ws-prov-ds

Authors:
Victor Tan, U. Southampton
Paul Groth, U. Southampton,
Sheng Jiang, U. Southampton
Simon Miles, U. Southampton
Steve Munroe, U. Southampton
Sofia Tsasakou, U. Southampton
Luc Moreau, U. Southampton

November 23, 2006

Basic Transformation Profile for Documentation Style

Status of this Memo

This document provides information to the community regarding the specification of a data model for documentation style. It is intended to supplement the data model for process documentation [MGJ+06], and to be read in conjunction with that document. This document does not define any standards or technical recommendations. Distribution is unlimited.

Copyright Notice

Abstract

The data model for process documentation [MGJ+06] provides descriptions of different ways in which processes may be documented. The primary method involves recording messages exchanged between interacting services, as well as the state of those services at the time of message exchange. These messages and states may have their contents transformed when recorded due to application dependent security and scalability requirements. The documentation style describes the types of transformations that can be performed. An actor that processes the recorded documentation must understand the transformations performed on it in order to interpret or utilise it appropriately. This document is a profile of several basic documentation style transformations that are likely to be useful in application domains that use process documentation. It is not intended to be exhaustive; other profiles may be provided of alternative documentation style transformations which may be generic or more specific in nature.

Contents

1 Introduction

The documentation style data model is presented here in the context of the process documentation model [MGJ+06]. In the process documentation model, p-assertions are atomic units of process documentation of which there exists three forms: relationship p-assertions, actor state p-assertions and interaction p-assertions. When an actor in a provenance-aware system documents an interaction, it constructs an interaction p-assertion, which states the content of a message received or sent by that actor. An actor can also make assertions about its internal state in the context of a specific interaction through an actor state p-assertion. This can be the state of the actor prior to or after a message exchange. Relationship p-assertions allow uni-directional relationships between both messages and data to be expressed.

The activity of constructing an interaction p-assertion from a message can be considered as a single atomic transformation. This transformation needs to be adequately defined by the actor creating that p-assertion in order that actors who retrieve that p-assertion from the provenance store are able to understand the nature of the transformation applied. This is equally true for an actor state p-assertion. Documentation styles are essentially descriptions of the types of transformations that can be applied to a message or to the internal state of an actor. Relationship p-assertions do not utilize documentation styles; this is further clarified in Section 3.

1.1 Goals and Requirements

The goal of this document is to develop an open, interoperable approach to process documentation.

1.1.1 Requirements

This specification intends to meet the following requirements:

- Provide a profile of a data model that describes some basic documentation style transformations.
- Provide extensibility in the data model for describing new types of documentation style transformations.

1.1.2 Non-Requirements

This document does not intend to meet the following requirements:

- Provide an implementation specific approach to realizing these documentation style transformations.

- Provide an exhaustive list of documentation style transformations relevant to process documentation.

2 Terminology and Notation

All definitions for the concepts and structures found within this document can be found in [TGJ+06].

2.1 XML Namespaces

The XML Namespace URI that MUST be used by implementations of this specification is: `http://www.pasoa.org/schemas/version023s1/docstyle`

Table 1 lists XML namespaces that are used in this specification. The choice of any namespace prefix is arbitrary and not semantically significant.

Prefix	XML Namespace	Specification(s)
pds	`http://www.pasoa.org/schemas/version023s1/docstyle`	[DocumentationStyle]
xenc	`http://www.w3.org/2001/04/xmlenc#`	[XML Encryption]
ds	`http://www.w3.org/2000/09/xmldsig#`	[XML Signature]
xs	`http://www.w3.org/2001/XMLSchema`	[XML Schema]

Table 1: Prefixes and XML Namespaces used in this specification

2.2 Notational Conventions

The keywords "MUST ", "MUSTNOT ", "REQUIRED ", "SHALL ", "SHALLNOT ", "SHOULD ", "SHOULDNOT ", "RECOMMENDED ", "MAY ", and "OPTIONAL " in this document are to be interpreted as described in [Bra97].

2.3 XML Schema Diagrams

This document adopts a graphical notation to describe XML Schema. Figure 1 gives an example of a small XML Schema displayed as a diagram, which is now explained with reference to the figure.

Figure 1 defines the structure of type ts:Test. The type Test contains a sequence of elements, which we now detail. One element in the sequence is ts:testName, which can be any type and must occur once and only once in an instance of ts:Test. ts:Name is followed by element ts:testNumber, which must contain a string. The ts:testNumber element must occur at least once and can occur as many times as needed. This is denoted by the "1..unbounded" under the element. Finally, the sequence contains a choice between two elements, ts:startTest and ts:stopTest, either of which must contain a date.

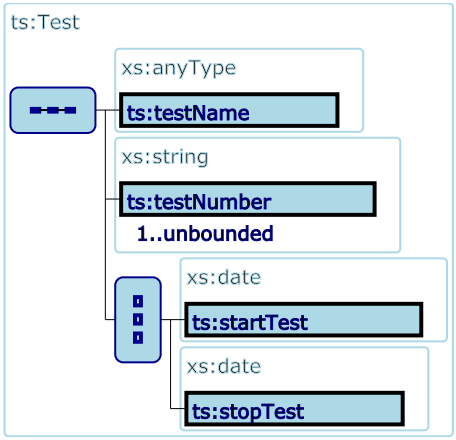

Figure 1: An example XML Schema diagram

Below is a simple of description of each of the parts of the XML Schema diagram format.

An element (instance) is represented by the qualified name of the element in the box. By default an element must occur once and only once. Where this restriction does not hold, the text "1..unbounded", "0..unbounded", "0..N", "1..N" (where N is an integer) appears under the element box. The left hand number is the minimum occurrences of the element at the position in the XML document, the right hand number is the maximum (with "unbounded" for no maximum).

A complex type is denoted by a lightly marked box with the qualified name of the type at the top left. The structure of the type is given by the elements, types and control structures within the box.

A horizontal sequence of dots represents a sequence of elements or control structures, that must appear in an element conforming to the type in the surrounding type box.

A vertical sequence of dots represents a choice between elements or control structures, that must appear in an element conforming to the type in the surrounding type box.

2.4 XPath notation

In addition to the XML Schema diagrams, an XPath notation [W3C99] is used to identify each individual element in the specification along with its context, in order to describe precisely its meaning and use.

3 Data Model for Transformation Description Document

In the process documentation model [MGJ+06], the content of an interaction p-assertion is intended to provide information about a single message exchanged between two actors. In some cases, the content may simply be the actual message itself verbatim. However, there may often be an application-specific need to transform or modify the actual message in some way before storing it as the content of an interaction p-assertion. For example, parts of the message may contain highly sensitive information and there may be a corresponding security requirement within the application environment to obscure these parts in some manner or remove them prior to placing the message itself into a p-assertion. A message may contain a large amount of raw data that is irrelevant to process documentation; the raw data could effectively be replaced with an external reference to minimize the size of the interaction p-assertion created.

Actor state p-assertions contain state information about an actor which may be structured in an arbitrary fashion by the actor concerned. This structured information can also be transformed in the same ways as interaction p-assertions, based on similar motivations as well.

Documentation style transformations then refer in general to all the possible types of transformations that can be applied to a message exchanged between actors or state information provided by an actor in order to obtain a transformed output that will become the content of either an interaction p-assertion or actor state p-assertion. It is possible, but not mandatory, that documentation style transformations are reversible i.e. the content of a p-assertion can be reverse-transformed to produce the input that it was originally derived from.

The information required to perform a specific transformation must be provided as a document (the *transformation definition document*). The transformation is performed using this document to produce an output that should be properly typed. For the case when the output of a transformation is an XML document, this requires that it should be well formed and be capable of being validated against a schema. The transformed output is then stored within a p-assertion while the transformation definition document is then made available at a repository. This should be accessible to all actors that process p-assertions in order to allow them to understand how the contents of a specific p-assertion was created.

Documentation style transformations shall not be applicable to relationship p-assertions. The information in the fields of a relationship p-assertion have an explicit meaning within the context of the process documentation data model, and must not be altered in any way in order to ensure that provenance queries that use them produce correct results.

3.1 Transform Definition and Transform Operation

The structure of a transformation definition document is shown in Figure 2. It describes a documentation style transformation and the nature of the input and output (an XML document, Java object, CORBA object) that it functions on. Both the input and output are application dependent and may not be based on the same technology. For example, a transformation could operate on a Java object as an input and produce an XML document as output. The location of the transformation definition document must be specified as a URI in the `/ps:interactionPAssertion /ps:documentationStyle` component of an interaction p-assertion (Figure 4 [MGJ+06]). The constituent components of this structure as described below must be provided, unless otherwise stated.

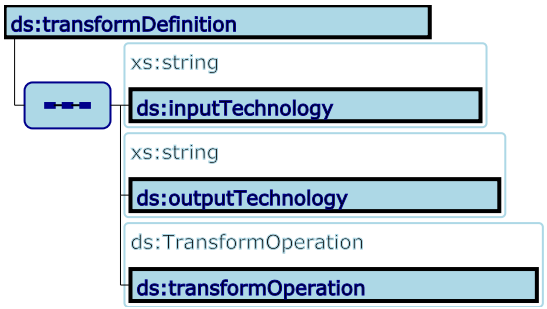

Figure 2: Model for a transform definition

`/ds:transformDefinition`

This is the root element and provides a logical structure for the document that describes the documentation style transform operation

`/ds:transformDefinition/ds:inputTechnology`

This element specifies the underlying technology for the object to be transformed, whether it is an XML document, CORBA object, etc.

`/ds:transformDefinition/ds:outputTechnology`

This element specifies the technology for the transformed object, whether it is an XML document, CORBA object, etc.

`/ds:transformDefinition/ds:transformOperation`

This component provides the relevant information required in the execution of the specific transform operation.

All the documentation style transformations that are described in the remaining sections in this document shall be extensions of the `/ds:transformDefinition /ds:transformOperation` element, each containing information specific to themselves. Any new user-defined documentation style should be defined as an instance in a similar manner as well.

3.2 Verbatim Documentation Style

The verbatim documentation style (Figure 3) denotes a null transformation applied to a given input; the output of the transformation is identical to its input. The main intention of this style is to indicate the creation of an interaction p-assertion where the contents of the p-assertion is the exchanged message as it is.

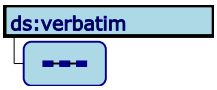

Figure 3: Model for a verbatim documentation style

`/ds:verbatim`

This is the root element for the description of the verbatim documentation style.

3.3 Reference Documentation Style

The reference documentation style (Figure 4) denotes a transformation of an input by which a part of (or the whole of) its contents has been replaced by a reference to the location where the actual contents can be found. This transformation operation may include the computation of a digest on the contents to be replaced. The digest, if included, shall be appended to the transformed message in a manner that associates it with the reference URI. The primary functionality of the digest is to verify that in a reverse transformation, data retrieved from a reference URI is identical to the data on which the digest was originally computed.

For the case of an XML document, the transformed result shall have a new namespace corresponding to the additional elements appended into it. In addition, namespaces of existing untransformed elements in the original document

may be changed to new namespaces in the transformed document in order to reflect that a transformation has occurred. If such namespace change is desired, a list mapping namespaces in the original document to new namespaces must be provided to allow the transformation functionality to make these namespace substitutions when constructing the transformed document. Subsequent validation of the output document against a specified schema, if so desired, shall then be based on these new namespaces.

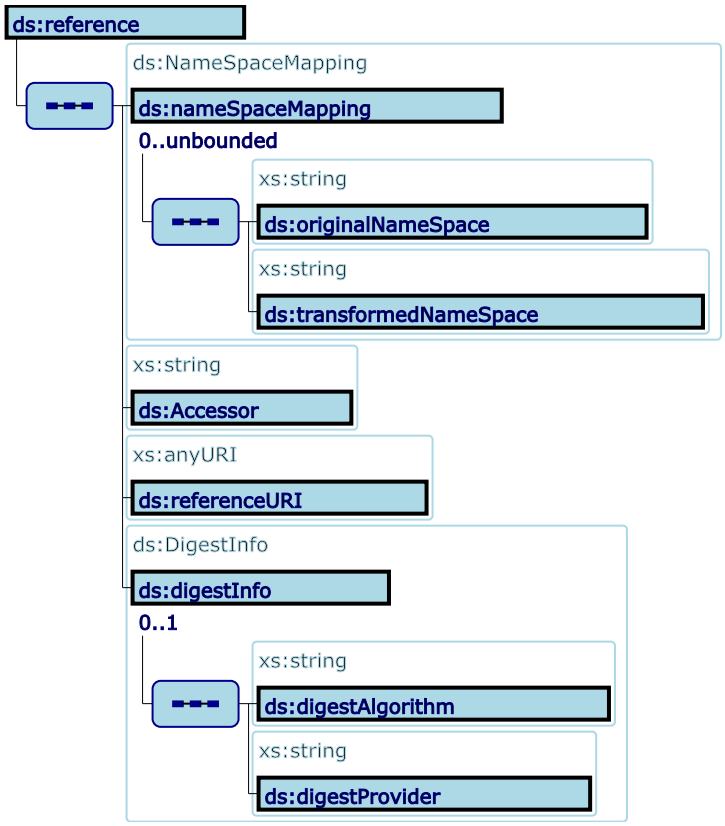

Figure 4: Model for a reference documentation style

`/ds:reference`

This is the root element for the description of the reference documentation style.

`/ds:reference/ds:nameSpaceMapping`

This element is a list mapping namespace URIs in the original XML document to namespace URIs in the transformed XML document and must be provided if the original document contains any namespace URIs. Such mapping must encompass all namespaces in the original document in order

for the transformed document to be meaningful. The namespace mapping must be applied in an equivalent manner for a reverse transformation to produce the original document.

`/ds:reference/ds:Accessor`

This element is used to specify the appropriate part of the input that the reference transformation shall operate on.

`/ds:reference/ds:referenceURI`

This element shall provide the URI for the location where the referenced contents can be found.

`/ds:reference/ds:digestInfo`

This optional element specifies that a digest operation is computed on the referenced contents using the digest algorithm value specified in the `digestAlgo` and the digest provider value specified in the `digestProvider` elements. If not provided, no digest is computed.

The schema for the new elements introduced into the transformed XML document is shown below, with the prefix rd.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
targetNamespace="http://www.gridprovenance.org/documentationstyle/referenceOutput"
elementFormDefault="qualified" attributeFormDefault="unqualified"
xmlns:rd="http://www.gridprovenance.org/documentationstyle/referenceOutput"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="referenceURI" type="xs:anyURI"/>
  <xs:element name="referenceDigest" type="xs:string"/>
</xs:schema>
```

3.4 Signature Documentation Style

The security-signing documentation style (Figure 5) denotes a transformation of an input by which a part of (or the whole of) its contents has been signed. The signature itself shall appear as a new element in the transformed output.

`/ds:sign`

This is the root element for the description of the signature documentation style.

`/ds:sign/ds:nameSpaceMapping`

This element is a list mapping namespace URIs in the original XML document to namespace URIs in the transformed XML document and must be provided if the original document contains any namespace URIs. Such

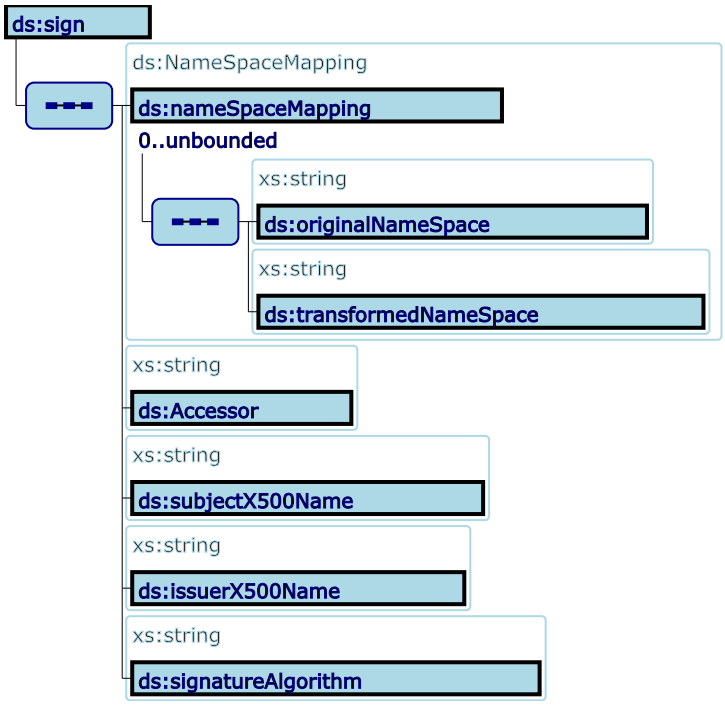

Figure 5: Model for a signature documentation style

mapping must encompass all namespaces in the original document in order for the transformed document to be meaningful. The namespace mapping must be applied in an equivalent manner for a reverse transformation to produce the original document.

`/ds:sign/ds:Accessor`

This element is used to specify the appropriate part of the input that the signature transformation shall operate on.

`/ds:sign/ds:subjectX500Name`

This element is used to specify the subject X500 distinguished name of the X509 certificate [HFPS99] used in the signature operation.

`/ds:sign/ds:issuerX500Name`

This element is used to specify the issuer X500 distinguished name of the X509 certificate [HFPS99] used in the signature operation.

`/ds:sign/ds:signatureAlgorithm`

This element is used to specify the signature algorithm used.

For an XML document, signing should be performed using the XML Signature Recommendation [BBF+02], and the corresponding schema for the new signature element in the transformed output is given at `http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd`.

3.5 Encryption Documentation Style

The encryption documentation style (Figure 6) denotes a transformation of an input by which a part of (or the whole of) its contents has been encrypted. The encrypted content shall appear in replacement of the original content in the transformed output.

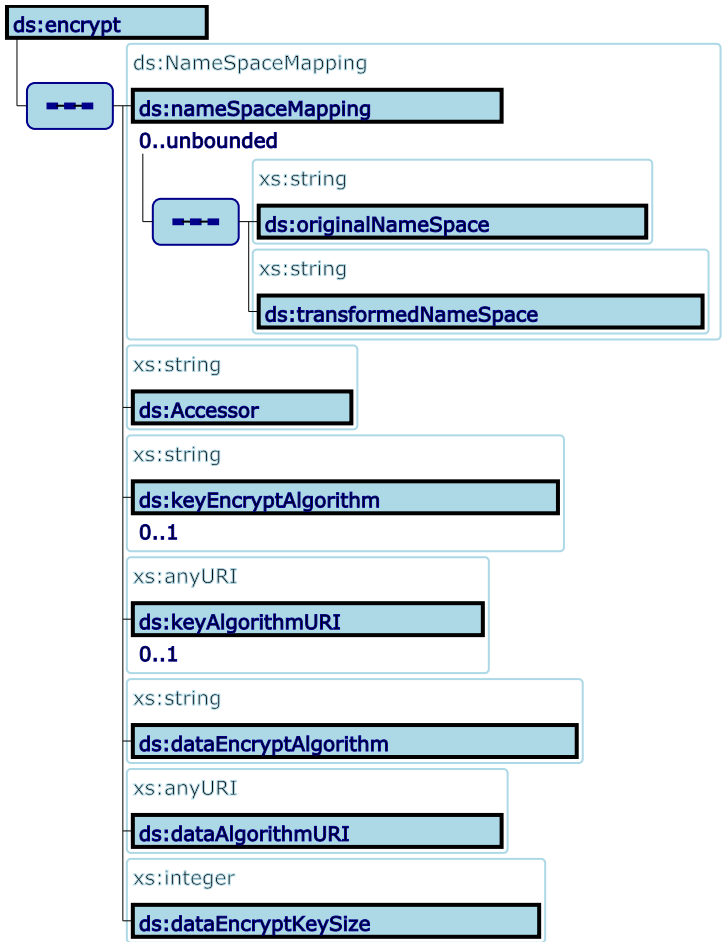

Figure 6: Model for an encryption documentation style

`/ds:encrypt`

This is the root element for the description of the encryption documentation style.

`/ds:encrypt/ds:nameSpaceMapping`

This element is a list mapping namespace URIs in the original XML document to namespace URIs in the transformed XML document and must be provided if the original document contains any namespace URIs. Such mapping must encompass all namespaces in the original document in order for the transformed document to be meaningful. The namespace mapping must be applied in an equivalent manner for a reverse transformation to produce the original document.

`/ds:encrypt/ds:Accessor`

This element is used to specify the appropriate part of the input that the encryption transformation shall operate on.

`/ds:encrypt/ds:keyEncryptAlgorithm`

This element is used to specify the encryption algorithm to be applied to the secret key used in the encryption operation.

`/ds:encrypt/ds:keyAlgorithmURI`

This element is used to specify the URI describing the previous algorithm.

`/ds:encrypt/ds:dataEncryptAlgorithm`

This element is used to specify the encryption algorithm to be applied to the designated contents to be encrypted in the encryption operation.

`/ds:encrypt/ds:dataAlgorithmURI`

This element is used to specify the URI describing the encryption algorithm above.

`/ds:encrypt/ds:dataEncryptKeySize`

This element is used to specify the size of the secret key used in encrypting the designated contents.

For an XML document, the encryption should be performed using the XML Encryption Recommendation [IDS02], and the corresponding schema for the new signature element in the transformed output is given at `http://www.w3.org/TR/xmlenc-core/xenc-schema.xsd`.

3.6 Composite Sequence Documentation Style

A composite sequence documentation style denotes a sequence of documentation styles or transformations that is applied in succession to a given input (message or actor state information). These transformations include the previously discussed documentation styles (encryption, signature, reference and verbatim), and may also include the composite style itself. The output of a transformation in a sequence becomes input to the next transformation in that same sequence, so that the final output of the documentation style is an initial input that has undergone all transformations specified within the sequence. Conversely, a reverse transformation involves the application of all the transformations in a sequence in a reversed order on a transformed input.

Figure 7: Model for composite sequence documentation style

`/ds:compositeSequence`

This is the root element for the description of the composite documentation style.

`/ds:compositeSequence/ds:transformOperation`

This element is a sequence of documentation style transformations unbounded in length, which may additionally include other composite sequence transformations within it.

3.7 Example

We demonstrate how documentation style transformations affect a given input that is an XML document using an example involving the a composite sequence transformation that encapsulates an encryption and reference documentation style. Consider two actors that exchange a single XML structured message between each other as shown in Figure 8.

This original message can optionally be validated against a schema prior to performing transformation to ensure that errors do not arise from attempting to transform an incorrectly typed input. An example of such a schema for the above message is shown in Figure 9.

Consider now a composite transformation involving an encryption documentation style followed by a reference documentation style to be enacted upon this

```
<?xml version="1.0" encoding="UTF-8"?>
<pd:author xmlns:pd="http://www.myexample.com/personaldetails"
xmlns:ad="http://www.myexample.com/addressdetails">

  <pd:firstname>John</pd:firstname>
  <pd:lastname>Doe</pd:lastname>
  <ad:address>
    <ad:street>123 High Street</ad:street>
    <ad:city>Gotham City</ad:city>
  </ad:address>
  <pd:biography>He led an undistinguished life explained in 200 pages</pd:biography>

</pd:author>
```

Figure 8: Original XML message

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.myexample.com/personaldetails"
elementFormDefault="qualified" attributeFormDefault="unqualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ad="http://www.myexample.com/addressdetails">

  <xs:element name="author">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="firstname" type="xs:string"/>
        <xs:element name="lastname" type="xs:string"/>
        <xs:element ref="ad:address" />
        <xs:element name="biography" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

Figure 9: Original XML message schema

sample message in order to produce a transformed message, that will become the content of an interaction p-assertion. The first transformation specifies that the contents of the **pd:lastname** element be encrypted. The next transformation specifies that the contents of the **pd:biography** element be removed and stored at a public repository, and the URI pointing to this stored content is placed in the transformed message along with a digest initially computed on the removed contents. The transformation definition document that specifies this composite transformation itself is shown in Figure 10.

The **Accessor** element within both transformations specifies XPath expressions that identify the specific element within the original message that would be operated upon. Also, the two namespaces present in the original message are now mapped to new namespaces through the namespace mapping specified in both transformations. The transformed message is shown in Figure 11.

This transformed message can optionally be validated against an output schema if one is provided. This output schema for the transformed message can be

```
<transformDefinition xmlns="http://www.gridprovenance.org/documentationstyle"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:pds="http://www.gridprovenance.org/documentationstyle">

  <inputTechnology>XML</inputTechnology>
  <outputTechnology>XML</outputTechnology>

  <transformOperation xsi:type="CompositeSequence">

    <transformOperation xsi:type="Encrypt">

      <nameSpaceMapping>
        <originalNameSpace>http://www.myexample.com/personaldetails</originalNameSpace>
        <transformedNameSpace>http://www.myexample.com/personaldetails/enc</transformedNameSpace>
      </nameSpaceMapping>
      <nameSpaceMapping>
        <originalNameSpace>http://www.myexample.com/addressdetails</originalNameSpace>
        <transformedNameSpace>http://www.myexample.com/addressdetails/enc</transformedNameSpace>
      </nameSpaceMapping>
      <Accessor>author/lastname</Accessor>
      <keyEncryptAlgorithm>DESede</keyEncryptAlgorithm>
      <keyAlgorithmURI>http://www.w3.org/2001/04/xmlenc#kw-tripledes</keyAlgorithmURI>
      <dataEncryptAlgorithm>AES</dataEncryptAlgorithm>
      <dataAlgorithmURI>http://www.w3.org/2001/04/xmlenc#aes128-cbc</dataAlgorithmURI>
      <dataEncryptKeySize>128</dataEncryptKeySize>

    </transformOperation>


    <transformOperation xsi:type="Reference">

      <nameSpaceMapping>
        <originalNameSpace>http://www.myexample.com/personaldetails/enc</originalNameSpace>
        <transformedNameSpace>http://www.myexample.com/personaldetails/enc/ref</transformedNameSpace>
      </nameSpaceMapping>
      <nameSpaceMapping>
        <originalNameSpace>http://www.myexample.com/addressdetails/enc</originalNameSpace>
        <transformedNameSpace>http://www.myexample.com/addressdetails/enc/ref</transformedNameSpace>
      </nameSpaceMapping>
      <Accessor>author/biography</Accessor>
      <referenceURI>http://www.mystorage.com/biography.txt</referenceURI>
      <digestInfo>
        <digestAlgorithm>MD5</digestAlgorithm>
        <digestProvider>Java</digestProvider>
      </digestInfo>

    </transformOperation>

  </transformOperation>

</transformDefinition>
```

Figure 10: Transformation definition document

```
<?xml version="1.0" encoding="UTF-8"?>

<pd:author xmlns:pd="http://www.myexample.com/personaldetails/enc/ref"
xmlns:ad="http://www.myexample.com/addressdetails/enc/ref"
xmlns:rd="http://www.gridprovenance.org/documentationstyle/referenceOutput"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">

  <pd:firstname>John</pd:firstname>
  <pd:lastname>

    <xenc:EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Content">
      <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc" />

      <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
        <xenc:EncryptedKey>
          <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#kw-tripledes"/>
            <xenc:CipherData>
              <xenc:CipherValue>
                dbZHVtHrAXoWDX3awB0G7RAYWd/lMgQz4o4B+wEh4yg=
              </xenc:CipherValue>
            </xenc:CipherData>
        </xenc:EncryptedKey>
      </KeyInfo>

      <xenc:CipherData>
        <xenc:CipherValue>
          g9Fe6ppIg4aN0wpxPA5KFUz+==
        </xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>

  </pd:lastname>
  <ad:address>
    <ad:street>123 High Street</ad:street>
    <ad:city>Gotham City</ad:city>
  </ad:address>
  <pd:biography>
    <rd:referenceURI>http://www.mystorage.com/biography.txt</rd:referenceURI>
    <rd:referenceDigest>1nIldFXC2BVSbqE9njc6gQ==</rd:referenceDigest>
  </pd:biography>

</pd:author>
```

Figure 11: Transformed XML message

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.myexample.com/personaldetails/enc/ref"
elementFormDefault="qualified" attributeFormDefault="unqualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ad="http://www.myexample.com/addressdetails/enc/ref"
xmlns:rd="http://www.gridprovenance.org/documentationstyle/referenceOutput"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">


  <xs:element name="author">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="firstname" type="xs:string"/>
        <xs:element name="lastname">
          <xs:complexType>
            <xs:sequence>
             <xs:element ref="xenc:EncryptedData"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>

        <xs:element ref="ad:address" />
        <xs:element name="biography">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="rd:referenceURI"/>
              <xs:element ref="rd:referenceDigest"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>

      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

Figure 12: Transformed XML message schema

predefined prior to runtime by the application user, or generated at runtime using the schema for the original message and the transformation definition document. Such an output schema should contain the new namespaces for the untransformed elements as well reference schemas for the new elements introduced as a result of the transformation. An example of such an output schema, which incorporates the predefined schema for reference elements (Section 3.3), as well as the schema for XML Encryption (`http://www.w3.org/TR/xmlenc-core/xenc-schema.xsd`) is shown in Figure 12.

The schema for the original and transformed message can be stored in a public repository in order for an actor to retrieve them and use them in a reverse transformation when processing interaction p-assertions. Optionally, an actor may also construct its own output schema for validation purposes using the namespace mapping provided in the transformation definition document.

4 Conclusion

In this document, we have presented a data model for some standard documentation style transformations that can be applied to either a message or actor state in order to produce a transformed output that becomes the content of an interaction p-assertion or actor state p-assertion. An example is provided to illustrate how the model can be used on XML type documents. This model is intended as a complement to the process documentation data model [MGJ+06], which describes the logical organisation of process documentation as well as models of different forms of p-assertions

A Schema for transformation definition document

Below we give the full schema for a transformation definition document that is structured in XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.gridprovenance.org/documentationstyle"
elementFormDefault="qualified" attributeFormDefault="unqualified"
xmlns:ds="http://www.gridprovenance.org/documentationstyle"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="transformDefinition" type="ds:TransformDefinition"/>
  <xs:element name="transformOperation" type="ds:TransformOperation"/>
  <xs:element name="nameSpaceMapping" type="ds:NameSpaceMapping"/>
  <xs:element name="verbatim" type="ds:Verbatim"/>
  <xs:element name="sign" type="ds:Sign"/>
  <xs:element name="encrypt" type="ds:Encrypt"/>
  <xs:element name="reference" type="ds:Reference"/>
  <xs:element name="compositeSequence" type="ds:CompositeSequence"/>
  <xs:complexType name="TransformDefinition">
    <xs:sequence>
      <xs:element name="inputTechnology" type="xs:string"/>
      <xs:element name="outputTechnology" type="xs:string"/>
      <xs:element ref="ds:transformOperation"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="TransformOperation" abstract="true"/>
  <xs:complexType name="Verbatim">
    <xs:complexContent>
      <xs:extension base="ds:TransformOperation"/>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="Sign">
    <xs:complexContent>
      <xs:extension base="ds:TransformOperation">
        <xs:sequence>
          <xs:element ref="ds:nameSpaceMapping" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="Accessor" type="xs:string"/>
          <xs:element name="subjectX500Name" type="xs:string"/>
          <xs:element name="issuerX500Name" type="xs:string"/>
          <xs:element name="signatureAlgorithm" type="xs:string"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="Encrypt">
    <xs:complexContent>
      <xs:extension base="ds:TransformOperation">
        <xs:sequence>
          <xs:element ref="ds:nameSpaceMapping" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="Accessor" type="xs:string"/>
          <xs:element name="keyEncryptAlgorithm" type="xs:string" minOccurs="0"/>
          <xs:element name="keyAlgorithmURI" type="xs:anyURI" minOccurs="0"/>
          <xs:element name="dataEncryptAlgorithm" type="xs:string"/>
          <xs:element name="dataAlgorithmURI" type="xs:anyURI"/>
          <xs:element name="dataEncryptKeySize" type="xs:integer"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="Reference">
```

```
    <xs:complexContent>
      <xs:extension base="ds:TransformOperation">
        <xs:sequence>
          <xs:element ref="ds:nameSpaceMapping" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="Accessor" type="xs:string"/>
          <xs:element name="referenceURI" type="xs:anyURI"/>
          <xs:element name="digestInfo" type="ds:DigestInfo" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="CompositeSequence">
    <xs:complexContent>
      <xs:extension base="ds:TransformOperation">
        <xs:sequence>
          <xs:element ref="ds:transformOperation" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="NameSpaceMapping">
    <xs:sequence>
      <xs:element name="originalNameSpace" type="xs:string"/>
      <xs:element name="transformedNameSpace" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="DigestInfo">
    <xs:sequence>
      <xs:element name="digestAlgorithm" type="xs:string"/>
      <xs:element name="digestProvider" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

References

[BBF+02] Mark Bartel, John Boyer, Barb Fox, Brian LaMacchia, and Ed Simon. XML Signature Syntax and Processing. http://www.w3.org/TR/xmldsig-core/, 2002.

[Bra97] Scott Bradner. Key words for use in RFCs to indicate requirement levels. http://www.ietf.org/rfc/rfc2119.txt, 1997.

[HFPS99] R. Housley, W. Ford, W. Polk, and D. Solo. Request For Comment: Internet X.509 Public Key Infrastructure Certificate and CRL Profile. http://www.ietf.org/rfc/rfc2459.txt, 1999.

[IDS02] Takeshi Imamura, Blair Dillaway, and Ed Simon. XML Encryption Syntax and Processing. http://www.w3.org/TR/xmlenc-core/, 2002.

[MGJ+06] Steve Munroe, Paul Groth, Sheng Jiang, Simon Miles, Victor Tan, and Luc Moreau. Data model for Process Documentation. Technical report, University of Southampton, June 2006.

[TGJ+06] Victor Tan, Paul Groth, Sheng Jiang, Simon Miles, Steve Munroe, and Luc Moreau. WS Provenance Glossary. Technical report, Electronics and Computer Science, University of Southampton, 2006.

[W3C99] W3C. XML Path Language (XPath) Version 1.0. W3C Recommendation 16 November 1999. http://www.w3.org/TR/xpath, 1999.