

Tara Metamodeling Ontology

V1.1

Author:

Goran Zugic

goran.zugic@semantion.com

Copyright © 2008-2009 Semantion Inc.

Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from Semantion (<http://www.semantion.com>).

1.0 Introduction

Ontology defines a set of representational primitives that are used to model a domain of knowledge. Definitions of the representational primitives include information about their meaning, properties, and rules for their consistent use if needed.

Tara Metamodeling Ontology (Tara) provides a generic ontological foundation for metamodeling of business, social, and technology related processes and systems.

Tara is designed to be used by software that process information provided in Tara format and for this use Tara is interpreted via Tara XML schema (<http://www.semantion.com/documentation/SBP/metamodeling/xsd/tara-1.1.xsd>). Tara can also be used for presenting metamodeling information in more human friendly form and for this purpose Tara Language (TL) is used.

Semantion Metamodeler (SM2) is a tool that provides full metamodel definitions in Tara.

The first release of Tara Metamodeling Ontology was published in December 2008:

http://www.semantion.com/documentation/SBP/metamodeling/TaraMetamodelingOntology_V1.0.pdf

This is the second release that contains new elements regarding association and concept rules, semantic web descriptions, inheritance, inference, and transitivity.

2.0 Tara Metamodeling Ontology Elements

The elements of Tara ontology are:

- **Association**
- **AssociationRule**
- **Attribute**
- **Classification**
- **Concept**
- **ConceptRule**
- **InformationalReference**
- **Metamodel**
- **Node**
- **Scheme**
- **SemanticDescription**

2.1 Association

Association models an association between two **Concepts**. **Association** has following properties:

Property Name	Property Description	Is property mandatory?
<i>node</i>	Association scheme node that belongs to the association. Association scheme contains all nodes representing association types used in the metamodel.	Yes
<i>associationRules</i>	List of association rules that specify constraints under which the association can be used.	Yes

2.2 AssociationRule

AssociationRule models a rule for an Association. To be functional the rule specifies source concept type and target concept type of the association at least. **AssociationRule** has following properties:

Property Name	Property Description	Is property mandatory?
<i>name</i>	Name of the AssociationRule	Yes
<i>description</i>	Description of the AssociationRule	No
<i>type</i>	The type of the rule. If the <i>Tara</i> type is specified this rule is used. If any other type is specified then the rule represented by either property <i>content</i> or property <i>rule</i> below is used.	Yes
<i>content</i>	A rule content written in a rule language specified by the <i>type</i> property. If this property is used the <i>reference</i> property is ignored.	No
<i>reference</i>	An informational reference of a document containing the association rule. If this reference is specified then the rest of the rule properties listed below are ignored.	No
<i>sourceType</i>	Type of the source concept in the association.	No
<i>sourceAttribute</i>	Source concept attribute (property).	No
<i>sourceValues</i>	List of source concept attribute values.	No
<i>targetType</i>	Type of the target concept in the association.	No
<i>targetAttribute</i>	Target concept attribute (property).	No
<i>targetValues</i>	List of target concept attribute values	No

The source concept type and target concept type are mandatory properties for the **AssociationRule** to be functional. They specify source concept and target concept in the association. Additional constraints can be added via attributes (properties) and their values. For example, we can have concept A and B where the concept A is the source concept and concept B is target concept in the association. By using the rule we can also specify that concept A and B can be associated only if a

specific attribute of the concept A (*sourceAttribute* property of the **AssociationRule**) has required list of values (*sourceValues* property of the **AssociationRule**) and a specific attribute of the concept B (*targetAttribute* property of the **AssociationRule**) has required list of values (*targetValues* property of the **AssociationRule**).

Any number of **AssociationRules** can be defined for an **Association**.

An **AssociationRule** can be defined in another language as well and it is referenced by the *rule* property that is an informational reference for a document containing the rule.

2.3 Attribute

Attribute models additional **Concept's properties**. These properties are any properties besides the default **Concept's** properties that include **name**, **description**, and **nodes** where **nodes** represents a list of all concept type nodes which the attribute belongs to. **Attribute** has following properties:

Property Name	Property Description	Is property mandatory?
<i>name</i>	Name of the Attribute	Yes
<i>description</i>	Description of the Attribute	No
<i>values</i>	List of property values. Default values can be specified.	No
<i>type</i>	Attribute's type (i.e., Integer, String, etc.)	No
<i>option</i>	A list of fixed pre-defined optional values	No
<i>required</i>	Specifies if the value for the property is mandatory (yes/no). If the value for this property is not specified the default value is "no".	No

2.4 Classification

Classification classifies **Concepts** based on common characteristics. **Classification** has following properties:

Property Name	Property Description	Is property mandatory?
<i>name</i>	Name of the classification	Yes
<i>description</i>	Description of the classification	No
<i>node</i>	Classification node	No
<i>concept</i>	Concept that is associated with the node via this classification	No

2.5 Concept

Concept models any physical or abstract thing that can have a computerized representation.

Property Name	Property Description	Is property mandatory?
<i>node</i>	Node that represents concept's type. This node is under the scheme that includes all concept type nodes.	Yes
<i>attributes</i>	List of concept's attributes	No
<i>conceptRules</i>	List of concept's rules	No

Node, attributes, and conceptRules are Tara constructs used to define a Concept. Node itself defines Concept's type specified by the name (type name) and description. *name* and *description* are default properties of Concept's instances. Additional properties can be added via Tara attributes.

2.6 ConceptRule

ConceptRule models a rule for a Concept. **ConceptRule** has following properties:

Property Name	Property Description	Is property mandatory?
<i>name</i>	Name of the ConceptRule	Yes
<i>description</i>	Description of the ConceptRule	No
<i>type</i>	The type of the rule.	Yes
<i>content</i>	A rule content written in a rule language specified by the <i>type</i> property. If this property is used the <i>reference</i> property is ignored.	No
<i>reference</i>	An informational reference of a document containing the concept rule.	No

2.7 InformationalReference

InformationalReference models a detailed reference to a document. **InformationalReference** has following properties:

Property Name	Property Description	Is property mandatory?
<i>name</i>	Name of the InformationalReference	Yes
<i>description</i>	Description of the InformationalReference	No
<i>node</i>	Node that represents document type	Yes
<i>value</i>	Document's reference	No
<i>version</i>	Version of the document represented by this InformationalReference	No
<i>time</i>	Time when InformationalReference is confirmed. The InformationalReference is confirmed when document becomes available. At that time, the <i>value</i> attribute of the InformationalReference gets the URI value representing the location of the document.	No
<i>author</i>	Author of the document which metadata is represented by the InformationalReference	No
<i>title</i>	The title of the document	No

2.8 Metamodel

Metamodel models a metamodel containing all concepts, associations, and rules needed to model processes, systems, and other things in business and social environments. Metamodel has following properties:

Property Name	Property Description	Is property mandatory?
<i>name</i>	Name of the Metamodel	Yes
<i>description</i>	Description of the Metamodel	No
<i>conceptScheme</i>	Scheme that defines all concept types	No
<i>associationScheme</i>	Scheme that defines all association types	No
<i>optionSchemes</i>	List of schemes that define all options for attributes	No
<i>concepts</i>	List of concepts	No
<i>associations</i>	List of associations	No

2.9 Node

Node models a node under the classification scheme. **Node** has following properties:

Property Name	Property Description	Is property mandatory?
<i>name</i>	Name of the node	Yes
<i>description</i>	Description of the node	No
<i>parent</i>	Parent scheme's reference	No

2.10 Scheme

Scheme is a root node in a scheme of nodes. **Scheme** has following properties:

Property Name	Property Description	Is property mandatory?
<i>name</i>	Name of the scheme	Yes
<i>description</i>	Description of the scheme	No
<i>type</i>	Type of the scheme that reflects the purpose of the scheme (Association/Concept/Option/Type/Document).	No
<i>nodes</i>	A list of nodes that belongs to the scheme	No

2.11 SemanticDescription

SemanticDescription models a formal (RDF, OWL, etc.) or informal description of a Concept or an Association.

Property Name	Property Description	Is property mandatory?
<i>name</i>	Name of the semantic description.	Yes
<i>description</i>	Description of the semantic description.	No
<i>type</i>	Type of the SemanticDescription (i.e., RDF, OWL, etc.)	Yes
<i>content</i>	Semantic description content.	No
<i>reference</i>	An informational reference of a document containing the semantic description.	No

3.0 Inheritance, Inference, and Transitivity

In Tara, inheritance can be defined via "Inherits" association with a rule where the source concept type represents child concept while the target concept type represents parent concept.

Associations between concepts makes it possible to use logic to discover new information/knowledge. Concepts can be linked together either directly or indirectly. An asserted association is a direct association between two concepts. For example, "*B Inherits A*" means that concept B inherits all properties of concept A. Inferred associations are indirect associations between two concepts. These associations can be found by tracking associations between two concepts across the intervening concepts and associations which include multiple association types. Novel inferred associations can be found by traversing the asserted associations via association rules that govern the transitivity of associations in a metamodel.

4.0 Metamodel Definition

The definition of a metamodel based on Tara includes following steps:

1. Create a metamodel
2. Define concept types
3. Define association types
4. Define attribute types
5. Define optional property values (options) if needed
6. Define additional properties for concepts where needed
7. Define association rules
8. Define concept rules
9. Define document types

Tara Language (TL) or Tara XML schema can be used for formal metamodel definitions. We will use formal model definitions in this document. Second part of this document will also show how Semantion Modeler can be used to create metamodels in Tara.

After creating a metamodel, the concept types have to be defined first.

If the metamodel contains associations, association types have to be modeled as well.

Define types that will be used as data types for attributes.

For properties that have pre-defined fixed optional values, these values have to be modeled too.

Attributes have to be defined for all concepts that require additional properties besides the default properties (name and description).

We also have to define association rule for each association used.

If concept rules are needed they will be defined as well.

Finally, if documents are used in a process/system metamodel, document types have to be defined.

4.1 Metamodel Definition Examples in Tara

Three examples of different complexity will be presented in this section. They are coded in Tara Language.

First example demonstrates how to model a customer with his/her contact information.

Second example shows how more complex association rules can be modeled.

The last example shows how to model decisions in business processes.

4.1.1 Customer

We will use four concepts to model a customer: Customer, PostalAddress, EmailAddress, and TelephoneNumber. By following Tara metamodeling steps we will define concepts, associations between them, optional properties' values, and additional properties via attributes, and attribute types. Document types will not be defined since they are not needed in this example.

4.1.1.1 Customer Concept Types Definitions

```
ConceptScheme {  
  name: Customer Concepts; description: This is a classification scheme for concept types;  
  {name: Customer; description: Customer type; }  
  {name: PostalAddress; description: PostalAddress type; }  
  {name: EmailAddress; description: EmailAddress type;}  
  {name: TelephoneNumber; description: TelephoneNumber type;}  
}
```

4.1.1.2 Customer Association Types Definitions

```
AssociationScheme {  
  name: Customer Associations;  
  description: This is a classification scheme for association types;  
  
  {name: IsPostalAddressOf; description: Models association between PostalAddress and Customer;  
    rule: { name: PostalAddress-Customer;  
      description: Defines a rule for association between PostalAddress and Customer;  
      sourceType: PostalAddress; targetType: Customer;}  
  }  
  
  {name: IsEmailAddressOf; description: Models association between EmailAddress and Customer;  
    rule: { name: EmailAddress-Customer;  
      description: Defines a rule for association between EmailAddress and Customer;  
      sourceType: EmailAddress; targetType: Customer;}  
  }  
  
  {name: IsTelephoneNumberOf; description: Models association between TelephoneNumber and Customer;  
    rule: { name: TelephoneNumber-Customer;  
      description: Defines a rule for association between TelephoneNumber and Customer;  
      sourceType: TelephoneNumber; targetType: Customer;}  
  }  
}
```

```
}  
}
```

4.1.1.3 Optional Property Values Definitions

OptionSchemes {

{name: LocationValues;

description: This is a classification scheme for the PostalAddress, EmailAddress, and
ContactNumber location optional values;

{name: Office; **description:** Office location;}

{name: Home; **description:** Home location;}

}

{name: TelephoneNumberTypeValues;

description: This is a classification scheme for the telephone number type values;

{name: Phone; **description:** Land phone;}

{name: Cell; **description:** : Cell phone;}

{name: Fax; **description:** : Fax;}

}

{name: StateProvinceValues; **description:** This is a classification scheme for the State/Province values;

{name: Alberta;}

{name: British Columbia;}

{name: Manitoba;}

{name: New Brunswick;}

{name: Newfoundland and Labrador;}

{name: Northwest Territories;}

{name: Nova Scotia;}

{name: Nunavut;}

{name: Ontario;}

{name: Prince Edward Island;}

{name: Quebec;}

{name: Saskatchewan;}

{name: Yukon;}

}

}

4.1.1.4 Additional Properties Definitions

Attributes {

{name: firstName; **description:** Customer's first name;

required: yes;

type: String;

concepts: [Customer];}

{**name**: middleName; **description**: Customer's middle name;
type: String;
concepts: [Customer];}

{**name**: lastName; **description**: Customer's last name;
required: yes;
type: String;
concepts: [Customer];}

{**name**: street; **description**: Street name;
required: yes;
type: String;
concepts: [PostalAddress];}

{**name**: streetNumber; **description**: Street number;
required: yes;
type: String;
concepts: [PostalAddress];}

{**name**: city; **description**: City name;
required: yes;
type: String;
concepts: [PostalAddress];}

{**name**: stateOrProvince; **description**: State/Province name;
option: StateProvinceValues;
required: yes;
concepts: [PostalAddress];}

{**name**: postalCode; **description**: Postal code;
required: yes;
type: String;
concepts: [PostalAddress];}

{**name**: country; **description**: Country name;
option: CountryValues;
required: yes;
type: String;
concepts: [PostalAddress];}

{**name**: address; **description**: Email address;
concepts: [EmailAddress];}

{**name**: number; **description**: Telephone number;
required: yes;
type: String;
concepts: [TelephoneNumber];}

Semantion Inc.

```
{name: locationType; description: Location type;  
  option: LocationValues;  
  concepts: [EmailAddress, TelephoneNumber];}  
  
{name: type; description: Telephone number type;  
  option : TelephoneNumberTypeValues;  
  concepts: [TelephoneNumber];}  
}
```

We do not list all attributes here. We listed just few of them to show how they can be created.

4.1.1.5 Customer Modeling in Tara

At this point we have a small metamodel for modeling information about our customers. Now, we are ready to model the "John Smith" customer using the metamodel we just created:

```
Concepts {  
  { name: John Smith; description: A customer;  
    node: Customer; firstName: John; middleName; lastName: Smith;  
    { PostalAddress #IsPostalAddressOf;  
      PhoneNumber #IsTelephoneNumberOf;  
      FaxNumber #IsTelephoneNumberOf;  
      EmailAddress #IsEmailAddressOf;  
    }  
  }  
  
  { name: PostalAddress; description: John Smith's postal address;  
    node: PostalAddress; streetNumber: 15; street: Lake Avenue;  
    city: Toronto; stateOrProvince: Ontario;  
    postalCode: M9A 2C2; country: Canada;  
  }  
  
  { name: PhoneNumber; description: John Smith's phone number;  
    node: TelephoneNumber; number: 416-123-4567; type: Phone;  
  }  
  
  { name: FaxNumber; description: John Smith's fax number;  
    node: TelephoneNumber; number: 416-123-5555; type: Fax;  
  }  
  
  { name: EmailAddress; description: John Smith's email address;  
    node: EmailAddress; address: john.smith@semantion.com;  
  }  
}
```

4.1.2 Association Rule

Each association needs at least one rule that defines its source concept type and target concept type. However, sometimes additional constraints need to be specified. For example, let us assume that we have two concepts, C1 and C2 that need to be associated with "RelatesTo" association type where C1 is the source concept in the association and C2 is the target concept in the association. Besides name and description property C1 also has two additional properties, *c1p1* and *c1p2*. C2 concept has also two additional properties *c2p1* and *c2p2*.

We need to define a following rule: C1 will be associated with C2 only if C1's property modeled with attribute *c1p1* has value *v1* and C2's property modeled with attribute *c2p1* has value *v2*. This is how the association definition will look in TL:

```

AssociationScheme {
  name: Associations;
  description: This is a classification scheme for association types. Only one association type "RelatesTo" is
    defined for now.

  {name: RelatesTo; description: Models association between C1 and C2;
    rule: {sourceType: C1; targetType: C2;
      sourceAttribute: c1p1; sourceValue: [v1];
      targetAttribute: c2p1; targetValue: [v2];
    }
  }
}

```

4.1.3 Decision

We will define only a small fragment of the SOA Information Model (SOA-IM) metamodel [1] that will be used to model business process decisions.

In order to define (create) a metamodel (a small fragment of SOA-IM in this example) that will be used to model decisions in business processes, we will follow five Tara metamodeling steps. The definitions specified in these steps are generic for the entire metamodel. They are done just once during this definition phase of the metamodel and then they are used to model entities supported by the metamodel.

Before we define a small part of the SOA-IM metamodel that will be used to model decisions in business processes, we will first extract the Decision definition from the SOA-IM metamodel specification.

A *Decision* is a specific activity in a collaborative process flow that makes choices. *Decisions* are supplied with inputs called *Criteria* and they create outputs called *Choices*. A *Decision* can be made by any of these three participants in a collaborative process: *User* (human), *Service*, or *Agent*.

There are three types of decision outputs (alternatives):

- *Binary*
- *Primary*
- *Derivative*

The *Binary* outputs belong to a choice (e.g., *Yes* or *No*, *True* or *False*, etc.) and only a single output (choice) is generated based on a rule associated with the decision. The *Primary* type enables a selection of one or more outputs (but not all of them) from the provided inputs based on a rule associated with the *Decision*. The *Primary* type outputs provide filtering. The *Derivative* type is like the *Binary* type with a difference that it can generate more than one output based on a rule associated with the *Decision*.

This is a minimized list of Concepts needed to model a *Decision* in SOA-IM:

- *Choice*
- *ChoiceReference*
- *InformationalReference*

Semantion Inc.

- *InputOutput*
- *Decision*
- *Service*

Choice models decision's choices, *ChoiceReference* models a reference of the document that models a choice that can be produced by the decision, *InformationalReference* models a reference to a document associated with an *InputOutput*, *InputOutput* models decision's criteria (inputs), *Decision* concept models decision, and *Service* models a service that will perform the decision. In this example we assume that the *Decision* will be performed by a *Service*. Some concepts and attributes have been removed from the definitions below to make this example simpler and easier to understand.

This is how SOA-IM defines these Concepts:

Choice

Attribute	Type	Description
id	String256	Unique ID
name	String256	Choice's name
description	String4000	Detailed description
alias	String256	The alias of the Choice. For example, if the alias is specified it will be used as a parameter name for a service which the Choice is associated with. Otherwise the name attribute will be used without spaces between the words included in the name.
time	DateTime	Time when Choice is confirmed

Associated with

- A Decision where the Decision is the target object and association type is "IsChoiceOf"

ChoiceReference

Attribute	Type	Description
id	String256	Unique ID
name	String256	ChoiceReference's name
description	String4000	Detailed description
type	String256	Document type (any type)
value	String256	Document's reference
time	DateTime	Time when ChoiceReference is confirmed

Associated with

- A Choice where the Choice is the target object and association type "IsChoiceReferenceFor"

InformationalReference

Attribute	Type	Description
id	String256	Unique ID
name	String256	InformationalReference's name
description	String4000	Detailed description
type	String256	Type of the referenced document (any document type or ChoiceDoc)
value	String256	Document's reference
time	DateTime	Time when InformationalReference is confirmed

Associated with

- A Criterion where the Criterion is the target object and association type is "IsReferenceFor"

Semantion Inc.

- An InputOutput where the InputOutput is the target object and association type is "IsReferenceFor"

InputOutput

Attribute	Type	Description
id	String256	Unique ID
name	String256	The name of the InputOutput.
description	String4000	Detailed description
alias	String256	The alias of the InputOutput. For example, if the alias is specified it will be used as a parameter name for a service which the InputOutput is associated with. Otherwise the name attribute will be used without spaces between the words included in the name.
type	String256	The type of InputOutput (Input/Output/Both)
time	DateTime	Time when InputOutput is confirmed

Associated with

- A Decision where the Decision is the target object and association type is "IsCriterionOf"

Decision

Attribute	Type	Description
id	String256	Unique ID
name	String256	Decision's name
description	String4000	Detailed description
choiceType	String256	The type of the decision's choices (Binary/Primary/Derivative)
timeToComplete	String256	A period of time for which the decision must be completed. If the value for this attribute is not provided the time to complete is unlimited. The value for this attribute is specified using the XSD duration format (e.g., PT1H means one hour).

Associated with

- A CollaborativeProcessFlow where the CollaborativeProcessFlow is the target object and association type is "IsDecisionIn".

Service

Attribute	Type	Description
id	String256	Unique ID
name	String256	Service's name
description	String4000	Detailed description
protocol	String256	The ID of the protocol used to communicate with the service (WSDL, CPPA, etc.).
modelReference	String256	The reference for the document that contains service logic in the original service modeling language format (UML, BPMN, text or other).
rule	String256	The ID of a Rule that the decision type services will use in making a decision.

Associated with

- A Decision where the Decision is the target object and association type is "IsServiceFor"

Now we will provide all definitions needed to define the fragment of SOA-IM for the decisions' metamodeling.

4.1.4.1 Concept Types Definitions

```

ConceptScheme {
  name: SOA-IM Concepts; description: This is a classification scheme for concept types;
  {name: Choice; description: Choice type; }
  {name: ChoiceReference; description: ChoiceReference type; }
  {name: InformationalReference; description: InformationalReference type; }
  {name: InputOutput; description: InputOutput type; }
  {name: Decision; description: Decision type; }
  {name: Service; description: Service type; }
}

```

4.1.4.2 Association Types Definitions

```

AssociationScheme {
  name: SOA-IM Associations; description: This is a classification scheme for association types;
  {name: IsChoiceOf; description: Models association between Decision and Choice;
  rule: {sourceType: Choice; targetType: Decision; }
  }
  {name: IsChoiceReferenceFor; description: Models association between Choice and ChoiceReference;
  rule: { name: ChoiceReference-Choice;
  description: Defines a rule for association between ChoiceReference and Choice;
  sourceType: ChoiceReference; targetType: Choice; }
  }
  {name: IsReferenceFor; description: Models association between InputOutput and InformationalReference;
  rule: { name: InformationalReference - InputOutput;
  description: Defines a rule for association between InformationalReference and InputOutput;
  sourceType: InformationalReference; targetType: InputOutput; }
  }
  {name: IsCriterionOf; description: Models association between Criterion and Decision;
  rule: { name: Criterion - Decision;
  description: Defines a rule for association between Criterion and Decision;
  sourceType: Criterion; targetType: Decision; }
  }
  {name: IsServiceFor; description: Models association between Service and Decision;
  rule: { name: Service - Decision; description: Defines a rule for association between Service and Decision;
  sourceType: Service; targetType: Decision; }
  }
}

```

4.1.4.3 Fixed Optional Property Values Definitions

```

OptionSchemes {
  {name: ChoiceTypeValues;

```

```

description: This is a classification scheme for the choiceType property optional values;
{name: Binary; description: Binary choice type;}
{name: Primary; description: Primary choice type;}
{name: Derivative; description: Derivative choice type;}
}

{name: InputOutputValues;
  description: This is a classification scheme for the InputOutput values;
  {name: Input; description: InputOutput models input only;}
  {name: Output; description: : InputOutput models output only;}
  {name: Both; description: : InputOutput models both input and output;}
}

{name: MimeTypeValues;
  description: This is a classification scheme for the mimeType property optional values;
  {name: text/xml; description: Binary choice type;}
  {name: text/plain; description: Primary choice type;}
  {name: application/msword; description: Derivative choice type;}
  {name: application/pdf; description: Derivative choice type;}
}

{name: ProtocolValues;
  description: This is a classification scheme for the service's protocol property optional values;
  {name: CPID; description: Collaborative Process Information Document based protocol;}
  {name: WSDL; description: Web Services Description Language based protocol;}
  {name: CPPA; description: Collaborative Protocol Profile and Agreement based protocol;}
}
}

```

4.1.4.4 Additional Properties Definitions

Attributes {

```

{name: alias;
  description: The alias of the Choice. For example, if the alias is specified
  it will be used as a parameter name for a service which the Choice
  is associated with. Otherwise the name attribute will be used without
  spaces between the words included in the name.;
  concepts: [Choice, InputOutput];}
{name: choiceType; description: The type of the decision's choices (Binary/Primary/Derivative);
  option: ChoiceTypeValues;
  required: yes;
  concepts: [Choice];}
{name: url; description: The URL of the document that will be accessed on-line;
  concepts: [ChoiceYes, ChoiceNo];}
{name: type; description: InputOutput types;
  option: InputOutputValues;
  required: yes;

```

Semantion Inc.

```
    concepts: [InputOutput];}  
{name: type; description: Reference types;  
  required: yes;  
  concepts: [ChoiceReference, InformationalReference];}  
}
```

We do not list all attributes here. We listed just few of them to show how they can be created.

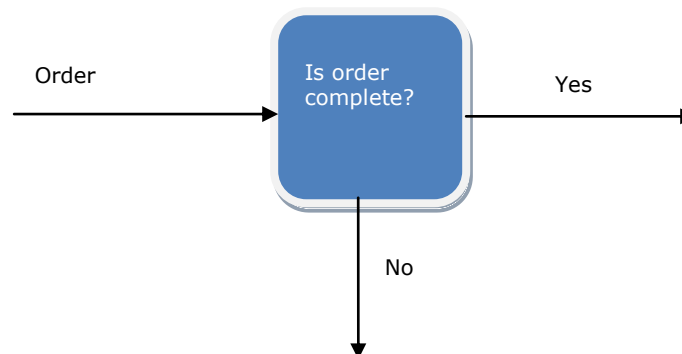
4.1.4.5 Document Types Definitions in Tara

```
DocumentScheme {  
  name: DocumentType;  
  description: This is a classification scheme for document types that will be used with the "Is order complete"  
    decision example;  
  {name: ChoiceYes; description: XML document type that contains "yes" choice;}  
  {name: ChoiceNo; description: XML document type that contains "no" choice;}  
  {name: Order; description: XML document type that contains an order;}  
}
```

4.1.4.6 Decision Modeling in Tara

In the previous sections we defined a metamodel in Tara that will be used to model decisions in business processes.

Now we are ready to model a concrete decision we call "Is order complete?". First we'll model the order via *InputOutput* and *InformationalReference*. During the execution of a process the decision will check the order and if it is correct and complete it will generate the "Yes" choice. Otherwise the generated choice will be "No". This decision is performed by a service.



The following Tara listing represents all Concepts that model the "Is order complete?" decision:

```
Concepts {  
  { name: Order; description: An order that will be processed;  
    node: InputOutput; alias: order;  
    type: Both; time:;  
    { name: OrderInfoRef; description: Order's informational reference;
```

```

        node: InformationalReference; alias: order;
        type: Order; value:;;
    } #IsReferenceFor;
}

{ name: ChoiceYes; description: Concept that represents "Yes" choice;
  node: Choice; alias: ChoiceYes; time:;;
  { name: ChoiceYesRef; description: Choice yes' reference;
    node: ChoiceReference; type: ChoiceYes;
    value:;;
  } #IsChoiceReferenceFor;
}

{ name: ChoiceNo; description: An Concept that represents "No" choice;
  node: Choice; alias: ChoiceNo;
  time:;;
  { name: ChoiceNoRef; description: Choice no's reference;
    node: ChoiceReference; type: ChoiceNo;
    value:;;
  } #IsChoiceReferenceFor;
}

{ name: IsOrderComplete; description: Service that will perform "Is order complete" decision;
  node: Service; protocol: WSDL;
  modelReference:;; rule:;;
}

{ name: Is order complete?;
  description: Checks if an order is complete. If the order is complete makes "Yes" choice. Otherwise
    makes "No" choice. This decision must be completed in 5 minutes;
  node: Decision; choiceType: Binary;
  timeToComplete: PT5M;
  { Order #IsCriterionOf;
    ChoiceYes #IsChoiceOf;
    ChoiceNo #IsChoiceOf;
    IsOrderComplete #IsServiceFor;
  }
}
}
}

```

This model of the "Is order complete" decision is ready for the execution on Semantion SOA Virtual Machine (SOA-VM).