTeX compatible name) as follows.

```
<!ENTITY cat "&#x2040;">
<!ENTITY mem "&#x2208;">
<!ENTITY uni "&#x222a;">
```

One benefit of using Unicode is for visualisation purposes, for example, we have developed an XSLT program (http://nt-appn.comp.nus.edu.sg/fm/zdaml/rdf2zml.xsl) to transform the RDF environment into ZML [27][3], an XML environment for display/browsing Z on the web directly (using the IE web browser).

The following is a simple *Buffer* schema and a *Join* operation.

$[MSG]$

Buffer

$max : \mathbb{Z}$
$items : \operatorname{seq} MSG$

$\# items \leq max$

Join

$\Delta Buffer$
$i? : MSG$

$\# items < max \land$
$items' = \langle i? \rangle \frown items \land$
$max' = max$

The corresponding RDF definition is as following.

```
<z:Type rdf:ID="msg">
  <z:type>MSG</z:type>
</z:Type>
<z:Schemabox rdf:ID="buffer">
  <z:name>Buffer</z:name>
    <z:decl> <z:Decl z:name="max" z:dtype="&integer;"/> </z:decl>
    <z:decl> <z:Decl z:name="items" z:dtype="&seq; MSG"/> </z:decl>
    <z:predicate> #items &leq; max </z:predicate>
</z:Schemabox>
<z:Schemabox rdf:ID="join">
  <z:name>Join</z:name>
  <z:del rdf:resource="#buffer"/>
  <z:decl> <z:Decl z:name="i?" z:dtype="MSG"/> </z:decl>
  <z:predicate>#items &lt; max  &land;
    items'=  {i?} &cat; items &land; max' = max </z:predicate>
</z:Schemabox>
```

Note that the RDF file is in XML format which can be edited by XML editing tools, i.e. XMLSpy. Alternatively, this RDF specification can be treated as an interchange format which can be generated from ZML via our XSL tool or from Latex (a tool is in the development stage).

2.2 Semantic Web environment — RDFS/DAML for CSP

Similarly a Semantic Web environment for CSP can be constructed based on its definition. Part of the RDFS/DAML definitions (for constructing a CSP process) is as follows:

[3] Our previous work, ZML, was developed mainly for visualising Z on the web and tranforming Object-Z to UML(XMI) [27].

```
<!-- some definition omitted -->
    <rdfs:Class rdf:ID="Event">
        <rdfs:label>Event</rdfs:label> </rdfs:Class>
    <rdfs:Class rdf:ID="Process">
        <rdfs:label>Process</rdfs:label> </rdfs:Class>

    <rdfs:Class rdf:ID="Simevent">
        <rdfs:label>SimpleEvent</rdfs:label>
        <!-- some definition omitted -->
    </rdfs:Class>

    <rdfs:Class rdf:ID="Communication">
        <rdfs:label>Communication</rdfs:label>
        <rdfs:subClassOf rdf:resource="#Event"/>
        <!-- some definition omitted -->
    </rdfs:Class>

    <!--STOP process-->
    <rdfs:Class rdf:ID="Stop">
        <rdfs:label>STOP</rdfs:label>
        <rdfs:subClassOf rdf:resource="#Process"/>
    </rdfs:Class>
    <!--prefix process-->
    <rdfs:Class rdf:ID="PrefixPro">
        <rdfs:label>prefixPro</rdfs:label>
        <rdfs:subClassOf rdf:resource="#Process"/>
        <rdfs:subClassOf>
            <daml:Restriction>
                <daml:onProperty rdf:resource="#prefix"/>
                <daml:toClass rdf:resource="#Event"/>
            </daml:Restriction> </rdfs:subClassOf>
        <rdfs:subClassOf>
            <daml:Restriction>
                <daml:onProperty rdf:resource="#toProc"/>
                <daml:toClass rdf:resource="#Process"/>
            </daml:Restriction> </rdfs:subClassOf>
    </rdfs:Class>
```

It states that there are two major kinds of constructs in CSP, events and processes. Events can be classified into simple ones and communications containing channels and messages. Processes can be classified into various forms including a special event `STOP`, prefix, sequential etc.

The main contribution of these Semantic Web environments is that they provide formal specifications on the web together with additional semantic information. Furthermore, they facilitate web browsing, collaborative formal design and some static semantics checking. For instance, given two CSP processes P_1

and P_2, the following incorrect CSP expression

$$P_1 \to P_2$$

will be detected by the CSP Semantic Web environment via RDF validator. In this paper we will focus on how these environments can be easily extended and integrated to form new environments for the extension and combination of formalisms.

2.3 Extending Z to Object-Z

Object-Z [10, 25] is an object-oriented extension to Z. A Z specification defines a number of state and operation schemas. In contrast, Object-Z associates individual operations with one state schema. The collective definition of a state schema with its associated operations constitutes the definition of a class. Each class has one state schema, at most one initial schema and number of operation schema. The state schema can be viewed as a nameless Z schema. The initial schema can be viewed as a Z schema which only contains some predicate properties. The following demonstrates parts of the Semantic Web environment for Object-Z. It extends the Z's environment.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema#"
xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
xmlns:oz="http://nt-appn.comp.nus.edu.sg/fm/zdaml/OZ#"
xmlns:z="http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z#"
xmlns="http://nt-appn.comp.nus.edu.sg/fm/zdaml/OZ#">
    <daml:Ontology rdf:about="">
        <daml:imports rdf:resource=
              "http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z"/>
    </daml:Ontology>
    <rdfs:Class rdf:ID="State">
        <rdfs:label>State</rdfs:label>
        <rdfs:subClassOf rdf:resource="z:Schemabox"/>
        <rdfs:subClassOf>
            <daml:Restriction>
                <daml:onProperty rdf:resource="z:name"/>
                <daml:hasValue>
                    <xsd:string rdf:value=""/> </daml:hasValue>
            </daml:Restriction> </rdfs:subClassOf>
    </rdfs:Class>
    <rdfs:Class rdf:ID="Init">
        <rdfs:label>INIT</rdfs:label>
```

```
    <!-- some definition omitted -->
</rdfs:Class>
<rdfs:Class rdf:ID="OP">
    <rdfs:label>OP</rdfs:label>
    <!-- some definition omitted -->
</rdfs:Class>

<rdfs:Class rdf:ID="Message">
    <rdfs:label>Message</rdfs:label>
    <rdfs:subClassOf rdf:resource="#OP"/>
    <rdfs:subClassOf>
        <daml:Restriction daml:cardinality="1">
            <daml:onProperty rdf:resource="oz:receiver"/>
        </daml:Restriction> </rdfs:subClassOf>
    <rdfs:subClassOf>
        <daml:Restriction daml:cardinality="1">
            <daml:onProperty rdf:resource="#method"/>
            <daml:toClass rdf:resource="#OP"/>
        </daml:Restriction> </rdfs:subClassOf>
</rdfs:Class>

<!-- some definition omitted -->
<rdfs:Class rdf:ID="Classdef"/>

<rdfs:Class rdf:ID="Classdef1">
    <rdfs:label>Classdef1</rdfs:label>
    <rdfs:subClassOf rdf:resource="#Classdef"/>
    <rdfs:subClassOf>
        <daml:Restriction daml:cardinality="1">
            <daml:onProperty rdf:resource="z:name"/>
        </daml:Restriction> </rdfs:subClassOf>
<!-- some definition omitted -->
    <rdfs:subClassOf>
        <daml:Restriction>
            <daml:maxCardinality>1</daml:maxCardinality>
            <daml:onProperty rdf:resource="#state"/>
            <daml:toClass rdf:resource="#State"/>
        </daml:Restriction> </rdfs:subClassOf>
    <rdfs:subClassOf>
        <daml:Restriction>
            <daml:onProperty rdf:resource="#op"/>
            <daml:toClass rdf:resource="#OP"/>
        </daml:Restriction> </rdfs:subClassOf>
</rdfs:Class>
<daml:DatatypeProperty rdf:ID="delObj">
```

```
        <rdfs:range rdf:resource=
         "http://www.w3.org/2000/10/XMLSchema#string"/>
    </daml:DatatypeProperty>
</rdf:RDF>
```

This Object-Z Semantic Web environment imports the definition of Z. Note that `Message` class is used to define message passing. It consists of a `receiver` property (object reference) and a `method` property (the operation of the declared class of the receiver).

A `Classdef1` class (an Object-Z class defined by a class box) was defined to have the following properties.

- a `name` property,
- a `state` property whose value must be a `State` class object,
- some `op` properties which value must be `OP` class object etc.

The `State` class is a subclass of `Schemabox` (class for a Z schema defined in schema box form). That is a `State` object is a special `Schemadef` object satisfying the restriction that the `name` property has no value. The `OP` class is the same as class `Schemadef` (for Z schema) except a new property `delObj` was added to it. This is due to the difference between the semantic requirements of Δ list in Z and Object-Z. In Z the entity following Δ is the name of state schema name, and in Object-Z the entity following the Δ are variables defined in the class state schema.

Consider the buffer example in Object-Z:

Buffer

(state schema)
$max : \mathbb{N}$
$items : \mathrm{seq}\, MSG$

$\# items \leq max$

INIT
$items = \langle\, \rangle$

Join
$\Delta(items)$
$i? : MSG$

$\# items < max$
$items' = \langle i? \rangle \frown items$

Leave
$\Delta(items)$
$i! : MSG$

$\# items \neq 0$
$items = items' \frown \langle i! \rangle$

under the Semantic Web environment this Buffer class can be edited as the following RDF file.

```
  <oz:Classdef1 rdf:ID="buffer">
    <z:name>Buffer</z:name>
      <oz:state>
        <oz:State>
```

```
        <z:decl> <z:Decl z:name="max" z:dtype="&integer;"/> </z:decl>
        <z:decl> <z:Decl z:name="items" z:dtype="&seq; MSG"/> </z:decl>
        <z:predicate>#items &leq; max</z:predicate>
      </oz:State>
   </oz:state>
    <!-- some definition omitted -->
   <oz:op><oz:OP rdf:ID="join">
      <z:name>Join</z:name>
      <oz:delObj> items</oz:delObj>
      <z:decl> <z:Decl z:name="i?" z:dtype="MSG"/> </z:decl>
      <z:predicate>#items &lt; max  &land;
                   items'=  {i?} &cat; items </z:predicate>
   </oz:OP></oz:op>
</oz:Classdef1>
```

2.4 Extending CSP to TCSP

The extension from CSP to TCSP can be achieved in a similar way. The following is part of the Semantic Web environment for TCSP.

```
<?xml version="1.0" encoding="UTF-8"?> <rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema#"
xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
xmlns:tcsp="http://nt-appn.comp.nus.edu.sg/fm/zdaml/TCSP#"
xmlns:csp="http://nt-appn.comp.nus.edu.sg/fm/zdaml/CSP#"
xmlns="http://nt-appn.comp.nus.edu.sg/fm/zdaml/TCSP#">
    <daml:Ontology rdf:about="">
        <daml:imports rdf:resource=
         "http://nt-appn.comp.nus.edu.sg/fm/zdaml/CSP"/>
    </daml:Ontology>
    <!--timed event-->
    <rdfs:Class rdf:about="csp:Event">
        <rdfs:subClassOf>
            <daml:Restriction daml:minCardinality="0">
                <daml:onProperty rdf:resource="#etime"/>
            </daml:Restriction> </rdfs:subClassOf>
    </rdfs:Class>
    <daml:DatatypeProperty rdf:ID="etime">
        <rdfs:range rdf:resource=
        "http://www.w3.org/2000/10/XMLSchema#string"/>
    </daml:DatatypeProperty>
    <!--Wait process-->
    <rdfs:Class rdf:ID="Wait">
        <rdfs:label>WAIT</rdfs:label>
```

```
        <rdfs:subClassOf rdf:resource="#process"/>
        <daml:Restriction daml:minCardinality="0">
            <daml:onProperty rdf:resource="#etime"/>
        </daml:Restriction> </rdfs:Class>
    <!-- some definition omitted -->
```

This TCSP environment is derived by first importing the definition of CSP, and then defining a new property `etime` for the events in CSP. The property `etime` shows the time of occurrence of events. Several new types of process are also defined. For example, the WAIT process is just a subclass of a general process.

One interesting point is that the physical size or number of 'subclass' clauses in the DAML file (above) may provide an indication of the degree of extension (how much modification and extension has been developed in the new language). Such a concrete number or ratio may give us some quantitive comparison, perhaps indicating how new (or faithful) is Object-Z relative to Z, TCSP to CSP or VDM++ to VDM.

In the next section, we will focus on the essential part of this paper – the use of the Semantic Web for linking formalisms.

3 Semantic Web for linking formalisms

Various modeling methods can be used in an effective combination for designing complex systems if the semantic links between those methods can be clearly established and defined. Given two sets of formalisms, say state-based ones and event-based ones, it's not too surprising to see that different possible integrations are more than the cross-product of the two sets. This is simply because the different semantic links between the two formalisms lead to different integrations. Furthermore, the semantic links can be directional and bi-directional.

Let's consider the case of linking Object-Z and CSP. Smith and Derrick's approach [26] is to identify Object-Z operations with CSP channel/events and Object-Z classes with CSP processes. The CSP-OZ approach taken by Fischer and Wehrheim [11] is similar to Smith and Derrick's approach except that it divides each Object-Z operation into two separate operations (enable and effect events). The TCOZ approach [19] identifies Object-Z operations with CSP processes[4].

Despite the differences, all those integrations are useful for modeling different kinds of complex systems. For example, Smith and Derrick's approach is good at modeling a system with a group of simple passive components and complex concurrent interactions (at a system level) between those components. On the other hand, TCOZ is good at modeling system with complex components which may have their own thread of control and support multi-layer compositions and concurrency.

In this paper, we will demonstrate how the Object-Z and (T)CSP Semantic Web environments can be linked to support Smith/Derrick and TCOZ approaches.

[4] TCOZ is an integration of Object-Z and TCSP[24].

3.1 *class* $\Longrightarrow$ *process*

In Smith/Derrick's approach [26], Object-Z classes are modeled as CSP processes and the Object-Z operations are modeled as CSP events. The event corresponding to an operation is a communication event with the operation name as the channel and the mapping from its parameters to their values as the value passed on that channel. In this approach any two operations with the same name and parameters will be modelled by identical events when their parameters have same values and hence will be able to synchronize. There are two main phases in specifying a concurrent system.

- The first phase is to decompose the complex system into components and specify each of these components using Object-Z.
- The second phase involves the specification of the system using CSP operators.

Consider the specification of two communicating buffers, the following model demonstrates this approach:

$$Buffer_1 \mathrel{\widehat=} Buffer[Transfer/Leave]$$
$$Buffer_2 \mathrel{\widehat=} Buffer[Transfer/Join]$$
$$System \mathrel{\widehat=} Buffer_1 \,|[\, Transfer \,]|\, Buffer_2$$

where the two buffers ($Buffer_1$ and $Buffer_2$) communicate through channel *Transfer*.

The semantic environment for this approach can be achieved in the following way:

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema#"
xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
xmlns:oz="http://nt-appn.comp.nus.edu.sg/fm/zdaml/OZ#"
xmlns:csp="http://nt-appn.comp.nus.edu.sg/fm/zdaml/CSP#"
xmlns:app1="http://nt-appn.comp.nus.edu.sg/fm/zdaml/APP1#">
   <daml:Ontology rdf:about="">
       <daml:imports rdf:resource=
        "http://nt-appn.comp.nus.edu.sg/fm/zdaml/OZ"/>
       <daml:imports rdf:resource=
        "http://nt-appn.comp.nus.edu.sg/fm/zdaml/CSP"/>
   </daml:Ontology>
   <rdfs:Class rdf:about="oz:Classdef">
       <rdfs:subClassOf rdf:resource="csp:Pro"/> </rdfs:Class>
   <rdfs:Class rdf:about="oz:OP">
       <rdfs:subClassOf rdf:resource="csp:Event"/> </rdfs:Class>
   <!--operation is one kind of process-->
</rdf:RDF>
```

It firstly imports the definition of CSP and Object-Z. The Object-Z class is declared as a subclass of the CSP process and the Object-Z operation (extended from Z operation schema) is declared as a subclass of the CSP event. The above two buffers example can be encoded in the Semantic Web environment as following.

```
<oz:Classdef2 rdf:ID="buffer1">
  <z:name>Buffer1</z:name>
  <oz:rename> Transfer/Leave</oz:rename>
  <oz:eqclass rdf:resource="#buffer"/> </oz:Classdef2>
<oz:Classdef2 rdf:ID="buffer2">
  <z:name>Buffer2</z:name>
  <oz:rename> Transfer/Join</oz:rename>
  <oz:eqclass rdf:resource="#buffer"/> </oz:Classdef2>
<oz:Classdef2 rdf:ID="system">
  <z:name>System</z:name>
  <oz:eqclass>
      <csp:ParallelPro>
          <csp:subprocess rdf:resource="buffer1"/>
          <csp:subprocess rdf:resource="buffer2"/>
          <csp:ParaSync>Transfer</csp:ParaSync>
      </csp:ParallelPro> </oz:eqclass> </oz:Classdef2>
```

3.2 *operation* $\Longleftrightarrow$ *process*

TCOZ approach is to identify Object-Z operations as CSP processes and all the communication must go through the explicitely declared channels. The behaviour of an active object is explicitely captured by a CSP process. To achieve this approach several new elements are introduced. They are:

Chan A channel is declared in an object's state.
Main This process defines the dynamic control behaviour of an active object.

The environment for this approach can be achieved in the following way:

```
<?xml version="1.0" encoding="UTF-8"?> <rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema#"
xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
xmlns:tcoz="http://nt-appn.comp.nus.edu.sg/fm/zdaml/TCOZ#"
xmlns:oz="http://nt-appn.comp.nus.edu.sg/fm/zdaml/OZ#"
xmlns:csp="http://nt-appn.comp.nus.edu.sg/fm/zdaml/CSP#"
xmlns="http://nt-appn.comp.nus.edu.sg/fm/zdaml/TCOZ#">
    <daml:Ontology rdf:about="">
        <daml:imports rdf:resource=
          "http://nt-appn.comp.nus.edu.sg/fm/zdaml/OZ"/>
```

```
        <daml:imports rdf:resource=
          "http://nt-appn.comp.nus.edu.sg/fm/zdaml/CSP"/>
    </daml:Ontology>
    <rdfs:Class rdf:about="oz:State">
        <rdfs:subClassOf>
            <daml:Restriction daml:minCardinality="0">
                <daml:onProperty rdf:resource="csp:chan"/>
            </daml:Restriction> </rdfs:subClassOf>
        <!-- the channel can be declared in sate schema-->
    </rdfs:Class>
    <daml:ObjectProperty rdf:ID="MAIN">
        <rdfs:range rdf:resource="csp:Process"/>
        <rdfs:domain rdf:resource="#Classdef"/>
    </daml:ObjectProperty>
    <rdfs:Class rdf:about="oz:OP">
        <rdfs:subClassOf rdf:resource="csp:Process"/>
    </rdfs:Class>
    <rdfs:Class rdf:about="csp:Process">
        <rdfs:subClassOf rdf:resource="oz:OP"/>
    </rdfs:Class>
    <!--operation is one kind of process-->
</rdf:RDF>
```

Note that the DAML allows the subclass-relation between classes to be cyclic, since a cycle of subclass relationships provides a useful way to assert equality between classes. In TCOZ, the two communicating buffer system (with timing constraints on input and output operations) can be modelled as:

TBuffer

Buffer

$left, right : \mathbf{chan}$ [input and output channels]
$t_j, t_l : \mathbb{T}$ [time durations for Join and Leave operations]

$$\textsc{Main} \mathrel{\widehat=} \mu\, Q \bullet ([i : MSG] \bullet left?i \to Join \bullet \textsc{Deadline}\ t_j\ \Box \\ \quad [size \neq 0] \bullet right!last(items) \to Leave \bullet \textsc{Deadline}\ t_l);\ Q$$

TSystem

$l : TBuffer[middle/right]$
$r : TBuffer[middle/left]$

$$\textsc{Main} \mathrel{\widehat=} l \,|[\, middle \,]|\, r$$

In the Semantic Web environment, the class *TSystem* can be encoded as follows.

```
<oz:Classdef1 rdf:ID="tsystem">
  <z:name>TSystem</z:name>
  <oz:state> <oz:State>
     <z:decl>
       <z:Decl z:name="l" z:dtype="TBuffer[middle/right]"/></z:decl>
     <z:decl>
       <z:Decl z:name="r" z:dtype="TBuffer[middle/left]"/></z:decl>
  </oz:State> </oz:state>
  <oz:MAIN>
     <csp:parallelPro>
       <csp:subprocess> <oz:Message oz:receiver="l"
         oz:method="#TBMAIN"></oz:Message> </csp:subprocess>
       <csp:subprocess> <oz:Message oz:receiver="r"
         oz:method="#TBMAIN"></oz:Message> </csp:subprocess>
       <csp:ParaSync>middle</csp:ParaSync>
     </csp:parallelPro> </oz:MAIN>
  </oz:Classdef1>
```

Clearly, unlike Smith and Derrick's approach, TCOZ is not a simple integration of Object-Z and TCSP, like CSP-OZ, TCOZ extends the two base notations with some new language constructs. Another distinct difference is that the semantic link between operation vs process in TCOZ is bi-directional ($\Longleftrightarrow$), while in Smith and Derrick's approach, the semantic link between class and process has a single direction ($\Longrightarrow$). By building the Semantic Web environments for the two approaches, one can improve the understanding of the difference. Such a Semantic Web environment is applicable for many other integrated formalisms.

4 Specifiation comprehension

One of the major contributions of the RDF model introduced by the Semantic Web community, is that it allows us to do more accurate and more meaningful searching. This strength of RDF can be applied in the specification context leading to the notion of *specification comprehension*. Useful RDF queries can be formulated for comprehending specification models particularly when models are large and complex.

There are many RDF query systems available or under development. In this paper the RDFQL [16], a RDF query language developed by Intellidimension, is used to demonstrate some queries which can be achieved in the environment.

Based on our simple *Buffer* and *TBuffer* examples, the following demonstrates various queries expressed in RDFQL.

4.1 Inter-class queries

Two typical queries can be formulated for search/understanding class relationships, such as inheritance hierarchy and composition structure.

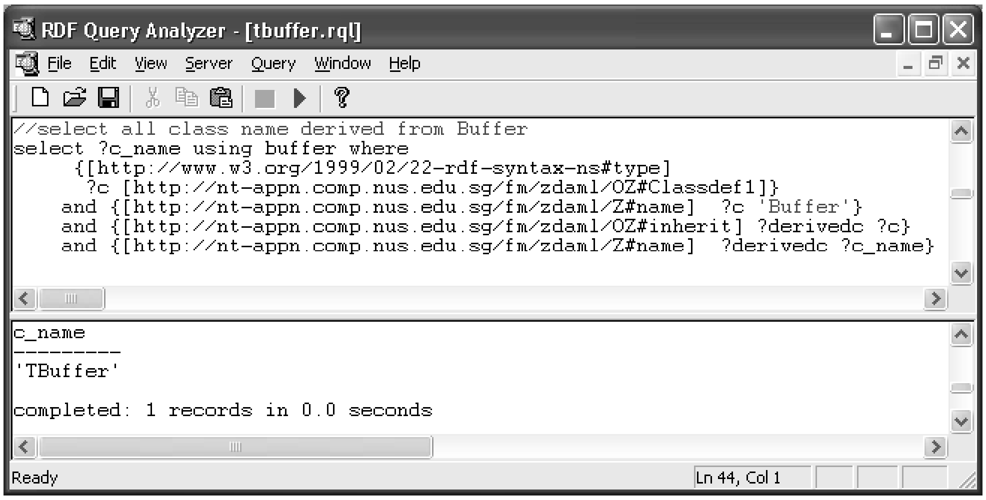

Fig. 1. Find all the sub-classes

(Inheritance) Find all the sub-classes derived from the class *Buffer* (Figure 1)

```
Query:
 select ?c_name using buffer where
     {[http://www.w3.org/1999/02/22-rdf-syntax-ns#type]
        ?c [http://nt-appn.comp.nus.edu.sg/fm/zdaml/OZ#Classdef1]}
 and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z#name]  ?c 'Buffer'}
 and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/OZ#inherit] ?derivedc ?c}
 and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z#name]  ?derivedc ?c_name}
Result: TBuffer
```

(Composition:) Find all classes containing *Buffer* instances (as attributes)

```
Query:
 select ?c_name using buffer where
     {[http://www.w3.org/1999/02/22-rdf-syntax-ns#type]
       ?c [http://nt-appn.comp.nus.edu.sg/fm/zdaml/OZ#Classdef1]}
 and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z#name]  ?c ?c_name}
 and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/OZ#state] ?c ?s}
 and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z#decl] ?s ?d}
 and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z#dtype] ?d ?dt}
 and (INSTR(?dt, 'Buffer') = 1)
Result: TSystem
```

4.2 Intro-class queries

A number of queries can be built for search/understanding class content (this is useful particularly when a class is large and has many operations).

Find all the operations which may change the attribute *items*:

```
Query:
 select ?op_name using buffer where
     {[http://www.w3.org/1999/02/22-rdf-syntax-ns#type]
         ?c [http://nt-appn.comp.nus.edu.sg/fm/zdaml/OZ#Classdef1]}
 and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z#name] ?c 'Buffer'}
 and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/OZ#op]  ?c ?op}
 and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/OZ#delObj]  ?op 'items'}
 and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z#name]  ?op ?op_name}
Result: Join, Leave
```

Find all the constant attributes in a class:

```
Query:
 select ?att using buffer where
     {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/OZ#state] ?c ?sta}
 and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z#decl] ?sta ?decl}
 and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z#name]  ?decl ?att}
 and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/OZ#delObj]  ?op ?att1}
 and (?att <> ?att1)
Result: max
```

Find all the operations which have the same interface (with common base names for output and input):

```
Query:
 select ?op_name1 ?op_name2 using buffer where
     {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/OZ#op]  ?c1 ?op1}
 and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/OZ#op]  ?c2 ?op2}
 and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z#name] ?op1 ?op_name1}
 and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z#name] ?op2 ?op_name2}
 and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z#decl]  ?op1 ?d1}
 and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z#name]   ?d1 ?n1}
 and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z#decl]  ?op2 ?d2}
 and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z#name]   ?d2 ?n2}
 and (?op1 <> ?op2) and (STRCMP(regexp(?n1,'*!'), regexp(?n2,'*?'))= 0)
Result: 'Join'    'Leave'
```

5 Related work, conclusion and further work

One of the early work by Bicarregui and Matthews [4] has proposed ideas to integrate SGML (earlier version of XML) and EXPRESS for documenting control systems design. Z notation on the web based on HTML and Java applets has been investigated by Bowen and Chippington [5] and Cinancarini, Mascolo and Vitali [9]. HTML has been successful in presenting information on the Internet, however the lack of content information has made the retrieval and exchange of resource more difficult to perform. Our previous work was to improve on those issues by taking an XML approach [27]. Recently, the Community Z Tools Initiative [20] has stated to consider to build a XML interchange format for Z according to Z standards. However, the focus of all those approaches (based on HTML and XML) was mainly for displaying and browsing Z/Object-Z specifications on the web without concern for semantic issues and integration with other formalisms. The aim of this paper is different, it focuses on building a Semantic Web (RDF/DAML) environment for supporting, extending and integrating many different formalisms. Such a *meta integrator* may bring together the strengths of various formal methods communities in a flexible and widely accessible fashion. The Semantic Web environment for formal specifications may lead to many benefits. One novel application which has been demonstrated in this paper is the notion of specification comprehension based RDF query techniques. The review process of a large specification can be facilitated by various RDF queries.

Using the terminology from Jackson and Wing [17], this paper has demonstrated the potential for constructing a lightweight supporting environment and tools for all formal specification languages and their various (existing or even possible future) integrations. Recent efforts and success in formal methods have been focused on building 'heavy' tools for formal specifications, such as proof tools (e.g. Mural[3] for VDM, EVE[23] for Z etc) and model checkers (e.g. FDR [22] for CSP). Although those tools are essential and important for applications of formal methods, in order to achieve wider acceptance, the development of light weight tools such as the Semantic Web environment for formal specifications is also important. Interfacing such a web environment with proof tools and model checkers would be an interesting future work. Another interesting different research direction will be to investigate how formal specification techniques can facilitate web-based ontology design such that formal methods not only can benefit from web technologies but also can contribute to the web applications.

Acknowledgements

We would like to thank Hugh Anderson, DSTA staffs and anonymous referees for many helpful comments. This work is supported by the Academic Research grant *Integrated Formal Methods* from National University of Singapore and Defence Innovative Research grant *Formal Design Methods and DAML* from Defence Science & Technology Agency (DSTA) Singapore.

References

1. K. Araki, A. Galloway, and K. Taguchi, editors. *IFM'99: Integrated Formal Methods, York, UK.* Springer-Verlag, June 1999.
2. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. Scientific American, May 2001.
3. J.C. Bicarregui, J.S. FitzGerald, P.A. Lindsay, R. Moore, and B. Ritchie. *Proof in VDM: A practioners Guide.* Springer Verlag, 1994.
4. J.C. Bicarregui and B. M. Matthews. Integrating EXPRESS and SGML for Document Modelling in Control Systems Design. In *EUG'95, 5th Annual EXPRESS User Group International Conference*, 1995.
5. J. P. Bowen and D. Chippington. Z on the Web using Java. In Bowen et al. [6], pages 66–80.
6. J. P. Bowen, A. Fett, and M. G. Hinchey, editors. *ZUM'98: The Z Formal Specification Notation, 11th International Conference of Z Users, Berlin, Germany, 24–26 September 1998*, volume 1493 of *Lect. Notes in Comput. Sci.* Springer-Verlag, 1998.
7. D. Brickley and R.V. Guha (editors). Resource description framework (rdf) schema specification 1.0. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/, March, 2000.
8. M. Butler. csp2B: A Practical Approach To Combining CSP and B. In J. Wing, J. Woodcock, and J. Davies, editors, *FM'99: World Congress on Formal Methods*, Lect. Notes in Comput. Sci., Toulouse, France, September 1999. Springer-Verlag.
9. P. Ciancarini, C. Mascolo, and F. Vitali. Visualizing Z notation in HTML documents. In Bowen et al. [6], pages 81–95.
10. R. Duke and G. Rose. *Formal Object Oriented Specification Using Object-Z.* Cornerstones of Computing. Macmillan, March 2000.
11. C. Fischer and H. Wehrheim. Model-Checking CSP-OZ Specifications with FDR. In Araki et al. [1].
12. A. J. Galloway and W. J. Stoddart. An operational semantics for ZCCS. In Hinchey and Liu [14], pages 272–282.
13. W. Grieskamp, T. Santen, and B. Stoddart, editors. *IFM'00: Integrated Formal Methods,*, Lect. Notes in Comput. Sci. Springer-Verlag, October 2000.
14. M. Hinchey and S. Liu, editors. *the IEEE International Conference on Formal Engineering Methods (ICFEM'97)*, Hiroshima, Japan, November 1997. IEEE Computer Society Press.
15. C.A.R. Hoare. *Communicating Sequential Processes.* International Series in Computer Science. Prentice-Hall, 1985.
16. Intellidimension Inc. Rdfql reference manual. http://www.intellidimension.com/RDFGateway/Docs/rdfqlmanual.asp, 2001.
17. D. Jackson and J. Wing. Lightweight formal methods. *IEEE Computer*, April 1996.
18. O. Lassila and R. R. Swick (editors). Resource description framework (rdf) model and syntax specification. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/, Feb, 1999.
19. B. Mahony and J. S. Dong. Timed Communicating Object Z. *IEEE Transactions on Software Engineering*, 26(2):150–177, February 2000.
20. Andrew P. Martin. Community z tools initiative. http://web.comlab.ox.ac.uk/oucl/work/andrew.martin/CZT/, 2001.
21. R. Paige. Formal method integration via heterogeneous notations. PhD Dissertation, University of Toronto, 1997.

22. A.W. Roscoe. *The Theory and Practice of Concurrency.* Prentice-Hall, 1997.
23. M. Saaltink. Z and EVES. In *Proceedings of Sixth Annual Z-User Meeting*, University of York, Dec 1991.
24. S. Schneider, J. Davies, D. M. Jackson, G. M. Reed, J. N. Reed, and A. W. Roscoe. Timed CSP: Theory and practice. In J. W. de Bakker, C. Huizing, W. P. de Roever, and G. Rozenberg, editors, *Real-Time: Theory in Practice*, volume 600 of *Lect. Notes in Comput. Sci.*, pages 640–675. Springer-Verlag, 1992.
25. G. Smith. *The Object-Z Specification Language.* Advances in Formal Methods. Kluwer Academic Publishers, 2000.
26. G. Smith and J. Derrick. Specification, refinement and verification of concurrent systems - an integration of Object-Z and CSP. *Formal Methods in System Design*, 18:249–284, 2001.
27. J. Sun, J. S. Dong, J. Liu, and H. Wang. Object-Z Web Environment and Projections to UML. In *WWW-10: 10th International World Wide Web Conference*, pages 725–734. ACM Press, May 2001.
28. K. Taguchi and K. Araki. The State-Based CCS Semantics for Concurrent Z Specification. In Hinchey and Liu [14], pages 283–292.
29. H. Treharne and S. Schneider. Using a Process Algebra to control B OPERATIONS. In Araki et al. [1].
30. F. van Harmelen, P. F. Patel-Schneider, and I. Horrocks (editors). Reference description of the daml+oil ontology markup language. Contributors: T. Berners-Lee, D. Brickley, D. Connolly, M. Dean, S. Decker, P. Hayes, J. Heflin, J. Hendler, O. Lassila, D. McGuinness, L. A. Stein, ..., March, 2001.
31. World Wide Web Consortium (W3C). Extensible markup language (xml). http://www.w3.org/XML.
32. J. Woodcock and A. Cavalcanti. The steam boiler in a unified theory of Z and CSP. In *The 8th Asia-Pacific Software Engineering Conference (APSEC'01)*, pages 291–298. IEEE Press, 2001.
33. J. Woodcock and J. Davies. *Using Z: Specification, Refinement, and Proof.* Prentice-Hall International, 1996.
34. P. Zave and M. Jackson. Where do operations come from?: A multiparadigm specification technique. *IEEE Transactions on Software Engineering*, 22(7):508–528, July 1996.