CSP SSDL Protocol Framework

Savas Parastatidis[1], Jim Webber[2]
Savas@Parastatidis.name, Jim@Webber.name

Abstract

The Communicating Sequential Processes (CSP) SSDL Protocol Framework defines a collection of XML Infoset element information items for defining a multi-message exchange using sequential process semantics.

SSDL documents

An Introduction to the SOAP Service Description Language [1]

SOAP Service Description Language (SSDL) v1.3 [2]

MEP SSDL Protocol Framework v1.3 [3]

CSP SSDL Protocol Framework v1.3 [4] (this document)

Rules SSDL Protocol Framework v1.3 [5]

Sequencing Constraints SSDL Protocol Framework v1.3 [6]

Status of this document

Version: 1.3

Date: April 2005

http://ssdl.org

Disclaimer

[1] School of Computing Science, University of Newcastle, Newcastle upon Tyne, NE1 7RU, UK
[2] ThoughtWorks Australia Pty. Ltd.

Table of Contents

1. Introduction

The Communicating Sequential Processes (CSP) SSDL Protocol Framework defines a collection of XML Infoset element information items for defining a multi-message exchange using CSP [7] semantics.

1.1. SSDL

The SOAP Service Description Language (SSDL) is a SOAP-centric contract description language for Web Services. It is meant as a means for exploring ideas in the areas of contract and protocol description and Web Services implementations using message-oriented programming abstractions.

1.1.1. Motivation

SOAP is the standard message transfer protocol for Web Services. However, the default description language for Web Services (WSDL) does not explicitly target SOAP but, instead, provides a generic framework for the description of network-exposed software artefacts. The work on SSDL aims to investigate the advantages/disadvantages of Web Services description when SOAP is assumed from the outset compared to the transfer-independent approach of WSDL. The use of formal models for describing message-based interactions is also a goal for SSDL. Finally, this work aims to demonstrate the benefits of focusing on message-orientation when architecting, designing, and building Web Services rather than on the interface and remote procedure call abstractions.

1.1.2. The Language

The SOAP Service Description Language provides the base framework for a range of protocol description frameworks which at one end of the spectrum can be a simpler, SOAP-focussed, direct replacement for WSDL MEPs while at the other end of the spectrum can enable formal validation and reasoning about the protocols that a Web Service supports.

The frameworks are componentised in a similar way to the WS-Policy suite of specifications, with a base SSDL framework providing the fundamental protocol building blocks for describing messages while other specifications utilise those building blocks to describe the way in which the messages participate in the protocols that a Web Service supports. This is illustrated in Figure 1.

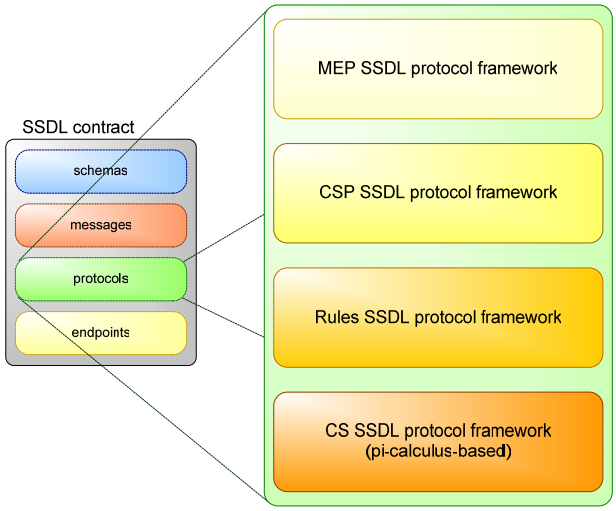

Figure 1 The SSDL Suite of Specifications

Four protocol description frameworks are provided with the base SSDL specification but it is expected that others will be implemented to meet different needs. The protocol description frameworks provided with the initial release of SSDL 1.0 are:

- MEP - The Message Exchange Patterns (MEP) Framework defines a collection of XML Infoset element information items that represent commonly used simple exchange patterns. The current set of message exchange patterns supported by the MEP framework is a superset of that found in the latest draft of the WSDL specification.
- CSP – The Communicating Sequential Processes (CSP) Framework defines a collection of XML Infoset element information items for defining a multi-message exchange using sequential process semantics, based on basic CSP semantics. Work is currently underway to make use of the full strength of the CSP in describing contracts.
- Rules - The Rules-based SSDL Protocol Framework defines a collection of XML Infoset element information items that can be used to describe a multi-message exchange protocol using conditions. The protocols captured using the Rules-based SSDL protocol framework can be validated for correctness, liveness, and other properties.
- SC - The Sequencing Constraints (SC) Protocol Framework defines a collection of XML Infoset element information items that can be used to describe a multi-party, multi-message exchange protocol using notations based on the pi-calculus. Protocols in the framework are specified using a sequential technique, specifying the legal set of actions at each stage of the protocol. The framework is intended to provide a simple way of specifying protocols but also have a formal basis to allow properties of the protocols to be determined.

1.1.3. Key Features

- SSDL assumes SOAP as the means of transferring messages between Web Services over arbitrary transport (and transfer) protocols. It has been designed to work harmoniously with all aspects of the underlying SOAP processing model. As a result, there is no need to define bindings for all possible transport protocols;
- SSDL assumes WS-Addressing as the standard means for embedding addressing information within SOAP envelopes and for binding those addresses onto underlying transport protocols;
- SSDL focuses on messages and protocols. As a result, there is no need for articles like ‘interface’, ‘inheritance’, and ‘operation’;
- XML Infoset is assumed as the underlying SSDL component model. There is no need (nor desire) to create a new component model simply for contract description;
- Modularisation of contracts is handled using XInclude. A shortcut mechanism is provided which is defined in terms of XInclude elements to simplify componentisation as far as is possible;
- SSDL promotes protocol framework extensibility. It allows different protocol description models to be plugged into the base SSDL framework which helps promote protocol-based integration and exposure of the messaging behaviour of a Web Service. Tools such as model checkers can verify the correctness of protocols defined in an SSDL contract, or automate the reasoning about the compatibility of Web Services. Hosting environments can even use the SSDL contract to validate the message exchanges between Web Services.

1.2. Goals

- Capture a multi-message protocol exchange using CSP semantics.

1.3. Example

```
<?xml version="1.0" encoding="utf-8" ?>
<ssdl:contract targetNamespace="http://example.org/service/contract"
               xmlns:ssdl="urn:ssdl:v1">
```

```
  <ssdl:schemas>
    <!-- schemas -->
  </ssdl:schemas>

  <ssdl:messages targetNamespace="http://example.org/service/messages"
                 xmlns:tns="http://exaxmple.org/service/schema.xsd">

    <ssdl:message name="Msg1">
      <ssdl:body ref="tns:Msg1Description"/>
    </ssdl:message>

    <ssdl:message name="Msg2">
      <ssdl:body ref="tns:Msg2Description"/>
    </ssdl:message>

    <ssdl:message name="Msg3">
      <ssdl:header ref="Header1Description" mustUnderstand="true" />
      <ssdl:body ref="tns:Msg3Description"/>
    </ssdl:message>

    <ssdl:fault name="Fault1">
      <ssdl:code value="Sender"/>
    </ssdl:fault>

  </ssdl:messages>

  <ssdls>
    <ssdl targetNamespace="http://example.org/service/protocol"
                  xmlns:msgs="http://example.org/service/messages"
                  xmlns:sp="urn:ssdl:csp:v1">
      <csp:process>
        <csp:sequence>
          <ssdl:msgref ref="msgs:Msg1" direction="in"/>
          <csp:d-choice>
            <csp:sequence>
              <ssdl:msgref ref="msgs:Msg2" direction="out" />
              <ssdl:msgref ref="msgs:Msg3" direction="in" />
            </csp:sequence>
            <ssdl:msgref ref="msgs:Fault1" direction="out" />
          </csp:d-choice>
        </csp:sequence>
    </ssdl>

   <ssdl:endpoints>
     <ssdl:endpoint xmlns:wsa="http://www.w3.org/2004/12/addressing">
       <wsa:Address>http://example.org/service</wsa:Address>
     </ssdl:endpoint>
   </ssdl:endpoints>
</ssdl:contract>
```

Listing 1: A contract with a protocol defined using the CSP Protocol Framework

In Listing 1, a service contract is defined for a Web Service which supports a protocol-behaviour described using the CSP SSDL protocol framework. The example indicates that when a `msgs:Msg1` message is received, the reply will be either a `msgs:Msg2` or a `msgs:Fault1` message. If a `msgs:Msg2` message is sent, then it is expected that a `msgs:Msg3` message is received.

1.4. Notational Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [8].

This specification uses properties from the XML Information Set [9]. Such properties are denoted by square brackets and in bold, e.g. **[namespace name]**.

This specification uses namespace prefixes throughout; they are listed in Table 1-1. Note that the choice of any namespace prefix is arbitrary and not semantically significant (see [9]).

We use the pseudo-schema notation used in WSDL 2.0 Core [10] as a convenient description of the structure of a component.

1.5. Namespaces

These namespaces and their prefixes are used throughout this document.

Prefix	Namespace	Notes
ssdl	urn:ssdl:v1	
xs	http://www.w3.org/2001/XMLSchema	
csp	urn:ssdl:csp:v1	Where elements are not qualified with a namespace prefix, urn:ssdl:csp:v1 is assumed

2. General Description

The CSP SSDL protocol framework enables Web Service contract authors to define multi-message interactions as a sequential process. A sequential process is effectively a state machine which can be represented as a graph. For example, the protocol description of Listing 1 can be represented as the following graph:

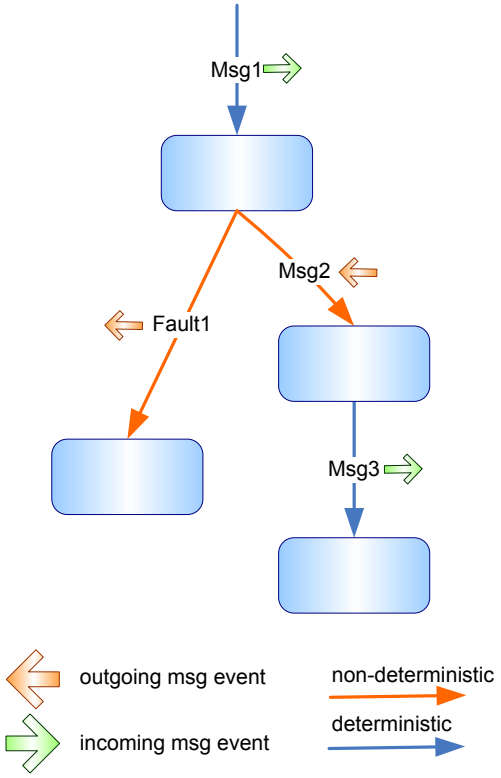

Figure 2: A graph representing the protocol described in Listing 1 (the attached arrows show the direction of the message while the arcs show the relative order of the messages)

More complex graphs which may even be cyclic can also be captured:

```
<csp:process>
  <csp:non-d-choice>
    <msgref ref="msgs:StreamEndMsg"
            direction="out" />
    <csp:d-choice>
      <csp:sub-process-ref ref="prtcl:subprocess" />
      <msgref ref="msgs:StreamMsg" direction="out" />
    </csp:d-choice>
  </csp:non-d-choice>
</csp:process>

<csp:sub-process name="subprocess">
  <csp:sequence>
    <msgref ref="msgs:StreamEndRequestMsg"
            direction="in" />
    <csp:non-d-choice>
      <msgref ref="msgs:StreamEndMsg"
              direction="out" />
      <msgref ref="msgs:NoStreamFaultMsg"
              direction="out" />
    </csp:non-d-choice>
  </csp:sequence>
</csp:sub-process>
```

Listing 2: An example of a protocol

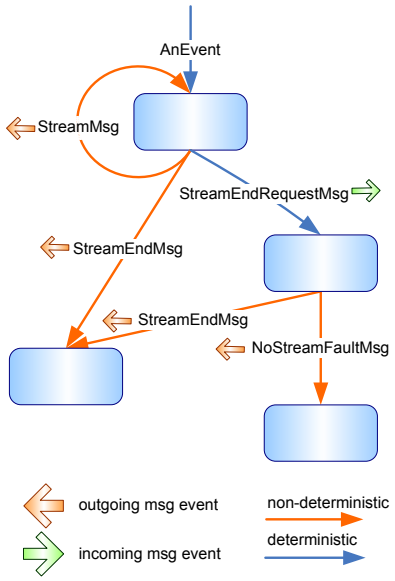

Figure 3: A state machine representing the protocol of Listing 2

Note that in Figure 3 each node represents the current state of the protocol execution while the arcs indicate the events. The example also shows how a protocol can be modularised through sub-process descriptions.

The CSP SSDL protocol framework is silent on the means of the contextualisation of the multi-message interaction. Specifications that support message contextualisation like WS-Context [11], WS-Security [12], WS-Addressing [13], etc. could be used to distinguish between different executions of a protocol. It is up to individual Web Service implementations to define the semantics of the contextualisation by using an existing context protocol or by examining information held in messages. It may also be possible that a Web Service does not support multiple executions of the same protocol.

3. Protocol Framework Structure

This section describes the CSP SSDL protocol framework structure using the XML Information Set [9] model.

```
<ssdl>
  <process>
  <sub-process> *
</ssdl>
```

The CSP SSDL protocol framework defines the following element information items which can be used within the `ssdl` element information item of an SSDL contract:

- A REQUIRED `process` element information item
- Zero or more `sub-process` element information items

3.1. process

```
<ssdl>
  <process>
    [ <d-choice /> |
      <non-d-choice /> |
      <sequence /> |
      <sub-process-ref /> ] +
  </process>
</ssdl>
```

The `process` element information item defines the entry point of the protocol execution.

A `process` element information item has the following properties:

- A **[local name]** of "process"
- A **[namespace name]** of "urn:ssdl:csp:v1"
- A REQUIRED element information item in the **[children]** property, which can be one of the following:
    - A `d-choice` element information item
    - A `non-d-choice` element information item
    - A `sequence` element information item
    - A `sub-process-ref` element information item

3.2. sub-process

```
<ssdl>
  <sub-process name="xs:string">
    [ <d-choice /> |
      <non-d-choice /> |
      <sequence /> |
      <sub-process-ref /> ] +
  </sub-process>
</ssdl>
```

The `sub-process` element information item is used to create sub-processes that can be referred through the `sub-process-ref` element information item. It has the following properties:

- A **[local name]** of "sub-process"
- A **[namespace name]** of "urn:ssdl:csp:v1"
- A REQUIRED `name` attribute information item
- A REQUIRED element information item in the **[children]** property, which can be one of the following:
    - A `d-choice` element information item
    - A `non-d-choice` element information item
    - A `sequence` element information item
    - A `sub-process-ref` element information item

3.2.1. name

The **[normalised value]** of the `name` attribute information item is the name of the defined sub-process. The sub-process is considered to be defined in the namespace indicated by the **[normalised value]** of the `targetNamespace` attribute information item of the ancestor `protocol` element information item. The name attribute information item has the following properties:

- A **[local name]** of "name"
- A **[namespace name]** of "urn:ssdl:csp:v1"

3.3. d-choice

```
<d-choice>
 [ <ssdl:msgref /> |
   <d-choice /> |
   <non-d-choice /> |
   <sequence /> |
   <sub-process-ref /> ] +
</d-choice>
```

A `d-choice` element information item represents the CSP *deterministic choice* operator and it means that only one of its **[children]** element information items representing events MUST be deterministically observed. The `d-choice` element information item has the following properties:

- A **[local name]** of "d-choice"
- A **[namespace name]** of "urn:ssdl:csp:v1"
- A REQUIRED element information item in the **[children]** property, which can be one of the following:
    - A `ssdl:msgref` element information item [14]
    - A `d-choice` element information item
    - A `non-d-choice` element information item
    - A `sequence` element information item
    - A `sub-process-ref` element information item

3.4. non-d-choice

```
<non-d-choice>
 [ <ssdl:msgref /> |
   <d-choice /> |
   <non-d-choice /> |
   <sequence /> |
   <sub-process-ref /> ] +
</non-d-choice>
```

A `non-d-choice` element information item represents the CSP *non-deterministic choice* operator and it means that only one of its **[children]** element information items representing events MUST be non-deterministically observed. The `non-d-choice` element information item has the following properties:

- A **[local name]** of "non-d-choice"
- A **[namespace name]** of "urn:ssdl:csp:v1"
- A REQUIRED element information item in the **[children]** property, which can be one of the following:
    - A `ssdl:msgref` element information item [14]
    - A `d-choice` element information item
    - A `non-d-choice` element information item
    - A `sequence` element information item
    - A `sub-process-ref` element information item

3.5. sequence

```
<sequence>
 [ <ssdl:msgref /> |
   <d-choice /> |
   <non-d-choice /> |
   <sequence /> |
   <sub-process-ref /> ] +
</sequence>
```

A `sequence` element information item represents the CSP *sequence* operator and mandates that all of its **[children]** element information items representing events MUST be observed in the order they appear. The `sequence` element information item has the following properties:

- A **[local name]** of "sequence"

- A **[namespace name]** of “urn:ssdl:csp:v1”
- A REQUIRED element information item in the **[children]** property, which can be one of the following:
    - A `ssdl:msgref` element information item [14]
    - A `d-choice` element information item
    - A `non-d-choice` element information item
    - A `sequence` element information item
    - A `sub-process-ref` element information item

3.6. sub-process-ref

```
<sub-process-ref ref="xs:QName" />
```

The `sub-process-ref` element information item is used to refer to a defined process. It has the following properties:

- A **[local name]** of “sub-process-ref”
- A **[namespace name]** of “urn:ssdl:csp:v1”
- A REQUIRED `ref` attribute information item in its **[children]** property

3.6.1. ref

The **[normalised value]** of the `ref` attribute information item is the qualified name (*xs:QName*) of a sub-process that is defined through a `sub-process` element information item. The `ref` attribute information item has the following properties:

- A **[local name]** of “ref”
- A **[namespace name]** of “urn:ssdl:csp:v1”

References

[1] S. Parastatidis, J. Webber, S. Woodman, D. Kuo, and P. Greenfield, "An Introduction to the SOAP Service Description Language," School of Computing Science, University of Newcastle, Newcastle upon Tyne CS-TR-898, 2005.

[2] S. Parastatidis, J. Webber, S. Woodman, D. Kuo, and P. Greenfield, "SOAP Service Description Language (SSDL)," School of Computing Science, University of Newcastle, Newcastle upon Tyne CS-TR-899, 2005.

[3] S. Parastatidis and J. Webber, "MEP SSDL Protocol Framework," School of Computing Science, University of Newcastle, Newcastle upon Tyne CS-TR-900, 2005.

[4] S. Parastatidis and J. Webber, "CSP SSDL Protocol Framework," School of Computing Science, University of Newcastle, Newcastle upon Tyne CS-TR-901, 2005.

[5] D. Kuo, S. Parastatidis, and J. Webber, "Rules SSDL Protocol Framework," School of Computing Science, University of Newcastle, Newcastle upon Tyne CS-TR-902, 2005.

[6] S. Woodman, S. Parastatidis, and J. Webber, "Sequencing Constraints SSDL Protocol Framework," School of Computing Science, University of Newcastle, Newcastle upon Tyne CS-TR-903, 2005.

[7] C. A. R. Hoare, *Communicating Sequential Processes*: Prentice Hall International, 1985.

[8] S. Bradner, "IETF RFC 2119: Key words for use in RFCs to Indicate Requirement Levels." http://www.ietf.org/rfc/rfc2119.txt: Internet Engineering Task Force, 1999.

[9] W3C, "XML Information Set." http://www.w3.org/TR/xml-infoset/, 2004.

[10] W3C, "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language," R. Chinnici, M. Gudgin, J.-J. Moreau, J. Schlimmer, and S. Weerawarana, Eds. http://www.w3.org/TR/2004/WD-wsdl20-20040803/, 2004.

[11] OASIS, "Web Services Composite Application Framework (WS-CAF)." http://www.oasis-open.org/committees/ws-caf.

[12] OASIS, "Web Services Security (WS-Security)." http://www.oasis-open.org/committees/wss.

[13] W3C, "Web Services Addressing (WS-Addressing)." http://www.w3.org/2002/ws/addr/.

[14] S. Parastatidis and J. Webber, "SOAP Service Description Language," 2005.

Acknowledgments

The authors would like to thank Jon Burton (J.I.Burton@newcastle.ac.uk, School of Computing Science, University of Newcastle upon Tyne, UK) for his significant contributions to the preparation of this specification.

Appendix A – XML Schema

```
<?xml version="1.0" encoding="utf-8" ?>
<!--
    $Modtime: 12/02/05 15:52 $
    $Revision: 12 $
-->
<xs:schema
      elementFormDefault="qualified"
      targetNamespace="urn:ssdl:csp:v1"
      xmlns:ssdl="urn:ssdl:v1"
      xmlns:csp="urn:ssdl:csp:v1"
      xmlns:xs="http://www.w3.org/2001/XMLSchema">

   <xs:import namespace="urn:ssdl:v1" />

   <xs:element name="process" type="csp:process-type" />

   <xs:element name="sub-process" type="csp:sub-process-type" />

   <xs:complexType name="sub-process-type">
      <xs:complexContent>
         <xs:extension base="csp:process-type">
            <xs:attribute name="name" type="xs:string" />
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>

   <xs:complexType name="process-type">
      <xs:choice minOccurs="1" maxOccurs="unbounded">
         <xs:element name="d-choice" type="csp:body-type" />
         <xs:element name="non-d-choice" type="csp:body-type" />
         <xs:element name="sequence" type="csp:body-type" />
         <xs:element name="all" type="csp:body-type" />
         <xs:element name="sub-process-ref" type="csp:sub-process-ref-type" />
      </xs:choice>
   </xs:complexType>

   <xs:complexType name="body-type">
      <xs:choice minOccurs="1" maxOccurs="unbounded">
         <xs:element ref="ssdl:msgref" />
         <xs:element name="d-choice" type="csp:body-type" />
         <xs:element name="non-d-choice" type="csp:body-type" />
         <xs:element name="sequence" type="csp:body-type" />
         <xs:element name="all" type="csp:body-type" />
         <xs:element name="sub-process-ref" type="csp:sub-process-ref-type" />
      </xs:choice>
   </xs:complexType>

   <xs:complexType name="sub-process-ref-type">
      <xs:attribute name="ref" type="xs:QName" use="required" />
   </xs:complexType>
</xs:schema>
```