Specification of the TOIF XML schema

Deliverable Version 1.1

October. 1st , 2012

Summary

This document provides specification of the TOIF XML schema – the common vulnerability reporting format. TOIF XML is one of the inputs to the TOIF Analyzer tool that integrates multiple vulnerability findings and the basic facts related to a system under assessment.

This specification describes TOIF schema at three different levels of abstraction:

- Level 1: Conceptual schema

  Describes the conceptual schema of the TOIF XML that addresses the following concerns:

  - TOIF logical concepts
  - TOIF fact-oriented organization
  - TOIF housekeeping concepts

- Level 2: UML Model

  Describes the UML model of the TOIF XML produced by a systematic transformation from the conceptual schema, and describes the automated transformation process by which an XML schema is generated from a UML model.

- Level 3: TOIF XML Schema

  Provides the resulting TOIF XML schema and additionally illustrates the usage of the TOIF XML schema by describing an example TOIF XML data representation that uses the TOIF XML schema.

Table of Contents

Table of Figures

1 INTRODUCTION

TOIF XML (XMI) is a common normalized format for representing the findings of vulnerability detection tools for the purpose of integrating multiple facts related to the system under assessment.

The TOIF XML schema is one of the input formats for the TOIF Analyzer. XML stands for eXtended Markup Language. The acronym XMI stands for XML Metadata Interchange format. XMI is a specific form of XML that is associated with the Model Driven Development approach. XMI has been developed for the purpose of exchanging metadata such as models. XMI is standardized by OMG (current specification is identified as MOF 2.0 / XMI Mapping Specification, v2.1.1, document formal/07-12-01) and ISO (19503:2005).

This specification has the following structure:

- Section 2 “Objectives” summarizes the key design objectives for the TOIF XML format.
- Section 3 “Original Conceptual Model” presents the conceptual schema for TOIF XML developed as part of the Extended Logical Weakness Model.
    - Section 3.1.”Basic Facts” describes the basic common facts related to vulnerability findings.
    - Section 3.2 “Fact-oriented organization of TOIF XML” elaborates the conceptual model and describes the organization of the TOIF XML that facilitates integration.
    - Section 3.3 “Housekeeping considerations for TOIF XML” describes several “housekeeping” facts that facilitate management of multiple TOIF XMI files during the enitre life cycle of the operation of the TOIF framework.
- Section 4 “From conceptual model to UML model” presents the UML model for TOIF XML which is developed as an intermediate step towards the TOIF XML schema.
    - Section 4.1 “Transformation for conceptual schema to a UML model” describes the systematic transformation from the conceptual schema to a UML model.
    - Section 4.2” The logical elements of the TOIF XML” presents 8 UML class diagrams that describe the logical elements of the TOIF XML, the logical entities and fact types.
    - Section 4.3 “The fact-oriented structure of the TOIF XML” presents a single UML class diagram that describes the fact-oriented organization of the TOIF XML.
    - Section 4.4 “The housekeeping elements of the TOIF XML “presents 8 UML class diagrams that describe the housekeeping elements of the TOIF XML.
- Section 5 “From UML model to XML/XMI schema” describes the TOIF XML schema.
    - Section 5.1. “Transformation from a UML model to XML/XMI schema” describes the process of generating the TOIF XMI schema from the UML model presented in section 4. The intermediate specification toif_xmi.ecore (machine-readable representation of the TOIF XMI in Eclipse Ecore) is provided as a separate machine-readable file.Section 5.2

"The TOIF XMI schema" presents the entire TOIF XML/XMI schema produced using the procedure outlined in section 5.1. The TOIF XMI schema is also available in a separate machine readable file. Section 5.3. "Example TOIF XMI data" presents an example TOIF XMI data corresponding to the TOIF XML schema.

2 OBJECTIVES

This section reviews the key design objectives related to the TOIF XMI format. TOIF is a standard vendor-neutral protocol between multiple proprietary vulnerability detection tools and the TOIF Analyzer.

The TOIF analyzer takes the output of a vulnerability detection tool, expressed in the common weakness format (the TOIF XMI) which is integrated into a full KDM model of the system under assessment, and determines which type of a standard CWE weakness it corresponds to. The TOIF analyzer will complete the normalization of the weakness information provided by existing tools.

The TOIF Analyzer constitutes the runtime apparatus of the TOIF, because the TOIF Analyzer physically achieves the normalization of the reports. TOIF Analyzer is based on the method for integrating vulnerabilities detected by 3rd party tools to the Software Assurance Framework by establishing mappings into KDM and Logical Weakness Model. As part of this method it is important that TOIF is itself a normalized fact-oriented protocol which is aligned with other protocols, such as the standard Knowledge Discovery Metamodel (KDM) protocol describing basic facts about the system under assessment, risk analysis interchange protocol, as well as other protocols of the System Assurance Ecosystem, by the Object Management Group (OMG). This alignment establishes a practical integrated environment where multiple tools can contribute software assurance information and facilitates systematic evaluation and measurement of existing vulnerability detection tools.

The key requirement for the TOIF Analyzer is that no modifications to the source code of the original vulnerability detection tools is made, in other words, the integration based on TOIF framework shall include an explicit **adaptation step** that is performed outside of an off-the-shelf vulnerability detection tool, and that is capable of transforming the original report from such tool into a normalized TOIF report that can be the input to the TOIF Analyzer tool for the purpose of semantic integration. This consideration is important to the design of the TOIF Analyzer as it separates the syntactic aspects of integration from the semantic ones.

The integration methodology shall include assurance that a reasonably broad range of vulnerability reports can be integrated (beyond the initial set of tools selected for the proof-of-concept).

The operation of the TOIF Analyzer involves two distinct phases. Phase 1 involves the application of:

- one or more vulnerability detection tools to the system under assessment.
- a KDM extractor tool to the same system under assessment in order to generate the basic KDM facts about the system.

Phase 2 involves the application of the TOIF Analyzer to the vulnerability reports. As the result, an integrated vulnerability report is produced.

Since vulnerability reports are produced by multiple off-the-shelf vulnerability detection tools, phase 1 shall perform normalization of the original reports by tool-specific TOIF Adaptors so that

the TOIF Analyzer can successfully read and process reports in multiple formats and populate the integrated repository.

Evaluation of the adaptation options leads us to believe that the intrusive approach shall be ruled out. Therefore it is required that the TOIF Analyzer shall not depend on the modification of the original vulnerability detection tool. The open description of the normalized report format, TOIF XMI, will encourage the vendors of the commercial vulnerability detection tools to support it natively, however regardless of the adoption by the tool vendors of the original tools, such tools can still be integrated into the framework by third parties providing the adaptation capability.

The second requirement is that the TOIF framework shall define a TOIF XMI format as the output for the adaptation tools. The TOIF XMI format shall normalize the syntax of the original vulnerability reports so that the TOIF Analyzer can focus at the semantic integration of the findings. The TOIF XMI may be positioned as the markup language such that the predefined markers can be inserted into the original report and the TOIF Analyzer will then read the report as it is structured by the markers.

3 CONCEPTUAL MODEL

3.1 BASIC FACTS

The conceptual model of the TOIF XMI has been developed as part of the Enhanced Logical Weakness Model. The original vulnerability findings were considered as partial clues to the fact-oriented Logical Weakness Model therefore a special conceptual schema was developed in support of the adaptation phase of the two-phase fact-oriented merging of the original facts into the common fact base. We also defined the facts where the original vulnerability findings can be merged with the basic facts about the system under assessment, as defined by the standard Knowledge Discovery Metamodel (KDM).

Finding

Definition: report of a flaw in the system under assessment.

Note: Some findings describe exploitable security vulnerabilities, however many findings reported by automated tools are related to various bad code practices.

Note: Finding is defined by a reference to the kind of flaw. This association is provided through a weakness description cited by the original report, as well as a CWE identifier from the Common WEakness Enumeration (CWE) catalog and an SFP and Cluster from the Software Fault Patterns (SFP) catalog).

Note: Finding is defined by a reference to the location in the system under assessment.

Code location

Note: this provides a unique reference schema for findings. The corresponding concept in KDM is SourceRef

File

Code location *references* file

Code location *has* line number

Description: for locations in source files; this is optional

Possibility: code location may have line number

Code location *has* position

Description: for locations in source files; this is optional

Possibility: code location may have position

Code location *has* offset

Description: for locations in binary files

Possibility: code location may have offset

Finding *has* code location

Name

File *has* name

Checksum

File *has* checksum

Note: the ability to compute the checksum of a file that is the source for the particular weakness report depends on the access to this file. In general, the application of the generator is done at a separate phase; therefore the adaptor may not be able to compute this information. However, availability of the checksum will facilitate management of multiple TOIF segment and reduce errors caused by merging unrelated TOIF segment.

Note: the Inventory Model of the KDM Model includes the checksum of each file in the system under assessment.

Version

File *has* version

Note: the ability to compute the version of a file that is the source for the particular weakness report depends on the access to this file. In general, the application of the generator tool is done at a separate phase; therefore the adaptor may not be able to compute this information. However, availability of the version will facilitate management of multiple TOIF segment and reduce errors caused by merging unrelated TOIF segment.

Note: the Inventory Model of the KDM Model includes the version of each file in the system under assessment.

Directory

File *is contained in* Directory

Directory *is contained in* Directory

Directory *has* name

CWE identifier

Definition: Identifier of a Common Weakness Enumeration element from the DHS/MITRE Common Weakness Enumeration catalog

CWE identifier *has* name

Finding *has* CWE identifier

Possibility: each finding may have many CWE identifier

Note: CWE identifier will be added during the adaptation phase. In the situation when it is not possible to map report of a static analysis tool to CWE, the CWE identifier will be omitted.

Note: In the situation when there is an ambiguity in a mapping of a particular finding (type) of a static analysis tool to CWE, multiple CWE identifier will be associated with the corresponding finding.

Note: the TOIF Analyzer may split finding with multiple CWE identifier into several finding with a single CWE identifier

SFP identifier

Definition: Identifier of a Software Fault Pattern from the DoD Software Fault Patterns catalog

Note: Software Fault Pattern (SFP) catalog is useful for mapping findings of proprietary tools into CWE as part of the TOIF adaptation phase. Since the SFP catalog has strict hierarchical organization (Cluster-> Secondary cluster/SFP -> CWE), it is easier to use this organization to systematically map a finding into CWE. On many occasions, the mapping to CWE is ambiguous. The mapping to SFP or even to an SFP Cluster is more meaningful

SFP identifier *has* name

Finding *has* SFP identifier

Necessity: each finding may have at most one SFP identifier

Note: SFP identifier will be added during the adaptation phase. In the situation when it is not possible to map report of a static analysis tool to SFP, the SFP identifier will be omitted.

Cluster identifier

Definition: Identifier of a Cluster from the DoD Software Fault Patterns catalogue

Cluster identifier *has* name

Finding *has* Cluster identifier

Necessity: each finding may have at most one Cluster identifier

Note: Cluster identifier will be added during the adaptation phase. In the situation when it is not possible to map report of a static analysis tool to SFP, the SFP identifier will be omitted.

Description

General concept: Text

Weakness Description

General concept: Description

Finding *is described by* Weakness Description

Statement

Note: this corresponds to KDM ActionElement, however the correspondance is not unique

Statement *has* code location

Statement *is involved in* Finding

Synonym: Finding *is associated with* statement

Possibility: each Finding may *be associated with* many statement

Statement *is part of sink of* Finding

Note: In is a stronger form of the fact type Statement *is involved in* Finding where the role of Statement with respect to the Logical Weakness Model is known (i.e. sink)

Statement *is part of source of* Finding

Note: In is a stronger form of the fact type Statement *is involved in* Finding where the role of Statement with respect to the Logical Weakness Model is known (i.e. source)

Statement *is preceded by* Statement

Data element

Note: this corresponds to KDM DataElement, however the correspondance is not unique

Data element *is involved in* Finding

Data element *has* name

Data element *is involved in* Statement

3.2 FACT-ORIENTED ORGANIZATION OF TOIF XML

This section elaborates the conceptual model and describes the fact-oriented organization of the TOIF XML.

First, we will use a Segment as the root element for the TOIF XML file. Thus a TOIF segment will be the main unit of information exchange within the TOIF framework. The TOIF Segment will *own* the corresponding entities and facts. The concept of element ownership is important from the design perspective of the XML. Ownership corresponds to the nested XML tags. We assume that every TOIF entity and every TOIF fact will be defined by a unique pair of XML tags. Facts and entities in a TOIF Segment will be flat, i.e a TOIF Segment is an order list of TOIF entities and facts. We will also put forward the following logical constraints:

- C1: TOIF Segment shall own all TOIF entities that are subjects of the TOIF facts owned by that Segment
- C2: TOIF entity that is the subject of a TOIF fact shall precede that fact in the TOIF Segment.

This gives us an additional conceptual schema:

Fact

Description: a TOIF fact

Entity

Description: a TOIF entity

General concept: Thing

General concept: Fact

Note: an Entity is introduced by the so-called existential fact

Entity *is subject of* Fact

Synonym: Fact *references* Entity

TOIF Segment

Definition: Collection of facts related to the assessment of a system.

Note: TOIF Segment may include basic facts as well as the corresponding housekeeping facts.

TOIF Segment *has* name

TOIF Segment *has* Description

TOIF Segment *owns* Fact

Fact1 *precedes* Fact2

Necessity: Entity *precedes* Fact that *references* the Entity

TOIF Segment *owns* Entity

Necessity: TOIF Segment *owns* each Entity that is *referenced by* Fact that is *owned by* the TOIF Segment

3.3 "HOUSEKEEPING" CONSIDERATIONS FOR TOIF XML

In this section we describe several "housekeeping" facts that facilitate management of multiple TOIF XML files during the entire life cycle of the operation of the TOIF framework. Most of these considerations are taken from the forthcoming OMG Structured Assurance Case Metamodel (SACM). specification

We need to associate some "audit" information with the TOIF XMI segment. The key objectives are

- to facilitate management of multiple TOIF Segment generated during the course of the TOIF project
- to reduce the possibility of errors caused by merging unrelated TOIF Segment
- to reduce the possibility of errors caused by merging TOIF Segment with the unrelated KDM model

The audit information shall include the following:

- project identifier (unique project name)
- name of the generator tool
- vendor name of the generator tool
- generator tool identifier (unique version of the generator tool)
- adaptor identifier (unique name and version of the adaptor tool)
- person name responsible to the TOIF Segment
- organization responsible for the TOIF Segment
- date when the TOIF Segment was produced

This gives us an additional conceptual schema with additional Entity and Fact:

Project

Description: a TOIF project

Project *has* name

Project *has* Description

Note: a separate TOIF Segment may be used to own all "housekeeping" elements and their descriptions

Generator

Description: a tool or a service capable of assessing a system based on its code artifacts and providing Finding

Generator *has* Description

version

Generator tool *has* name

Generator tool *has* version

Vendor

Description: supplier of a Generator used in project

General concept: Organization

Adaptor

Description: a tool or a service capable of transforming Finding from original logical and physical format into TOIF format

Adaptor *has* name

Adaptor *has* version

Generator *is supplied by* Vendor

Synonym: Vendor *supplies* Generator

Adaptor *is supplied by* Vendor

Synonym: Vendor *supplies* Adaptor

Adaptor *supports* Generator

Synonym: Generator *requires* Adaptor

Person

Person *has* name

Person *has* email address

Person *has* phone number

Email address

Phone number

Address

Role

Role *has* name

Role *has* Description

Person *is employed by* Organization *as* Role

Note: we assume that the dynamics of this relationship is not relevant to the TOIF. So this relationship means that the Person was employed by Organization for the duration of the project.

Organization

Description: an Organization involved in project

Organization *has* Description

Organization *has* name

Organization *has* address

Organization *has* email address

Organization *has* phone number

Organization *is part of* Organization *as* Role

Date

Person *is involved in* Project *as* Role

Organization *is involved in* Project *as* Role

TOIF Segment *is related to* Project

TOIF Segment *is generated by* Person

TOIF Segment *is supervised by* Person

Synonym: Person *is responsible for* TOIF Segment

TOIF Segment *is produced by* Organization

TOIF Segment *is owned by* Organization

TOIF Segment *is created at* Date

TOIF Segment *is generated by* Generator

Synonym: Generator *generates* TOIF Segment

TOIF Segment *is processed by* Adaptor

Synonym: Adaptor *processes* TOIF Segment

Necessity: Adaptor that *processes* TOIF Segment *supports* Generator that *generates* the TOIF Segment

4 FROM CONCEPTUAL SCHEMA TO A UML MODEL

4.1 TRANSFORMATION FOR CONCEPTUAL SCHEMA TO A UML MODEL

This section presents the UML model for TOIF XML which is developed as an intermediate step towards the TOIF XML schema.

Conceptual schemas for the TOIF XML can be systematically transformed into a UML model without attributes. Each concept in the conceptual model is transformed into a UML class. Each fact type in the conceptual model is transformed into an association. We generated the UML model manually. The model consists of a single UML package and includes 17 class diagrams to represent the following:

- TOIF logical concepts (section 3.1)
- TOIF fact-oriented concepts (section 3.2)
- TOIF housekeeping concepts (section 3.3)

We applied the following systematic transformation of the TOIF XMI conceptual schema into UML class diagrams:

- Each noun concept is represented by a UML class
    - The name of the UML class representing a noun concept is the same as the name of the noun concept, except for the deleted spaces.
- Each fact type is represented by a separate UML class
    - The name of the UML class representing a fact type is the full designation of the fact type, for example, for the fact type "Person *is employed by* organization *in* role" the name of the UML class is `PersonIsEmployedByOrganizationInRole`.
- The UML class that represents a fact type has one or more associations to the classes that represent noun concepts that are referenced by the fact type.
    - The role names of the associations are same as the names of the noun concepts
    - The multiplicity is always 1
    - The association is navigable only in the direction to the noun concept
- Additionally we reviewed the fact types that involve the verb "has" and represented some of them as attributes
    - The corresponding noun concepts have stereotype <<attribute>>
    - Attributes do not involve a UML association class, but have a direct aggregate association for the UML class that represents the owner of the attribute.

    - The multiplicity of the attribute class corresponds to the necessity of the corresponding fact type in the conceptual model

- Each UML class that represents a concept of the conceptual model (noun concept or fact type) has an additional identifier

The UML model involves the following:

- 8 classes that represent logical entities (noun concepts)
- 3 classes that represent attributes of the logical entities
- 13 association classes that represent logical facts
- 4 classes that represent physical concepts
- 7 classes that represent housekeeping entities
- 6 classes that represent attributes of the housekeeping concepts
- 1 association classes that represent housekeeping facts

We made a decision to structure our UML model to have an explicit set of classes corresponding to the conceptual schema. This allows the best utilization of the "out-of-the-box" type checking capabilities provided by UML tools and Eclipse EMF tools. An alternative decision will be to use a small Core that is instantiated with names of entities, fact types and role bindings. This is the approach taken in the OMG Structured Assurance Case Metamodel (SACM). In this alternative approach the resulting TOIF XML data format is almost the same as our current format for the explicit model However the XML schema resulting from the alternative approach will lack guidance, and will need to be accompanied with the conceptual schema.

The rest of this section has the following organization. Section 4.2 presents 8 UML class diagrams that describe the logical elements of the TOIF XML, the logical entities and fact types. Section 4.3 presents a single UML class diagram that describes the physical structure of the TOIF XML. Section 4.4 presents 8 UML class diagrams that describe the housekeeping elements of the TOIF XML.

4.2 THE LOGICAL ELEMENTS OF THE TOIF XML

This section presents 8 UML class diagrams that represent the logical entities and facts of the TOIF XML.

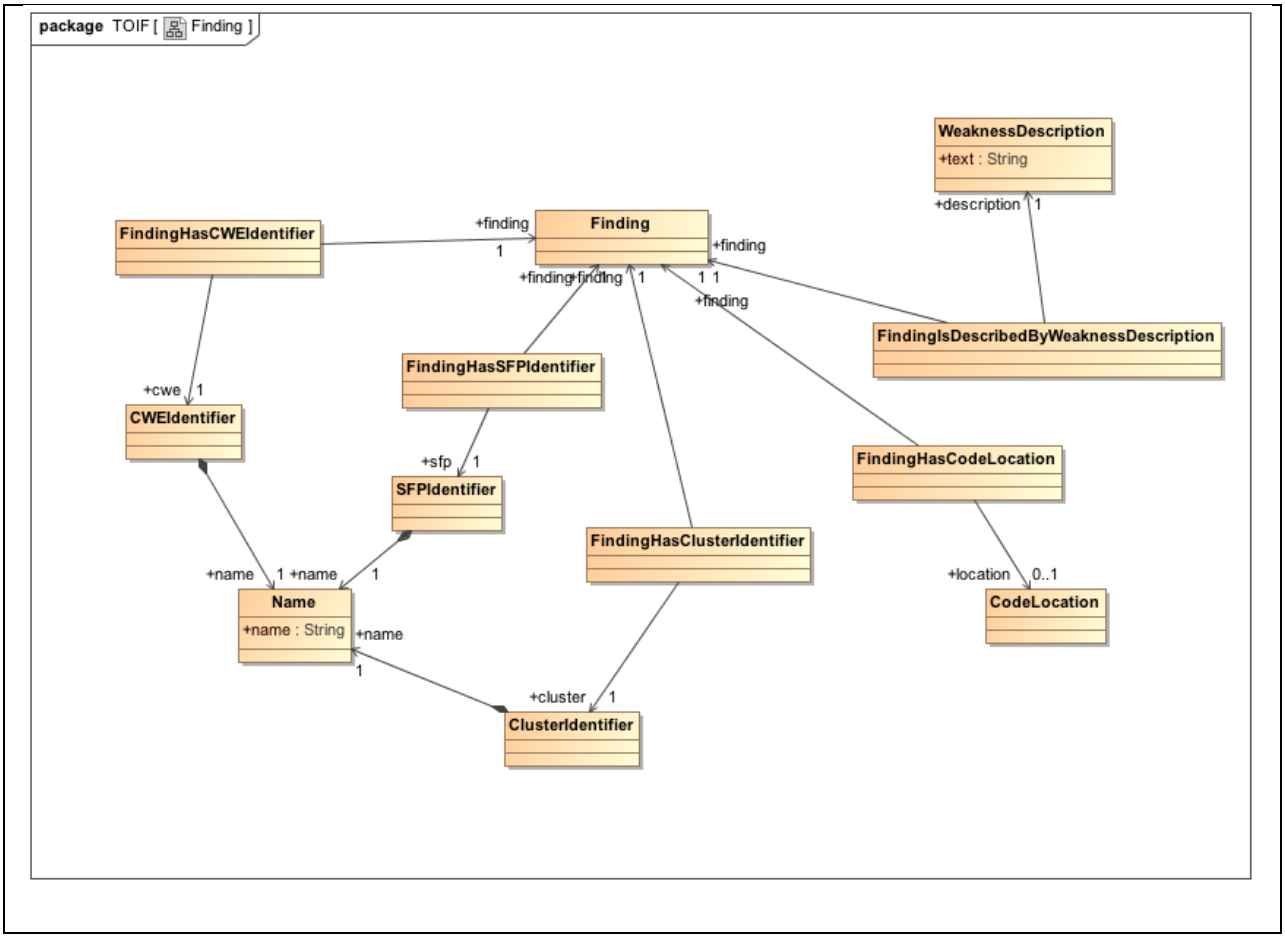

FIGURE 1. UML CLASS DIAGRAM FINDING

Note that the element "WeaknessDescription" does not fully comply to the mapping from the TOIF conceptual model to a UML model. Instead of a logical representation of the attribute "text" using an owned "Attribute" class, it is represented as a direct physical attribute of class "WeaknessDescription". This was introduced in TOIF 1.0 and was preserved in TOIF 1.1 for compatibility. In terms of the mapping, "WeaknessDescription" class is an "Entity" which usually does not have any logical attributes. Note, that this decision leads to two slightly different TOIF XML representations. The usual logical attribute (for example class "Project") is represented as follows:

```
<fact xsi:type="toif:Project" id="pr1">
    <name name="Tool Integration Framework (TOIF)"/>
    <description text="DHS SBIR project"/>
  </fact>
```

On the other hand, the "WeaknessDescription" element is represented as follows:

```
<fact
   text="variableScope: The scope of the variable val can be
reduced"
   xsi:type="toif:WeaknessDescription"
id="45"/>
```

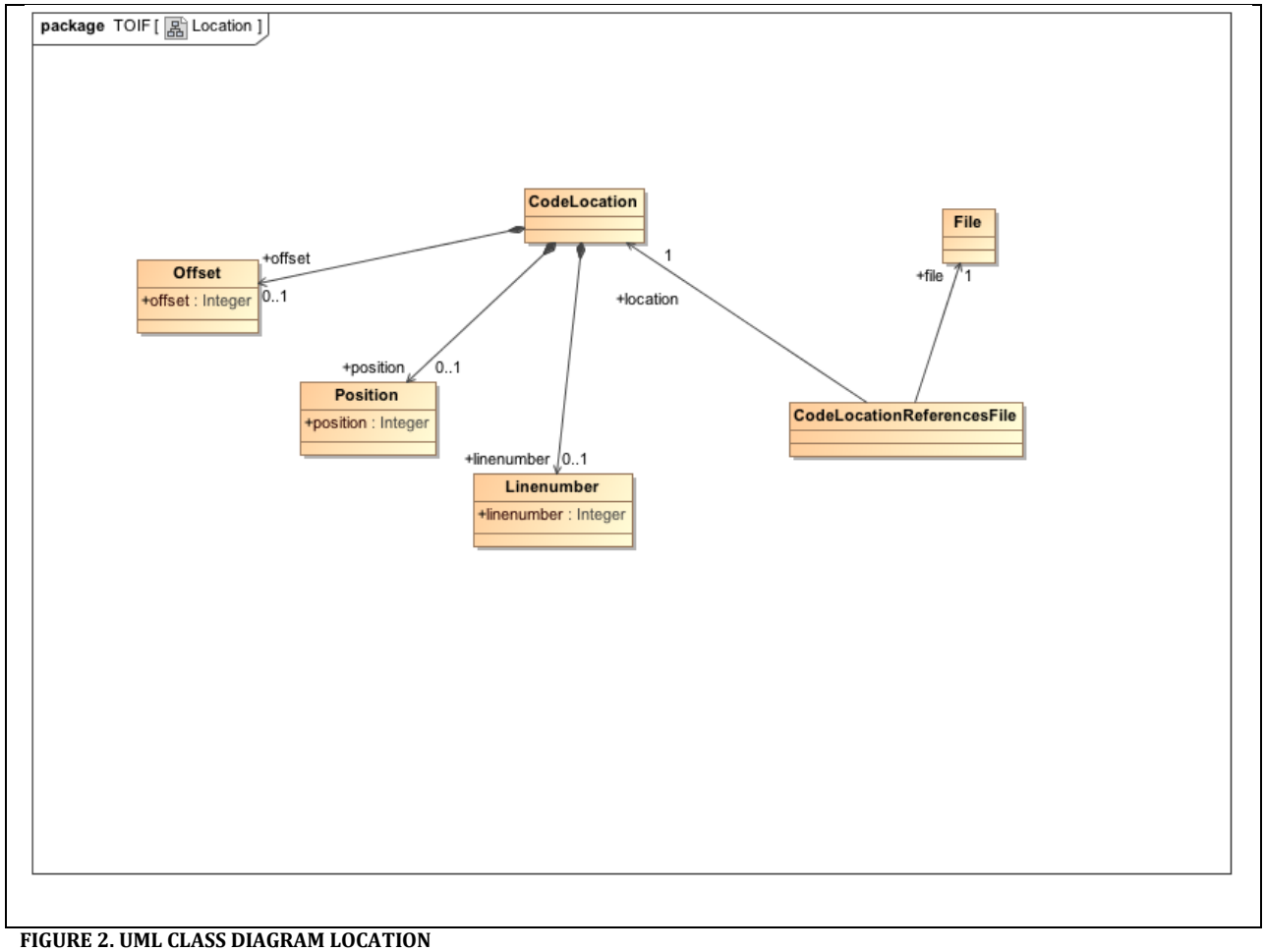

FIGURE 2. UML CLASS DIAGRAM LOCATION

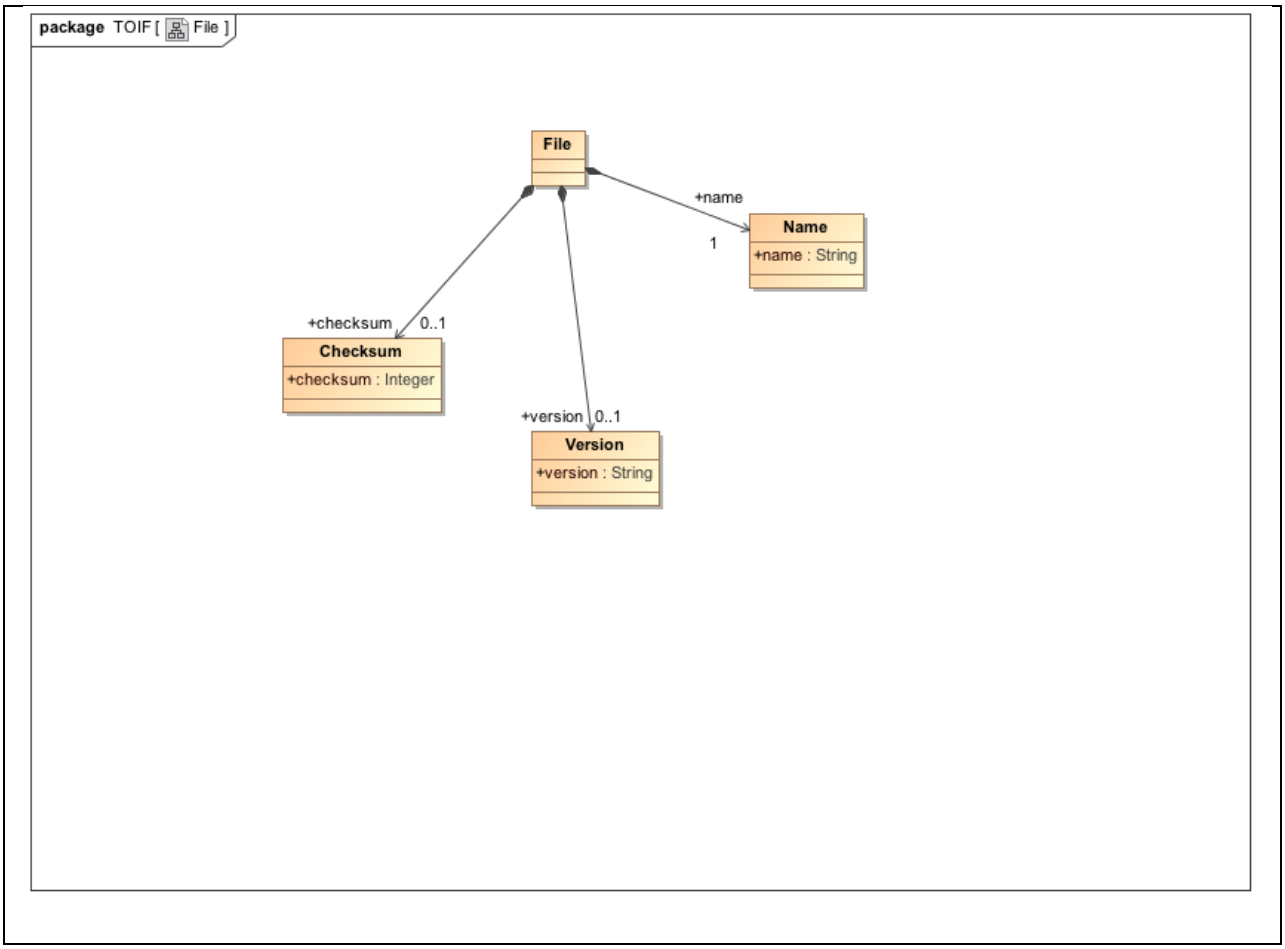

FIGURE 3. UML CLASS DIAGRAM FILE

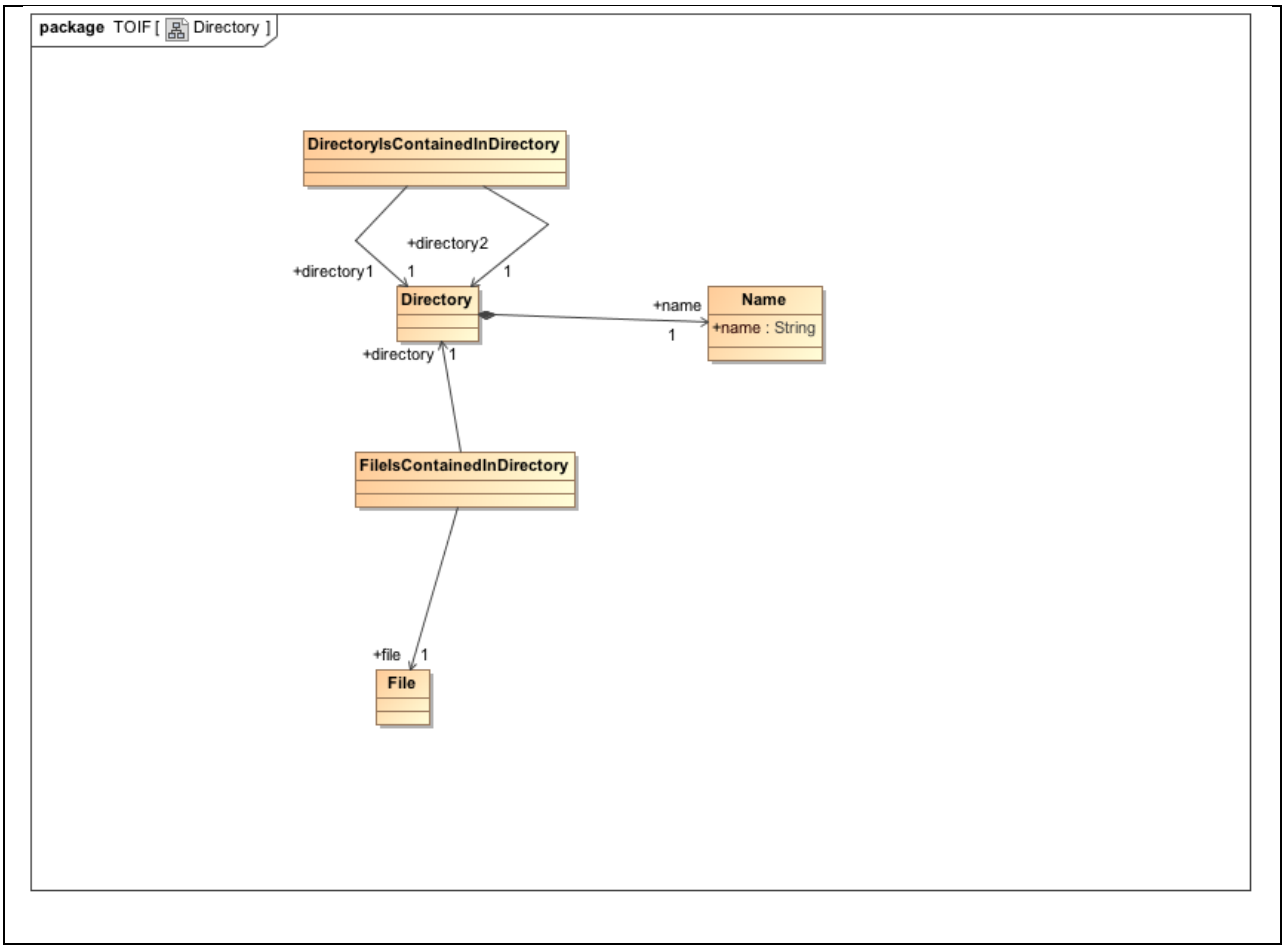

FIGURE 4. UML CLASS DIAGRAM DIRECTORY

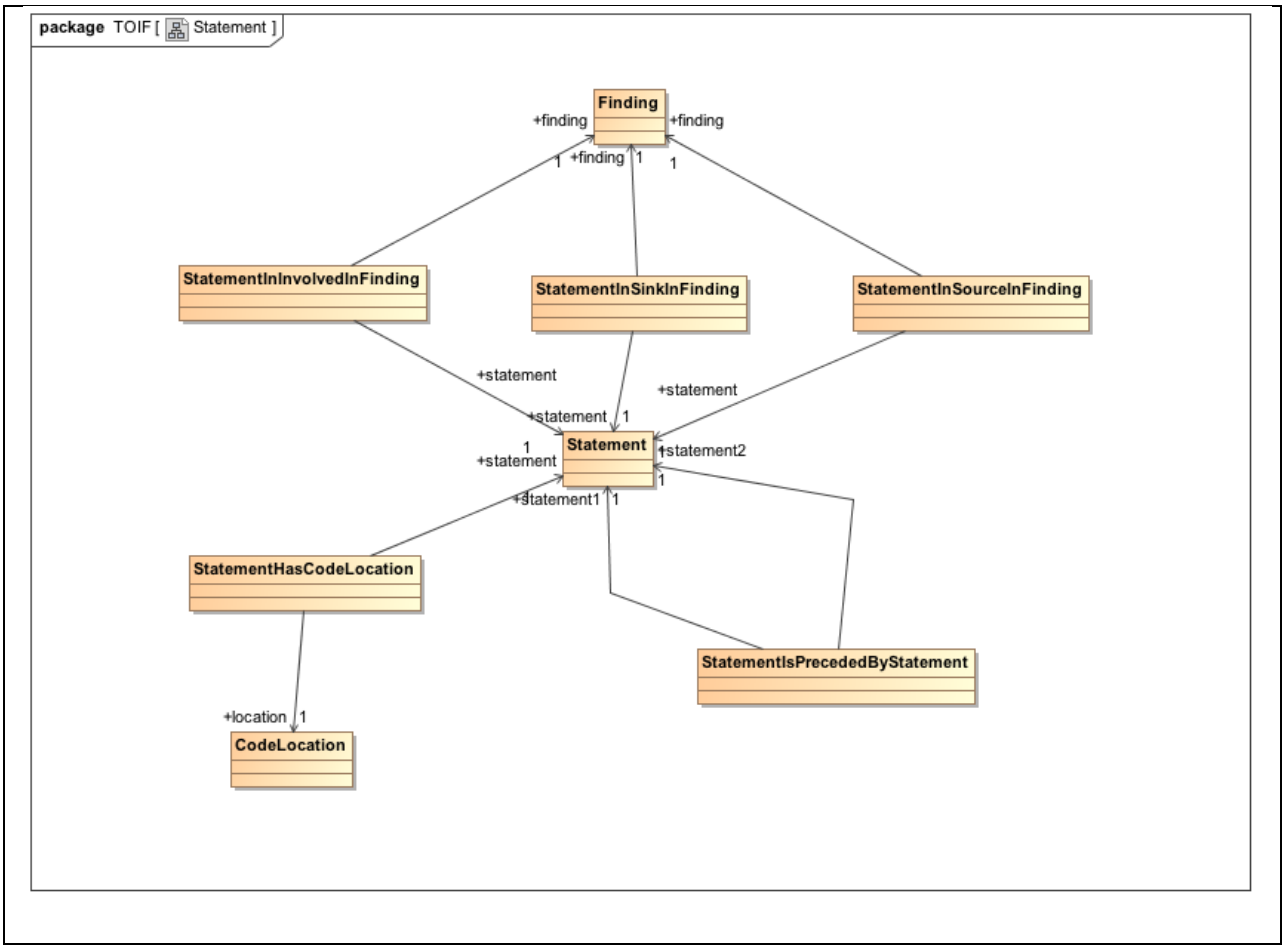

FIGURE 5. UML CLASS DIAGRAM STATEMENT

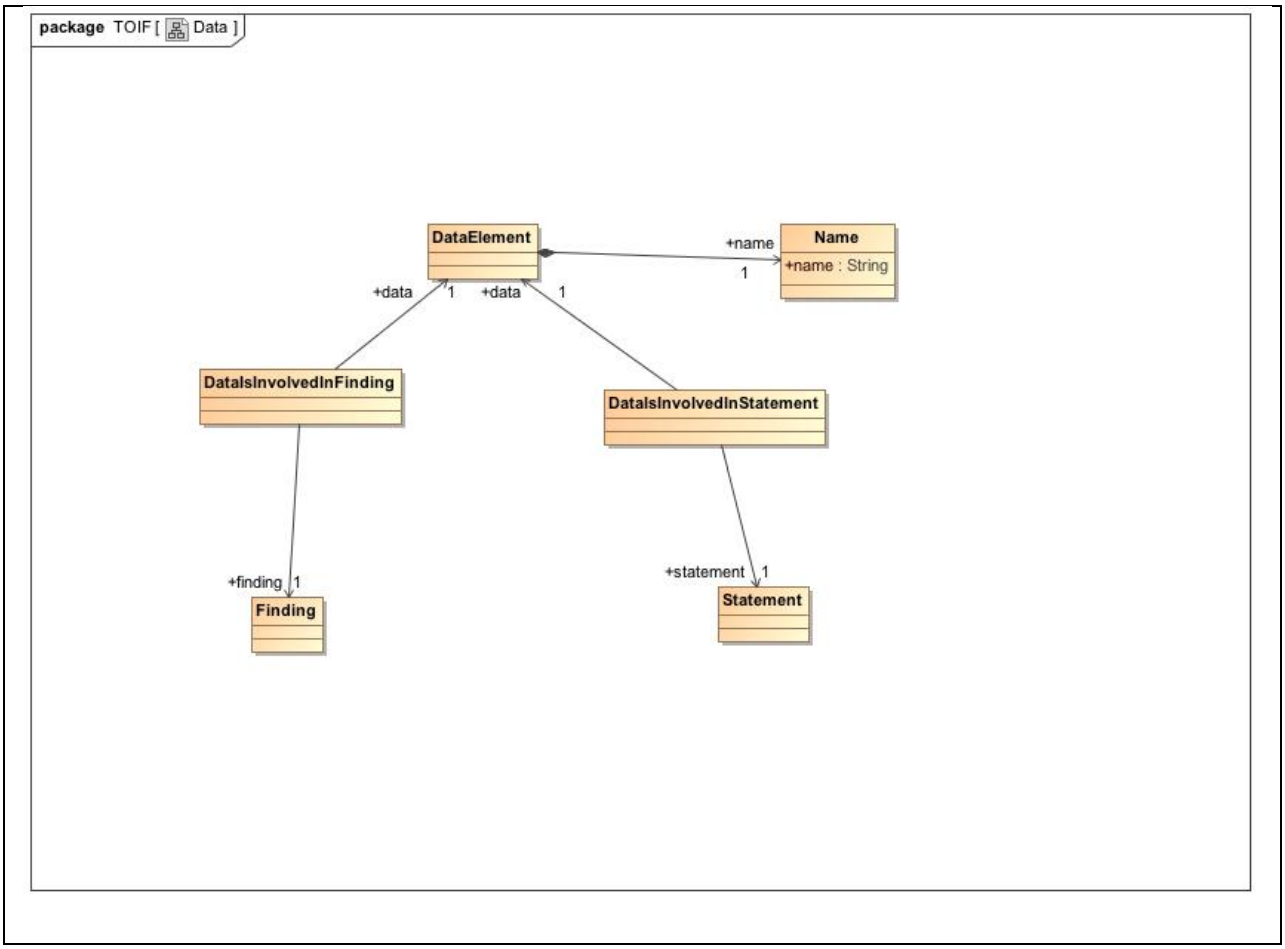

FIGURE 6. UML CLASS DIAGRAM DATA

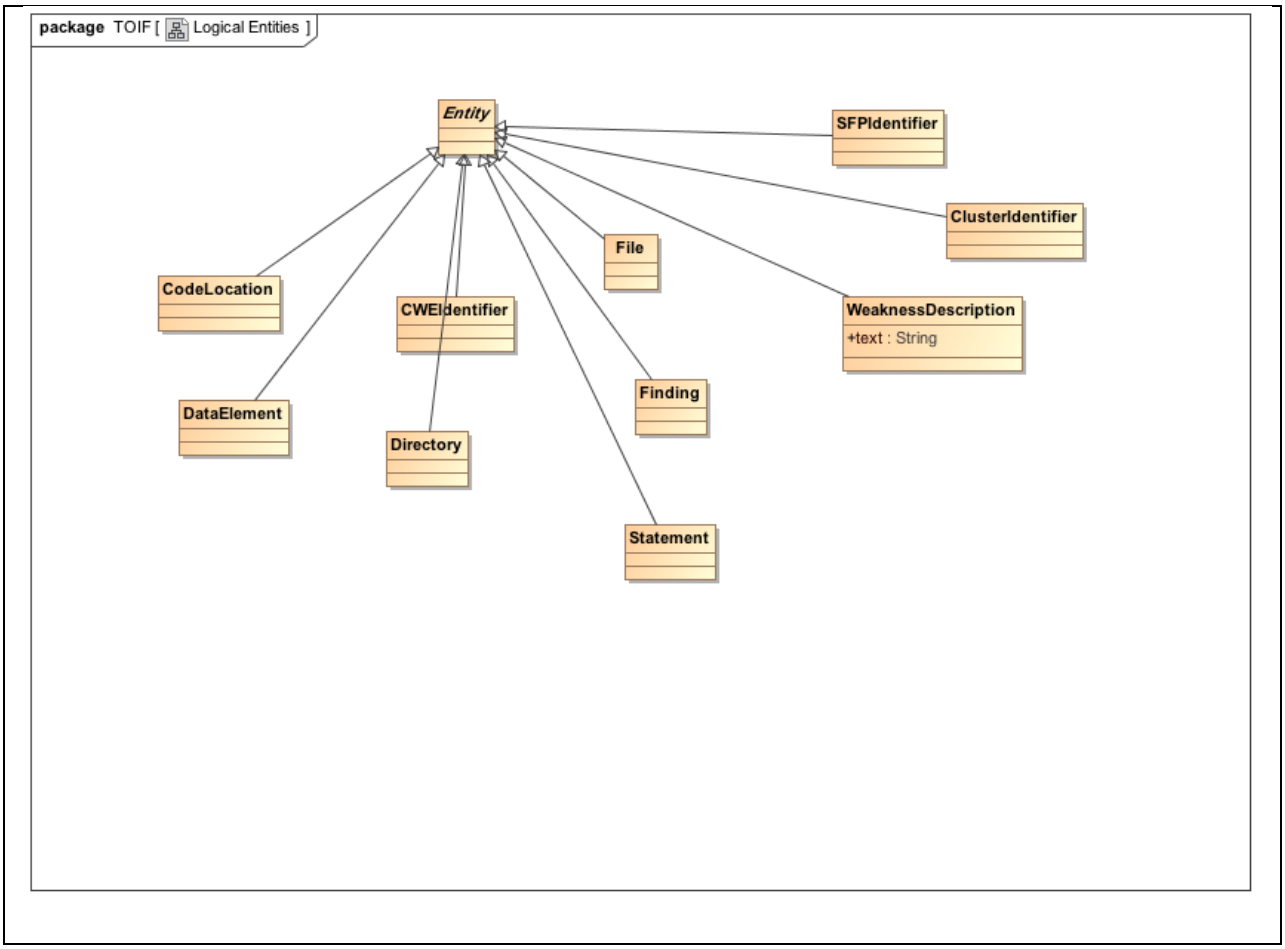

FIGURE 7. UML CLASS DIAGRAM LOGICAL ENTITIES

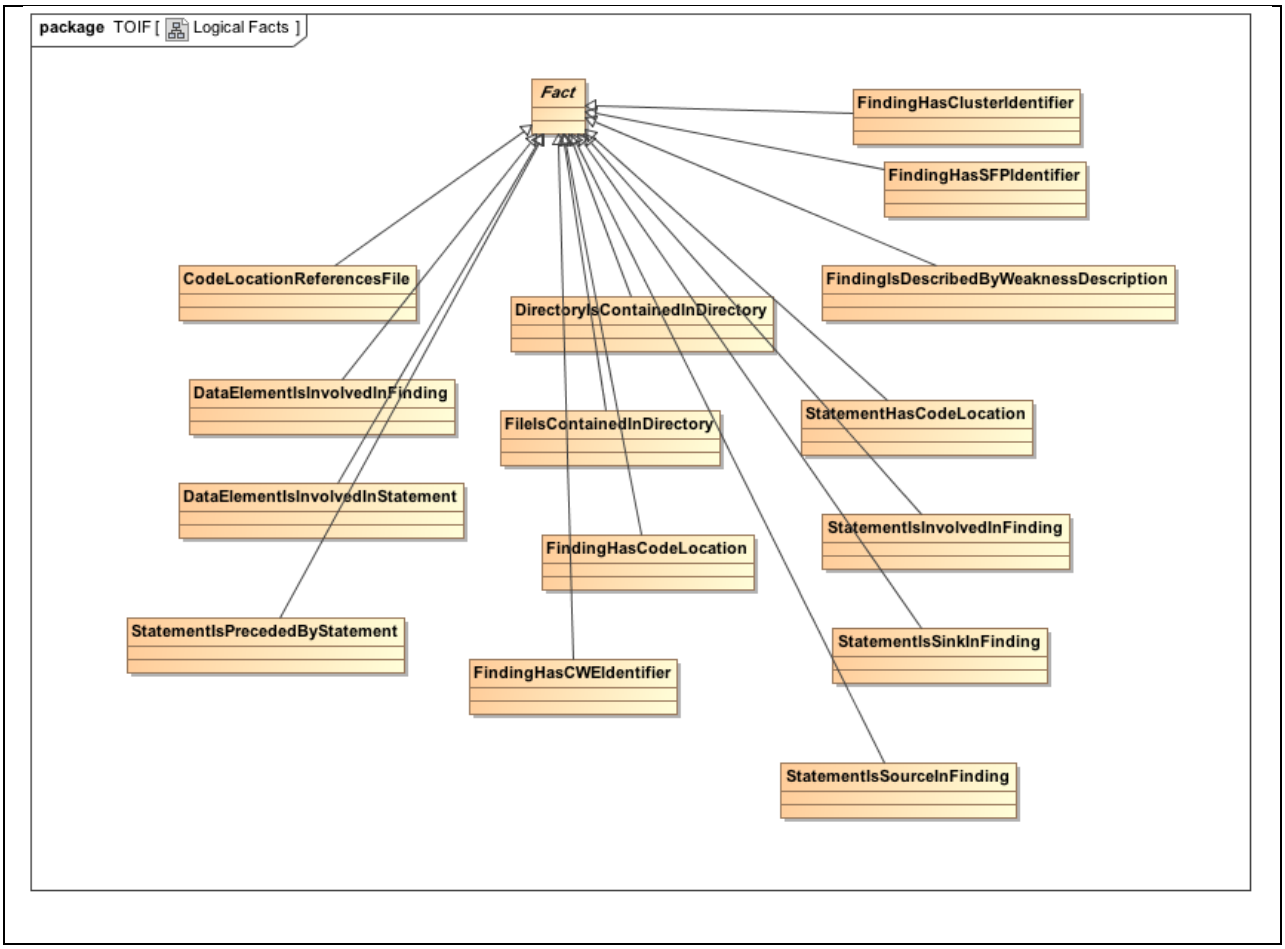

FIGURE 8. UML CLASS DIAGRAM LOGICAL FACTS

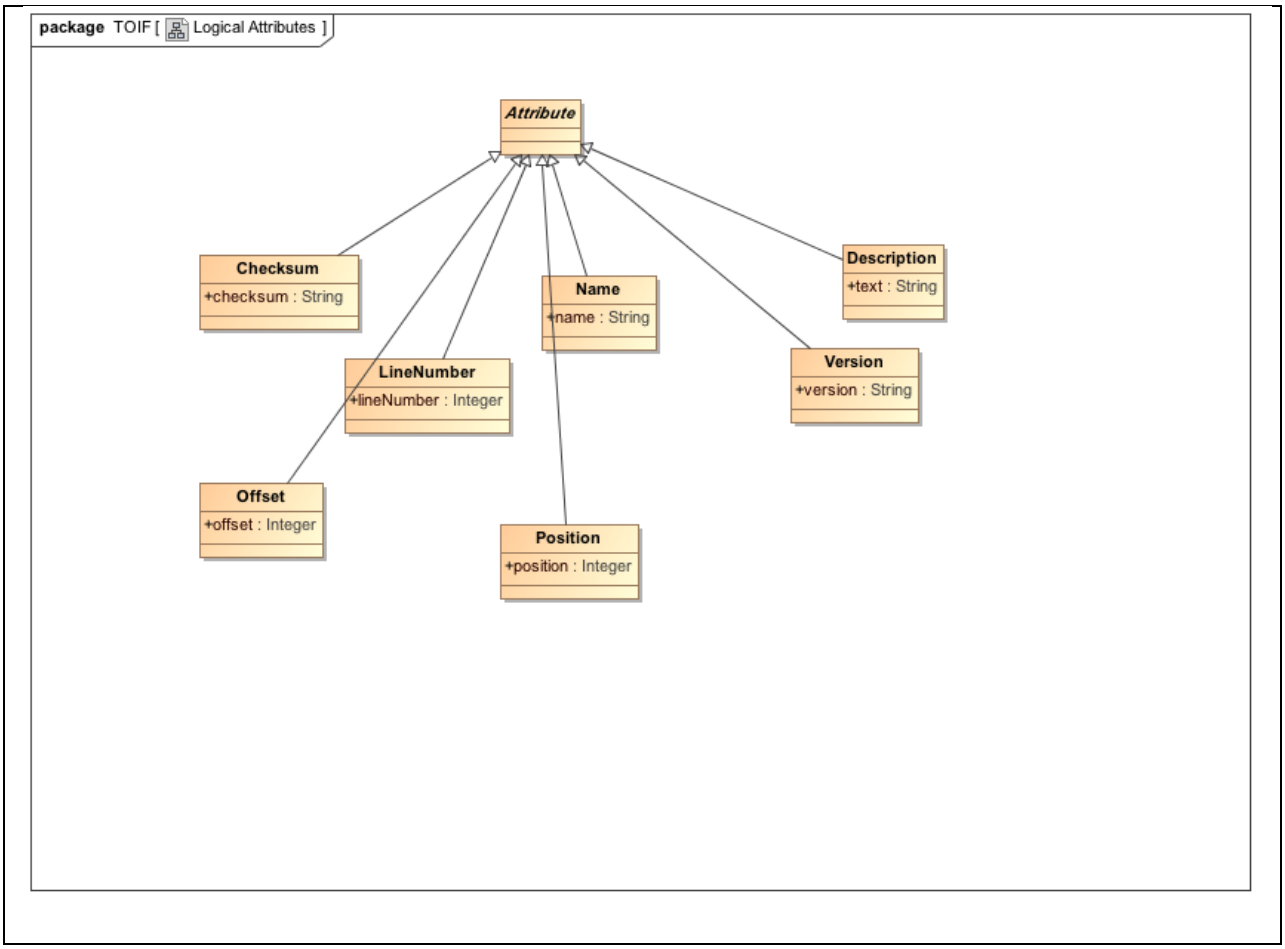

FIGURE 9. UML CLASS DIAGRAM LOGICAL ATTRIBUTES

4.3 THE FACT-ORIENTED STRUCTURE OF THE TOIF XML

This section presents a single UML diagram that describes the physical structure of the TOIF XML.

The following UML diagram represents the physical structure of the TOIF XML. All logical entities are defined as subclasses of class Entity. All logical facts are defined as subclasses of class Fact. Similarly, all housekeeping entities are defined as subclasses of class Entity and all housekeeping facts are defined as subclasses of class Fact. This allows the TOIFSegment class to own an ordered list of both Entities and Facts. Attributes are owned by the entities to which they belong, so they are not owned directly by the TOIFSegment. Class Facts has a UML attribute "id" which is used in the TOIF XML to reference elements. We decided to allow both Entity and Fact to be referenceable.

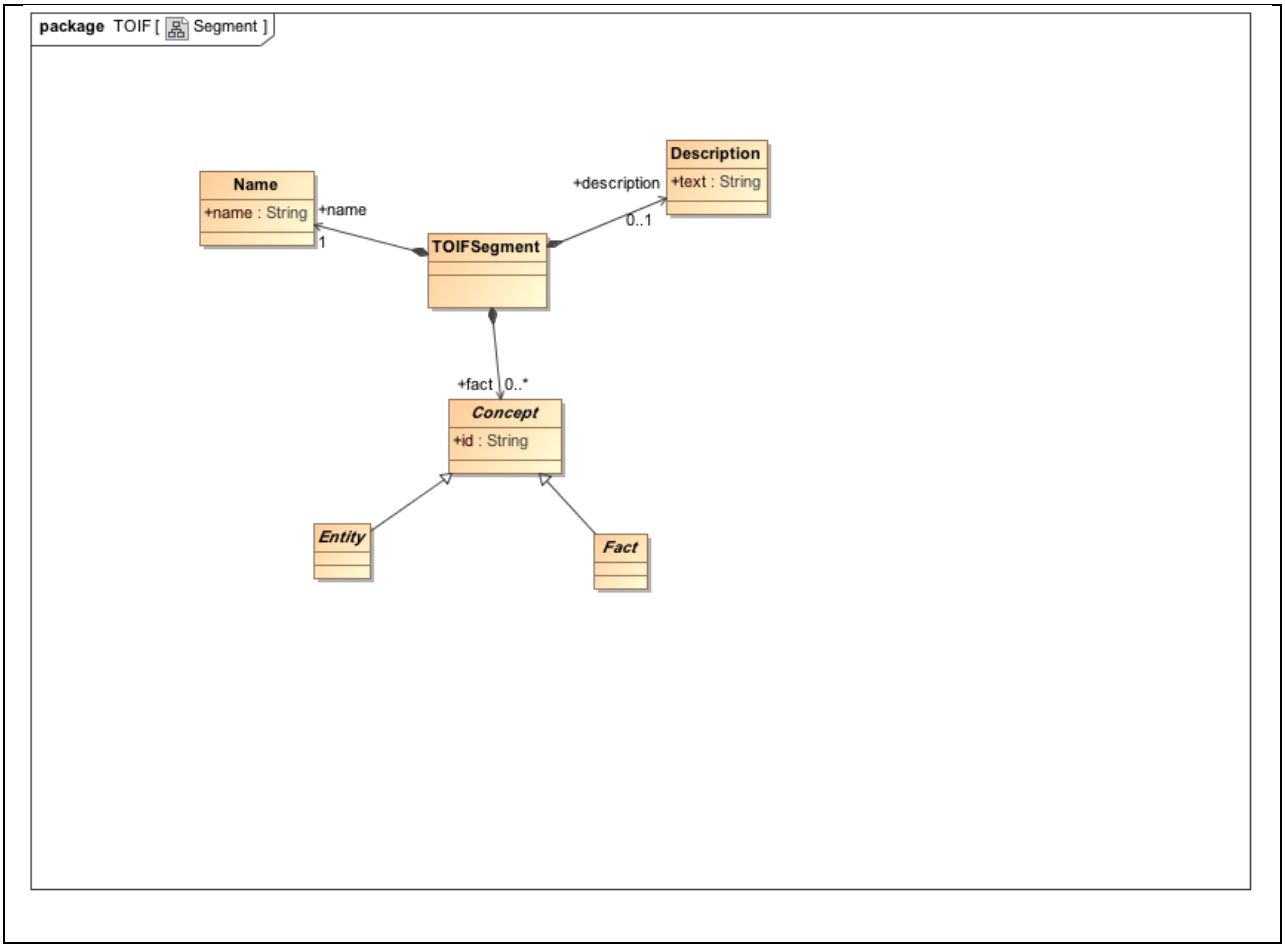

FIGURE 10. UML CLASS DIAGRAM SEGMENT

4.4 THE HOUSEKEEPING ELEMENTS OF THE TOIF XML

This section presents 8 UML class diagrams that describe the housekeeping elements of the TOIF XML.

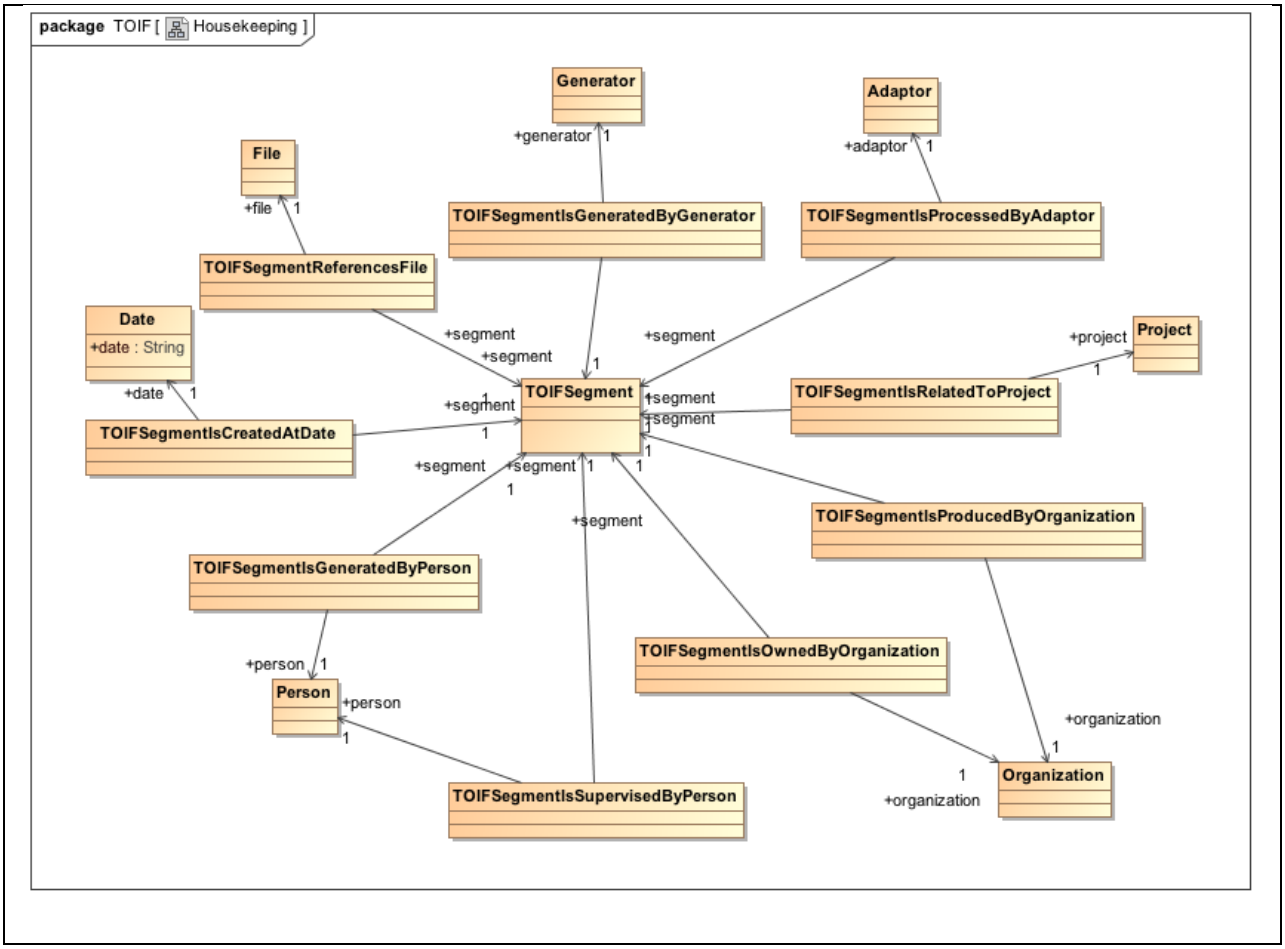

FIGURE 11. UML CLASS DIAGRAM HOUSEKEEPING

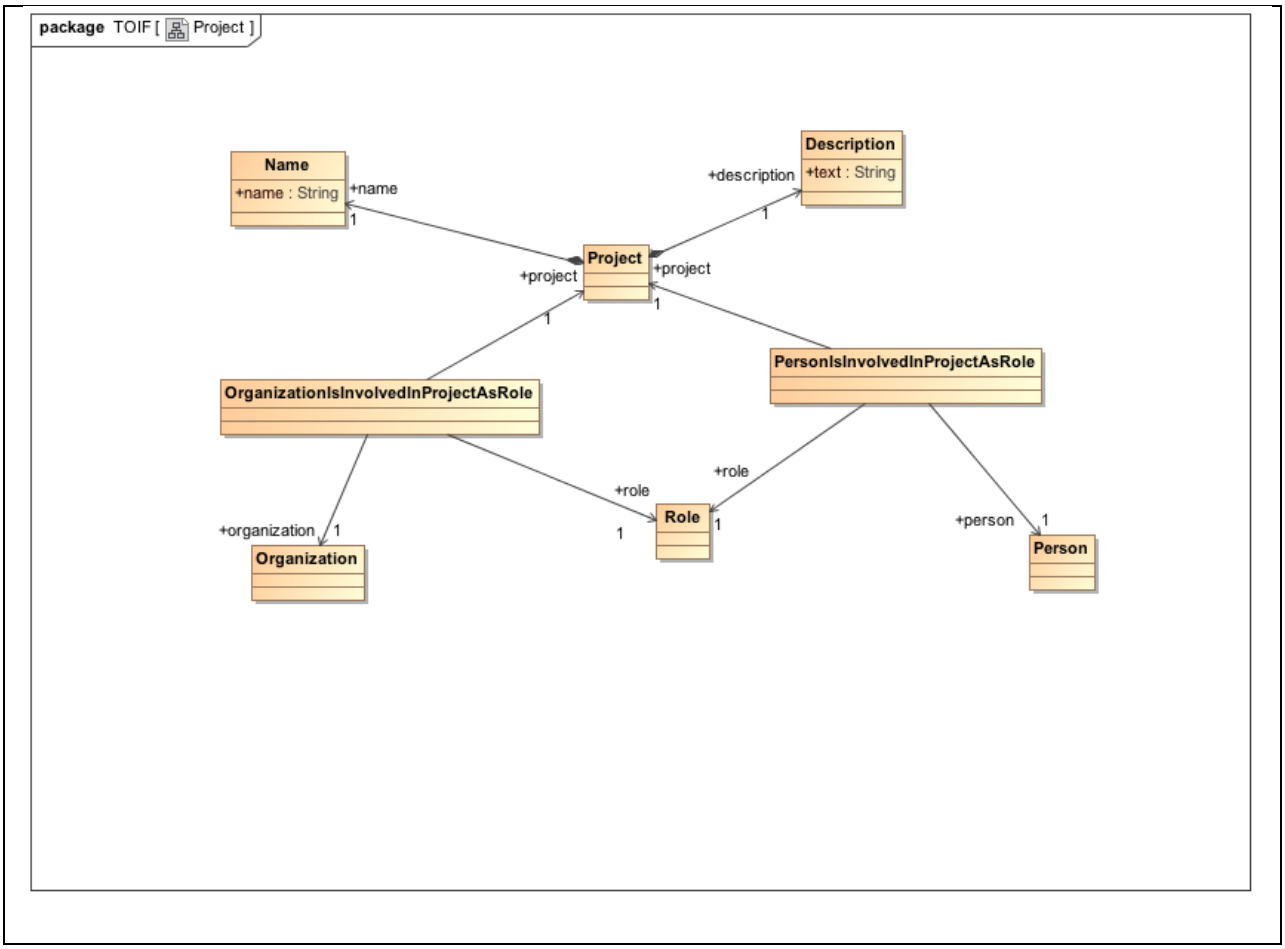

FIGURE 12. UML CLASS DIAGRAM PROJECT

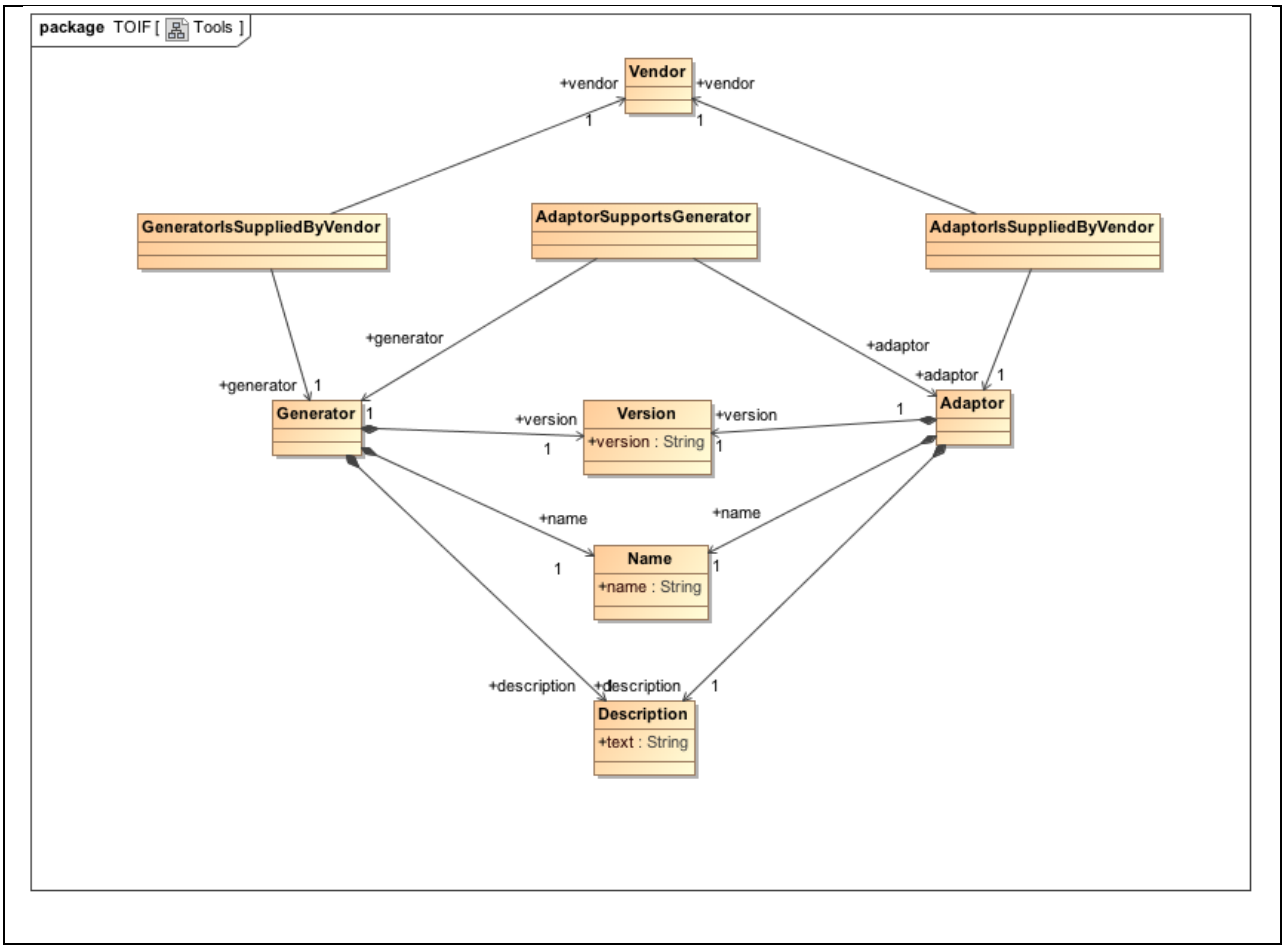

FIGURE 13. UML CLASS DIAGRAM TOOLS

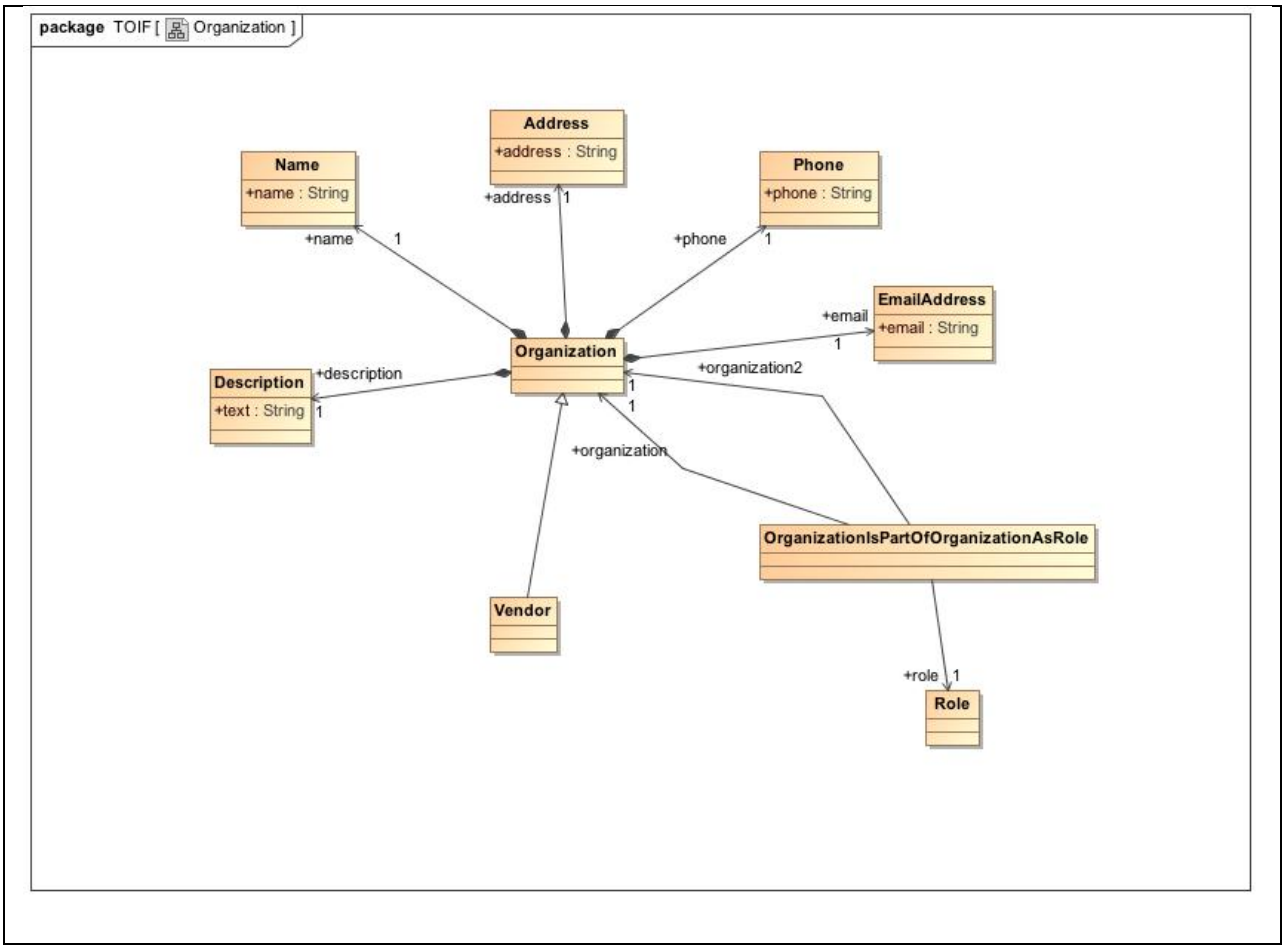

FIGURE 14. UML CLASS DIAGRAM ORGANIZATION

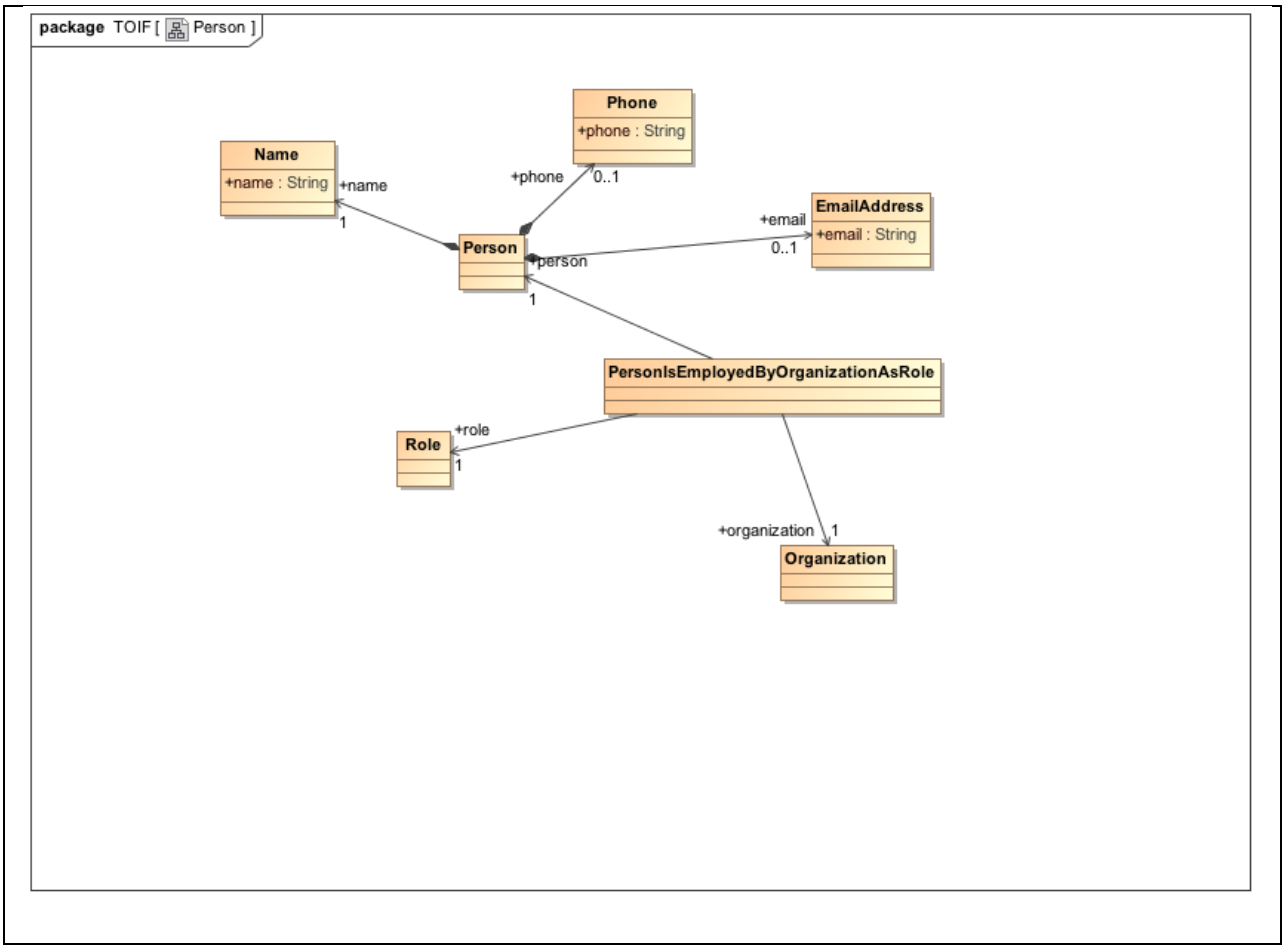

FIGURE 15. UML CLASS DIAGRAM PERSON

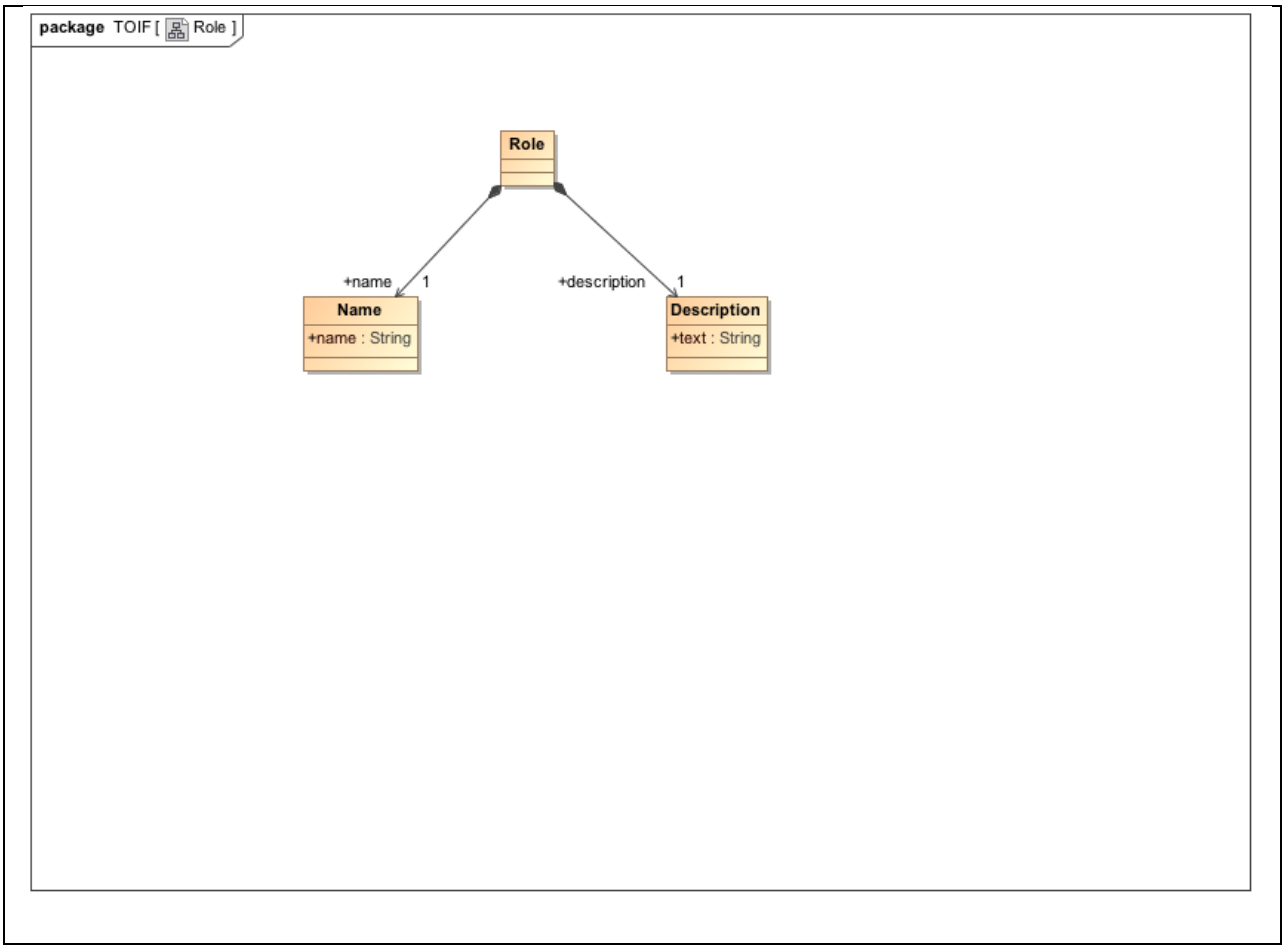

FIGURE 16. UML CLASS DIAGRAM ROLE

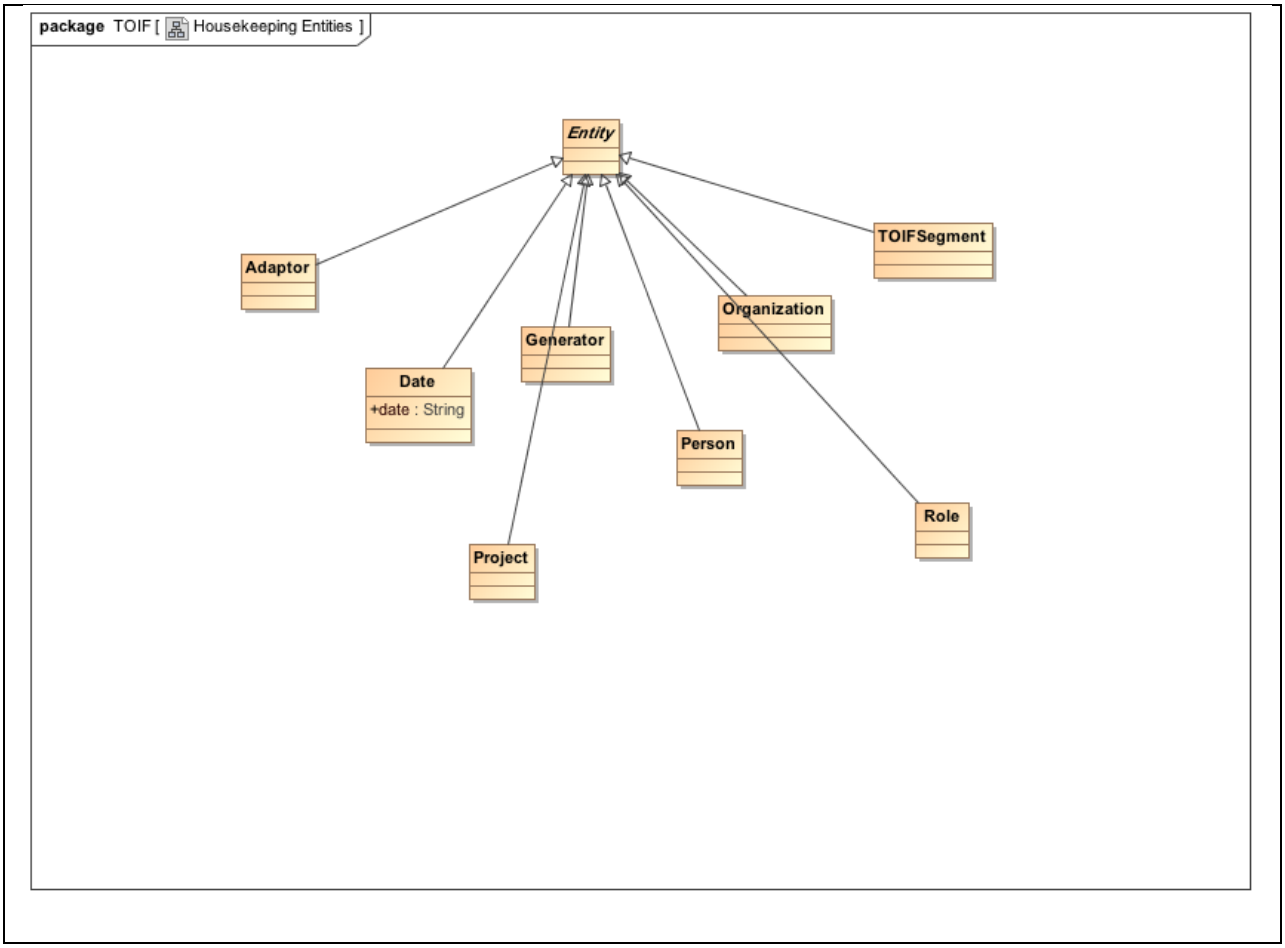

FIGURE 17. UML CLASS DIAGRAM HOUSEKEEPING ENTITIES

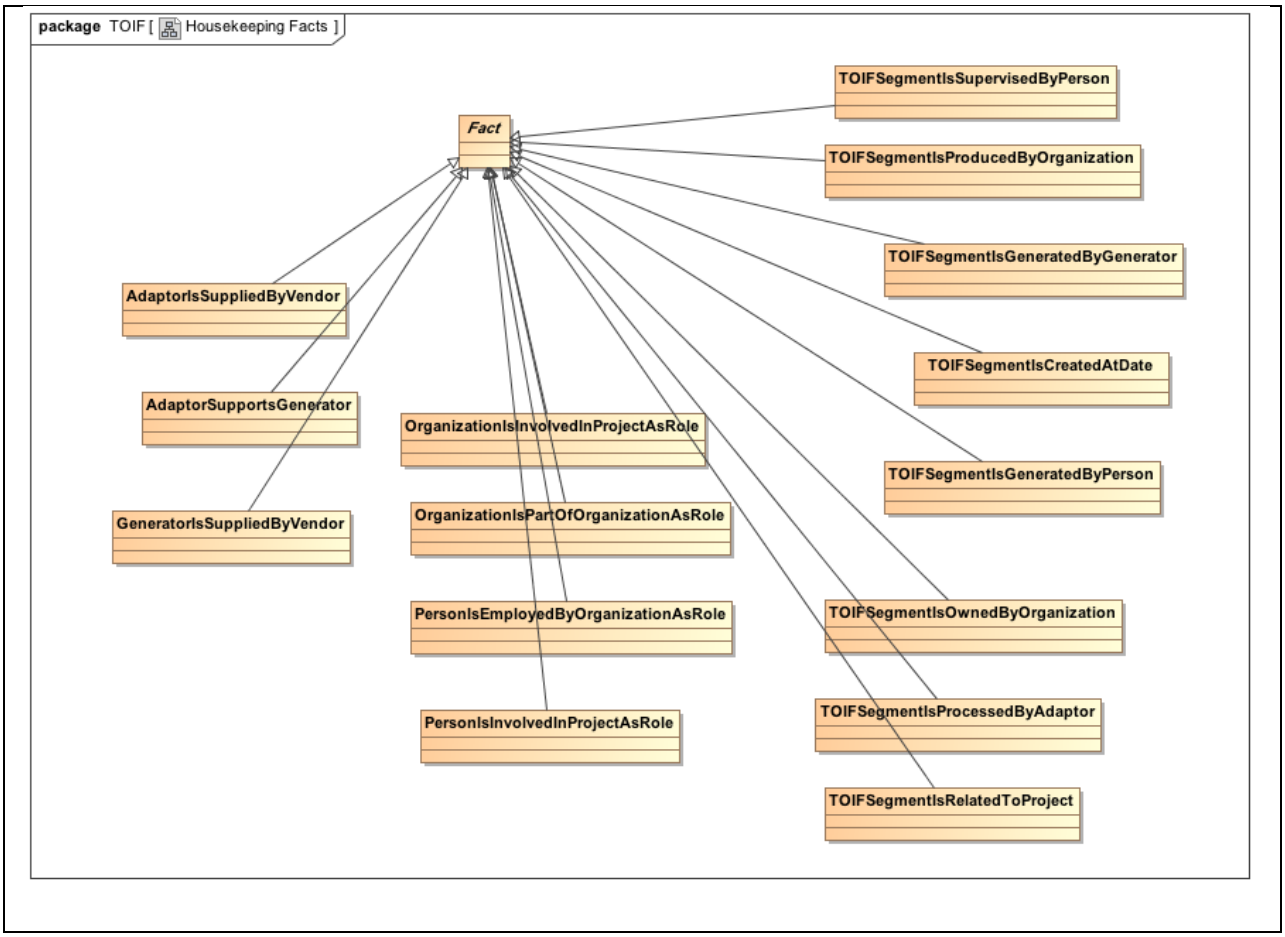

FIGURE 18. UML CLASS DIAGRAM HOUSEKEEPING FACTS

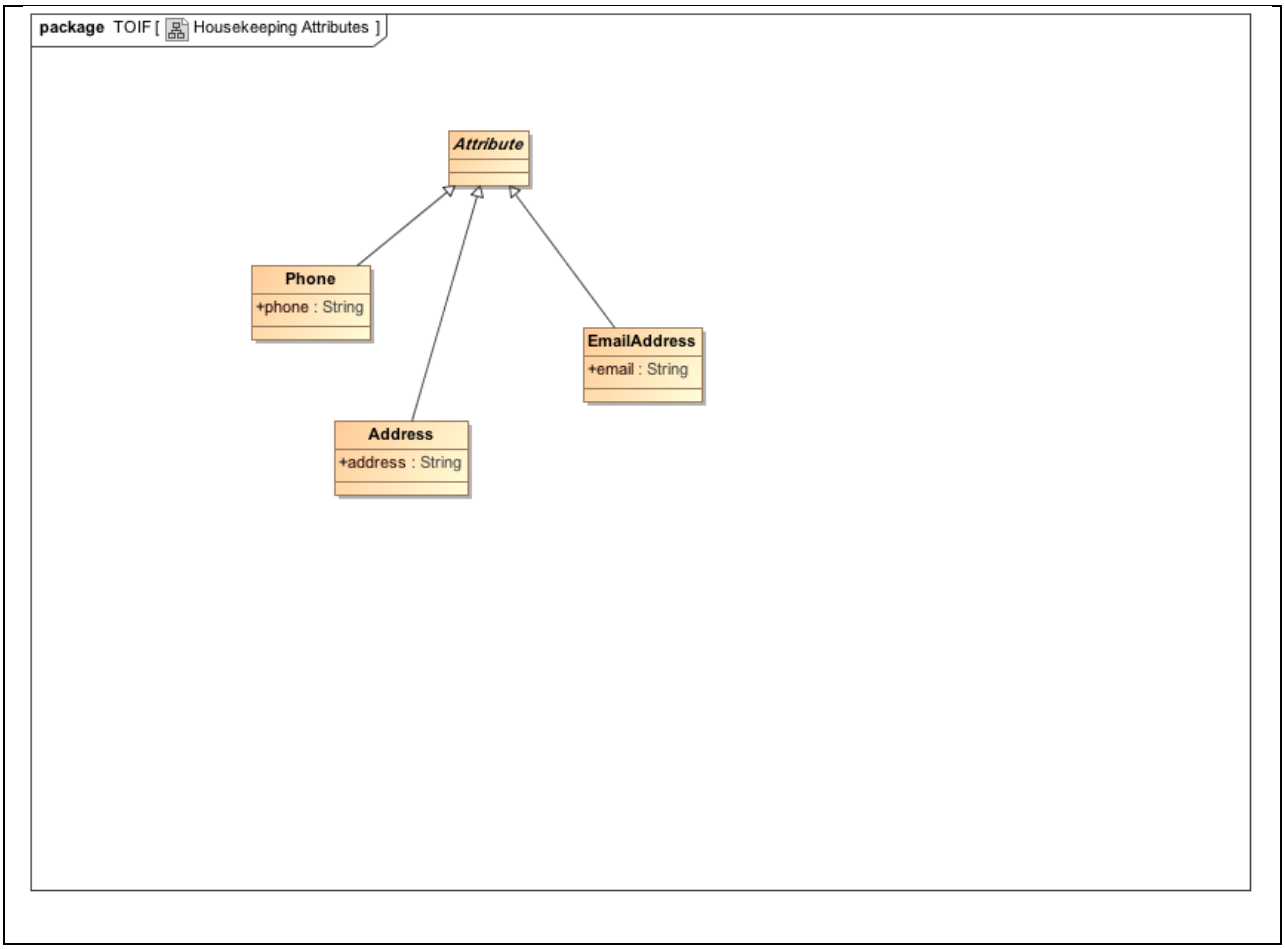

FIGURE 19. UML CLASS DIAGRAM HOUSEKEEPING ATTRIBUTES

5 FROM UML MODEL TO XML/XMI SCHEMA

5.1 TRANSFORMATION FROM A UML MODEL TO XML/XMI SCHEMA

This section describes the process of generating the TOIF XML schema from the UML model presented in section 4.

UML model can be automatically transformed into an XML schema using the following process.

- Step 1. UML model is imported into Rational Rose UML modeling tool.
- Step 2. The UML model in Rose format (.mdl) is imported into Eclipse EMF tool and an Ecore project is created. As the outcome of this step, the model is represented as a .ecore file.
- Step 3. EMF code generator is used to produce the TOIF SDK (Software Development Kit) that can be used to author TOIF XML examples.
- Step 3. EMF XSD generator is used to produce XMI XSD (the TOIF XML Schema).

The intermediate specification toif_xmi.ecore (machine-readable representation of the TOIF XML in Eclipse Ecore) is provided as a separate deliverable.

The mapping from UML to XML is defined by the OMG standard XMI (XML Metadata Interchange) specification.

5.2 THE TOIF XML SCHEMA

This section presents the entire TOIF XML schema produced using the procedure outlined in section 5.1. The TOIF XML schema is also available in a separate machine readable file. The following section illustrates sample TOIF XML data corresponding to this schema.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xsd:schema xmlns:toif="http:///toif.ecore" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="http:///toif.ecore">
  <xsd:import namespace="http://www.omg.org/XMI"
schemaLocation="platform:/plugin/org.eclipse.emf.ecore/model/XMI.xsd"/>
  <xsd:complexType name="Project">
    <xsd:complexContent>
      <xsd:extension base="toif:Entity">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="name" type="toif:Name"/>
          <xsd:element name="description" type="toif:Description"/>
        </xsd:choice>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="Project" type="toif:Project"/>
  <xsd:complexType name="Name">
    <xsd:choice maxOccurs="unbounded" minOccurs="0">
      <xsd:element ref="xmi:Extension"/>
    </xsd:choice>
```

```
  <xsd:attribute ref="xmi:id"/>
  <xsd:attributeGroup ref="xmi:ObjectAttribs"/>
  <xsd:attribute name="name" type="xsd:string"/>
</xsd:complexType>
<xsd:element name="Name" type="toif:Name"/>
<xsd:complexType name="Description">
  <xsd:choice maxOccurs="unbounded" minOccurs="0">
    <xsd:element ref="xmi:Extension"/>
  </xsd:choice>
  <xsd:attribute ref="xmi:id"/>
  <xsd:attributeGroup ref="xmi:ObjectAttribs"/>
  <xsd:attribute name="text" type="xsd:string"/>
</xsd:complexType>
<xsd:element name="Description" type="toif:Description"/>
<xsd:complexType name="Generator">
  <xsd:complexContent>
    <xsd:extension base="toif:Entity">
      <xsd:choice maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="version" type="toif:Version"/>
        <xsd:element name="description" type="toif:Description"/>
        <xsd:element name="name" type="toif:Name"/>
      </xsd:choice>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="Generator" type="toif:Generator"/>
<xsd:complexType name="Adaptor">
  <xsd:complexContent>
    <xsd:extension base="toif:Entity">
      <xsd:choice maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="name" type="toif:Name"/>
        <xsd:element name="version" type="toif:Version"/>
        <xsd:element name="description" type="toif:Description"/>
      </xsd:choice>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="Adaptor" type="toif:Adaptor"/>
<xsd:complexType name="Version">
  <xsd:choice maxOccurs="unbounded" minOccurs="0">
    <xsd:element ref="xmi:Extension"/>
  </xsd:choice>
  <xsd:attribute ref="xmi:id"/>
  <xsd:attributeGroup ref="xmi:ObjectAttribs"/>
  <xsd:attribute name="version" type="xsd:string"/>
</xsd:complexType>
<xsd:element name="Version" type="toif:Version"/>
<xsd:complexType name="Organization">
  <xsd:complexContent>
    <xsd:extension base="toif:Entity">
      <xsd:choice maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="Description" type="toif:Description"/>
        <xsd:element name="Name" type="toif:Name"/>
        <xsd:element name="Address" type="toif:Address"/>
        <xsd:element name="Phone" type="toif:Phone"/>
        <xsd:element name="EmailAddress" type="toif:EmailAddress"/>
      </xsd:choice>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="Organization" type="toif:Organization"/>
<xsd:complexType name="Vendor">
  <xsd:complexContent>
    <xsd:extension base="toif:Organization"/>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="Vendor" type="toif:Vendor"/>
<xsd:complexType name="Address">
  <xsd:choice maxOccurs="unbounded" minOccurs="0">
    <xsd:element ref="xmi:Extension"/>
  </xsd:choice>
```

```
    <xsd:attribute ref="xmi:id"/>
    <xsd:attributeGroup ref="xmi:ObjectAttribs"/>
    <xsd:attribute name="address" type="xsd:string"/>
  </xsd:complexType>
  <xsd:element name="Address" type="toif:Address"/>
  <xsd:complexType name="Phone">
    <xsd:choice maxOccurs="unbounded" minOccurs="0">
      <xsd:element ref="xmi:Extension"/>
    </xsd:choice>
    <xsd:attribute ref="xmi:id"/>
    <xsd:attributeGroup ref="xmi:ObjectAttribs"/>
    <xsd:attribute name="phone" type="xsd:string"/>
  </xsd:complexType>
  <xsd:element name="Phone" type="toif:Phone"/>
  <xsd:complexType name="EmailAddress">
    <xsd:choice maxOccurs="unbounded" minOccurs="0">
      <xsd:element ref="xmi:Extension"/>
    </xsd:choice>
    <xsd:attribute ref="xmi:id"/>
    <xsd:attributeGroup ref="xmi:ObjectAttribs"/>
    <xsd:attribute name="email" type="xsd:string"/>
  </xsd:complexType>
  <xsd:element name="EmailAddress" type="toif:EmailAddress"/>
  <xsd:complexType name="Person">
    <xsd:complexContent>
      <xsd:extension base="toif:Entity">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="Name" type="toif:Name"/>
          <xsd:element name="EmailAddress" type="toif:EmailAddress"/>
          <xsd:element name="Phone" type="toif:Phone"/>
        </xsd:choice>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="Person" type="toif:Person"/>
  <xsd:complexType name="PersonIsEmployedByOrganizationAsRole">
    <xsd:complexContent>
      <xsd:extension base="toif:Fact">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="role" type="toif:Role"/>
          <xsd:element name="person" type="toif:Person"/>
          <xsd:element name="organization" type="toif:Organization"/>
        </xsd:choice>
        <xsd:attribute name="role" type="xsd:string"/>
        <xsd:attribute name="person" type="xsd:string"/>
        <xsd:attribute name="organization" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="PersonIsEmployedByOrganizationAsRole"
type="toif:PersonIsEmployedByOrganizationAsRole"/>
  <xsd:complexType name="OrganizationIsPartOfOrganizationAsRole">
    <xsd:complexContent>
      <xsd:extension base="toif:Fact">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="organization2" type="toif:Organization"/>
          <xsd:element name="organization1" type="toif:Organization"/>
          <xsd:element name="role" type="toif:Role"/>
        </xsd:choice>
        <xsd:attribute name="organization2" type="xsd:string"/>
        <xsd:attribute name="organization1" type="xsd:string"/>
        <xsd:attribute name="role" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="OrganizationIsPartOfOrganizationAsRole"
type="toif:OrganizationIsPartOfOrganizationAsRole"/>
  <xsd:complexType name="OrganizationIsInvolvedInProjectAsRole">
    <xsd:complexContent>
      <xsd:extension base="toif:Fact">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
```

```
          <xsd:element name="project" type="toif:Project"/>
          <xsd:element name="organization" type="toif:Organization"/>
          <xsd:element name="role" type="toif:Role"/>
        </xsd:choice>
        <xsd:attribute name="project" type="xsd:string"/>
        <xsd:attribute name="organization" type="xsd:string"/>
        <xsd:attribute name="role" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="OrganizationIsInvolvedInProjectAsRole"
type="toif:OrganizationIsInvolvedInProjectAsRole"/>
  <xsd:complexType name="Role">
    <xsd:complexContent>
      <xsd:extension base="toif:Entity">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="name" type="toif:Name"/>
          <xsd:element name="description" type="toif:Description"/>
        </xsd:choice>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="Role" type="toif:Role"/>
  <xsd:complexType name="PersonIsInvolvedInProjectAsRole">
    <xsd:complexContent>
      <xsd:extension base="toif:Fact">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="project" type="toif:Project"/>
          <xsd:element name="role" type="toif:Role"/>
          <xsd:element name="person" type="toif:Person"/>
        </xsd:choice>
        <xsd:attribute name="project" type="xsd:string"/>
        <xsd:attribute name="role" type="xsd:string"/>
        <xsd:attribute name="person" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="PersonIsInvolvedInProjectAsRole"
type="toif:PersonIsInvolvedInProjectAsRole"/>
  <xsd:complexType name="AdaptorSupportsGenerator">
    <xsd:complexContent>
      <xsd:extension base="toif:Fact">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="adaptor" type="toif:Adaptor"/>
          <xsd:element name="generator" type="toif:Generator"/>
        </xsd:choice>
        <xsd:attribute name="adaptor" type="xsd:string"/>
        <xsd:attribute name="generator" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="AdaptorSupportsGenerator" type="toif:AdaptorSupportsGenerator"/>
  <xsd:complexType name="GeneratorIsSuppliedByVendor">
    <xsd:complexContent>
      <xsd:extension base="toif:Fact">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="generator" type="toif:Generator"/>
          <xsd:element name="vendor" type="toif:Vendor"/>
        </xsd:choice>
        <xsd:attribute name="generator" type="xsd:string"/>
        <xsd:attribute name="vendor" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="GeneratorIsSuppliedByVendor" type="toif:GeneratorIsSuppliedByVendor"/>
  <xsd:complexType name="AdaptorIsSuppliedByVendor">
    <xsd:complexContent>
      <xsd:extension base="toif:Fact">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="adaptor" type="toif:Adaptor"/>
          <xsd:element name="vendor" type="toif:Vendor"/>
```

```
      </xsd:choice>
      <xsd:attribute name="adaptor" type="xsd:string"/>
      <xsd:attribute name="vendor" type="xsd:string"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="AdaptorIsSuppliedByVendor" type="toif:AdaptorIsSuppliedByVendor"/>
<xsd:complexType name="TOIFSegment">
  <xsd:choice maxOccurs="unbounded" minOccurs="0">
    <xsd:element name="fact" type="toif:Concept"/>
    <xsd:element name="Name" type="toif:Name"/>
    <xsd:element name="Description" type="toif:Description"/>
    <xsd:element ref="xmi:Extension"/>
  </xsd:choice>
  <xsd:attribute ref="xmi:id"/>
  <xsd:attributeGroup ref="xmi:ObjectAttribs"/>
</xsd:complexType>
<xsd:element name="TOIFSegment" type="toif:TOIFSegment"/>
<xsd:complexType name="Entity">
  <xsd:complexContent>
    <xsd:extension base="toif:Concept"/>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="Entity" type="toif:Entity"/>
<xsd:complexType name="Fact">
  <xsd:complexContent>
    <xsd:extension base="toif:Concept"/>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="Fact" type="toif:Fact"/>
<xsd:complexType name="TOIFSegmentIsGeneratedByPerson">
  <xsd:complexContent>
    <xsd:extension base="toif:Fact">
      <xsd:choice maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="segment" type="toif:TOIFSegment"/>
        <xsd:element name="person" type="toif:Person"/>
      </xsd:choice>
      <xsd:attribute name="segment" type="xsd:string"/>
      <xsd:attribute name="person" type="xsd:string"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="TOIFSegmentIsGeneratedByPerson" type="toif:TOIFSegmentIsGeneratedByPerson"/>
<xsd:complexType name="TOIFSegmentIsProducedByOrganization">
  <xsd:complexContent>
    <xsd:extension base="toif:Fact">
      <xsd:choice maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="segment" type="toif:TOIFSegment"/>
        <xsd:element name="organization" type="toif:Organization"/>
      </xsd:choice>
      <xsd:attribute name="segment" type="xsd:string"/>
      <xsd:attribute name="organization" type="xsd:string"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="TOIFSegmentIsProducedByOrganization"
type="toif:TOIFSegmentIsProducedByOrganization"/>
<xsd:complexType name="Date">
  <xsd:complexContent>
    <xsd:extension base="toif:Entity">
      <xsd:choice maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="date" type="toif:Date"/>
      </xsd:choice>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="Date" type="toif:Date"/>
<xsd:complexType name="TOIFSegmentIsCreatedAtDate">
  <xsd:complexContent>
    <xsd:extension base="toif:Fact">
      <xsd:choice maxOccurs="unbounded" minOccurs="0">
```

```
          <xsd:element name="segment" type="toif:TOIFSegment"/>
          <xsd:element name="date" type="toif:Date"/>
        </xsd:choice>
        <xsd:attribute name="segment" type="xsd:string"/>
        <xsd:attribute name="date" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="TOIFSegmentIsCreatedAtDate" type="toif:TOIFSegmentIsCreatedAtDate"/>
  <xsd:complexType name="TOIFSegmentIsSupervisedByPerson">
    <xsd:complexContent>
      <xsd:extension base="toif:Fact">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="segment" type="toif:TOIFSegment"/>
          <xsd:element name="person" type="toif:Person"/>
        </xsd:choice>
        <xsd:attribute name="segment" type="xsd:string"/>
        <xsd:attribute name="person" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="TOIFSegmentIsSupervisedByPerson"
type="toif:TOIFSegmentIsSupervisedByPerson"/>
  <xsd:complexType name="TOIFSegmentIsOwnedByOrganization">
    <xsd:complexContent>
      <xsd:extension base="toif:Fact">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="organization" type="toif:Organization"/>
          <xsd:element name="segment" type="toif:TOIFSegment"/>
        </xsd:choice>
        <xsd:attribute name="organization" type="xsd:string"/>
        <xsd:attribute name="segment" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="TOIFSegmentIsOwnedByOrganization"
type="toif:TOIFSegmentIsOwnedByOrganization"/>
  <xsd:complexType name="TOIFSegmentIsRelatedToProject">
    <xsd:complexContent>
      <xsd:extension base="toif:Fact">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="segment" type="toif:TOIFSegment"/>
          <xsd:element name="project" type="toif:Project"/>
        </xsd:choice>
        <xsd:attribute name="segment" type="xsd:string"/>
        <xsd:attribute name="project" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="TOIFSegmentIsRelatedToProject" type="toif:TOIFSegmentIsRelatedToProject"/>
  <xsd:complexType name="TOIFSegmentIsGeneratedByGenerator">
    <xsd:complexContent>
      <xsd:extension base="toif:Fact">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="segment" type="toif:TOIFSegment"/>
          <xsd:element name="generator" type="toif:Generator"/>
        </xsd:choice>
        <xsd:attribute name="segment" type="xsd:string"/>
        <xsd:attribute name="generator" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="TOIFSegmentIsGeneratedByGenerator"
type="toif:TOIFSegmentIsGeneratedByGenerator"/>
  <xsd:complexType name="TOIFSegmentIsProcessedByAdaptor">
    <xsd:complexContent>
      <xsd:extension base="toif:Fact">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="adaptor" type="toif:Adaptor"/>
          <xsd:element name="segment" type="toif:TOIFSegment"/>
        </xsd:choice>
```

```
        <xsd:attribute name="adaptor" type="xsd:string"/>
        <xsd:attribute name="segment" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="TOIFSegmentIsProcessedByAdaptor"
type="toif:TOIFSegmentIsProcessedByAdaptor"/>
  <xsd:complexType name="File">
    <xsd:complexContent>
      <xsd:extension base="toif:Entity">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="name" type="toif:Name"/>
          <xsd:element name="version" type="toif:Version"/>
          <xsd:element name="checksum" type="toif:Checksum"/>
        </xsd:choice>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="File" type="toif:File"/>
  <xsd:complexType name="Checksum">
    <xsd:choice maxOccurs="unbounded" minOccurs="0">
      <xsd:element ref="xmi:Extension"/>
    </xsd:choice>
    <xsd:attribute ref="xmi:id"/>
    <xsd:attributeGroup ref="xmi:ObjectAttribs"/>
    <xsd:attribute name="checksum" type="xsd:string"/>
  </xsd:complexType>
  <xsd:element name="Checksum" type="toif:Checksum"/>
  <xsd:complexType name="Directory">
    <xsd:complexContent>
      <xsd:extension base="toif:Entity">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="name" type="toif:Name"/>
        </xsd:choice>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="Directory" type="toif:Directory"/>
  <xsd:complexType name="DirectoryIsContainedInDirectory">
    <xsd:complexContent>
      <xsd:extension base="toif:Fact">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="directory1" type="toif:Directory"/>
          <xsd:element name="directory2" type="toif:Directory"/>
        </xsd:choice>
        <xsd:attribute name="directory1" type="xsd:string"/>
        <xsd:attribute name="directory2" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="DirectoryIsContainedInDirectory"
type="toif:DirectoryIsContainedInDirectory"/>
  <xsd:complexType name="FileIsContainedInDirectory">
    <xsd:complexContent>
      <xsd:extension base="toif:Fact">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="file" type="toif:File"/>
          <xsd:element name="directory" type="toif:Directory"/>
        </xsd:choice>
        <xsd:attribute name="file" type="xsd:string"/>
        <xsd:attribute name="directory" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="FileIsContainedInDirectory" type="toif:FileIsContainedInDirectory"/>
  <xsd:complexType name="Finding">
    <xsd:complexContent>
      <xsd:extension base="toif:Entity"/>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="Finding" type="toif:Finding"/>
```

```
  <xsd:complexType name="WeaknessDescription">
    <xsd:complexContent>
      <xsd:extension base="toif:Entity">
        <xsd:attribute name="text" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="WeaknessDescription" type="toif:WeaknessDescription"/>
  <xsd:complexType name="FindingIsDescribedByWeaknessDescription">
    <xsd:complexContent>
      <xsd:extension base="toif:Fact">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="finding" type="toif:Finding"/>
          <xsd:element name="description" type="toif:WeaknessDescription"/>
        </xsd:choice>
        <xsd:attribute name="finding" type="xsd:string"/>
        <xsd:attribute name="description" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="FindingIsDescribedByWeaknessDescription"
type="toif:FindingIsDescribedByWeaknessDescription"/>
  <xsd:complexType name="CWEIdentifier">
    <xsd:complexContent>
      <xsd:extension base="toif:Entity"/>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="CWEIdentifier" type="toif:CWEIdentifier"/>
  <xsd:complexType name="FindingHasCWEIdentifier">
    <xsd:complexContent>
      <xsd:extension base="toif:Fact">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="finding" type="toif:Finding"/>
          <xsd:element name="cwe" type="toif:CWEIdentifier"/>
        </xsd:choice>
        <xsd:attribute name="finding" type="xsd:string"/>
        <xsd:attribute name="cwe" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="FindingHasCWEIdentifier" type="toif:FindingHasCWEIdentifier"/>

<xsd:complexType name="SFPIdentifier">
    <xsd:complexContent>
      <xsd:extension base="toif:Entity"/>
    </xsd:complexContent>
  </xsd:complexType>

<xsd:element name="SFPIdentifier" type="toif:SFPIdentifier"/>
  <xsd:complexType name="FindingHasSFPIdentifier">
    <xsd:complexContent>
      <xsd:extension base="toif:Fact">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="finding" type="toif:Finding"/>
          <xsd:element name="cwe" type="toif:SFPIdentifier"/>
        </xsd:choice>
        <xsd:attribute name="finding" type="xsd:string"/>
        <xsd:attribute name="sfp" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="FindingHasSFPIdentifier" type="toif:FindingHasSFPIdentifier"/>

<xsd:complexType name="ClusterIdentifier">
    <xsd:complexContent>
      <xsd:extension base="toif:Entity"/>
    </xsd:complexContent>
  </xsd:complexType>

<xsd:element name="ClusterIdentifier" type="toif:ClusterIdentifier"/>
  <xsd:complexType name="FindingHasClusterIdentifier">
```

```
  <xsd:complexContent>
    <xsd:extension base="toif:Fact">
      <xsd:choice maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="finding" type="toif:Finding"/>
        <xsd:element name="cwe" type="toif:ClusterIdentifier"/>
      </xsd:choice>
      <xsd:attribute name="finding" type="xsd:string"/>
      <xsd:attribute name="cluster" type="xsd:string"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="FindingHasClusterIdentifier" type="toif:FindingHasClusterIdentifier"/>

<xsd:complexType name="CodeLocation">
  <xsd:complexContent>
    <xsd:extension base="toif:Entity">
      <xsd:choice maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="linenumber" type="toif:Linenumber"/>
        <xsd:element name="position" type="toif:Position"/>
        <xsd:element name="Offset" type="toif:Offset"/>
      </xsd:choice>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="CodeLocation" type="toif:CodeLocation"/>
<xsd:complexType name="FindingHasCodeLocation">
  <xsd:complexContent>
    <xsd:extension base="toif:Fact">
      <xsd:choice maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="location" type="toif:CodeLocation"/>
        <xsd:element name="finding" type="toif:Finding"/>
      </xsd:choice>
      <xsd:attribute name="location" type="xsd:string"/>
      <xsd:attribute name="finding" type="xsd:string"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="FindingHasCodeLocation" type="toif:FindingHasCodeLocation"/>
<xsd:complexType name="Position">
  <xsd:choice maxOccurs="unbounded" minOccurs="0">
    <xsd:element ref="xmi:Extension"/>
  </xsd:choice>
  <xsd:attribute ref="xmi:id"/>
  <xsd:attributeGroup ref="xmi:ObjectAttribs"/>
  <xsd:attribute name="position" type="xsd:int"/>
</xsd:complexType>
<xsd:element name="Position" type="toif:Position"/>
<xsd:complexType name="Linenumber">
  <xsd:choice maxOccurs="unbounded" minOccurs="0">
    <xsd:element ref="xmi:Extension"/>
  </xsd:choice>
  <xsd:attribute ref="xmi:id"/>
  <xsd:attributeGroup ref="xmi:ObjectAttribs"/>
  <xsd:attribute name="linenumber" type="xsd:int"/>
</xsd:complexType>
<xsd:element name="Linenumber" type="toif:Linenumber"/>
<xsd:complexType name="CodeLocationReferencesFile">
  <xsd:complexContent>
    <xsd:extension base="toif:Fact">
      <xsd:choice maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="CodeLocation" type="toif:CodeLocation"/>
        <xsd:element name="File" type="toif:File"/>
      </xsd:choice>
      <xsd:attribute name="CodeLocation" type="xsd:string"/>
      <xsd:attribute name="File" type="xsd:string"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="CodeLocationReferencesFile" type="toif:CodeLocationReferencesFile"/>
<xsd:complexType name="Offset">
```

```
  <xsd:choice maxOccurs="unbounded" minOccurs="0">
    <xsd:element ref="xmi:Extension"/>
  </xsd:choice>
  <xsd:attribute ref="xmi:id"/>
  <xsd:attributeGroup ref="xmi:ObjectAttribs"/>
  <xsd:attribute name="offset" type="xsd:int"/>
</xsd:complexType>
<xsd:element name="Offset" type="toif:Offset"/>
<xsd:complexType name="Statement">
  <xsd:complexContent>
    <xsd:extension base="toif:Entity"/>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="Statement" type="toif:Statement"/>
<xsd:complexType name="StatementHasCodeLocation">
  <xsd:complexContent>
    <xsd:extension base="toif:Fact">
      <xsd:choice maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="statement" type="toif:Statement"/>
        <xsd:element name="location" type="toif:CodeLocation"/>
      </xsd:choice>
      <xsd:attribute name="statement" type="xsd:string"/>
      <xsd:attribute name="location" type="xsd:string"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="StatementHasCodeLocation" type="toif:StatementHasCodeLocation"/>
<xsd:complexType name="StatementIsInvolvedInFinding">
  <xsd:complexContent>
    <xsd:extension base="toif:Fact">
      <xsd:choice maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="statement" type="toif:Statement"/>
        <xsd:element name="finding" type="toif:Finding"/>
      </xsd:choice>
      <xsd:attribute name="statement" type="xsd:string"/>
      <xsd:attribute name="finding" type="xsd:string"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="StatementIsInvolvedInFinding" type="toif:StatementIsInvolvedInFinding"/>
<xsd:complexType name="StatementIsSinkInFinding">
  <xsd:complexContent>
    <xsd:extension base="toif:Fact">
      <xsd:choice maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="finding" type="toif:Finding"/>
        <xsd:element name="statement" type="toif:Statement"/>
      </xsd:choice>
      <xsd:attribute name="finding" type="xsd:string"/>
      <xsd:attribute name="statement" type="xsd:string"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="StatementIsSinkInFinding" type="toif:StatementIsSinkInFinding"/>
<xsd:complexType name="StatementIsSourceInFinding">
  <xsd:complexContent>
    <xsd:extension base="toif:Fact">
      <xsd:choice maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="statement" type="toif:Statement"/>
        <xsd:element name="finding" type="toif:Finding"/>
      </xsd:choice>
      <xsd:attribute name="statement" type="xsd:string"/>
      <xsd:attribute name="finding" type="xsd:string"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="StatementIsSourceInFinding" type="toif:StatementIsSourceInFinding"/>
<xsd:complexType name="StatementIsPreceededByStatement">
  <xsd:complexContent>
    <xsd:extension base="toif:Fact">
      <xsd:choice maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="statement1" type="toif:Statement"/>
```

```
          <xsd:element name="statement2" type="toif:Statement"/>
        </xsd:choice>
        <xsd:attribute name="statement1" type="xsd:string"/>
        <xsd:attribute name="statement2" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="StatementIsPreceededByStatement"
type="toif:StatementIsPreceededByStatement"/>
  <xsd:complexType name="DataElement">
    <xsd:complexContent>
      <xsd:extension base="toif:Entity">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="name" type="toif:Name"/>
        </xsd:choice>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="DataElement" type="toif:DataElement"/>
  <xsd:complexType name="DataElementIsInvolvedInFinding">
    <xsd:complexContent>
      <xsd:extension base="toif:Fact">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="data" type="toif:DataElement"/>
          <xsd:element name="finding" type="toif:Finding"/>
        </xsd:choice>
        <xsd:attribute name="data" type="xsd:string"/>
        <xsd:attribute name="finding" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="DataElementIsInvolvedInFinding" type="toif:DataElementIsInvolvedInFinding"/>
  <xsd:complexType name="DataElementIsInvolvedInStatement">
    <xsd:complexContent>
      <xsd:extension base="toif:Fact">
        <xsd:choice maxOccurs="unbounded" minOccurs="0">
          <xsd:element name="data" type="toif:DataElement"/>
          <xsd:element name="statement" type="toif:Statement"/>
        </xsd:choice>
        <xsd:attribute name="data" type="xsd:string"/>
        <xsd:attribute name="statement" type="xsd:string"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="DataElementIsInvolvedInStatement"
type="toif:DataElementIsInvolvedInStatement"/>
  <xsd:complexType name="Concept">
    <xsd:choice maxOccurs="unbounded" minOccurs="0">
      <xsd:element ref="xmi:Extension"/>
    </xsd:choice>
    <xsd:attribute ref="xmi:id"/>
    <xsd:attributeGroup ref="xmi:ObjectAttribs"/>
    <xsd:attribute name="id" type="xsd:string"/>
  </xsd:complexType>
  <xsd:element name="Concept" type="toif:Concept"/>
</xsd:schema>
```

5.3 EXAMPLE TOIF XML DATA

This section presents an example of how TOIF data is represented in TOIF XML.

Suppose we want to represent the following hosekeeping facts:

There exists a project TOIF that *has* name “Tool Integration Framework (TOIF)”.

There exists an organization that *has* name "Data Access Technologies"

There exists a person that *has* name "Ed S."

There exists a role that *has* name "Prime Investigator"

There exists a role that *has* name "Director"

Ed S. *is employed by* Data Access Technologies *as* Director

Ed S. *is involved in* TOIF *as* Prime Investigator

The first 5 facts are the so-called existential facts that introduce 5 entities. Entity project will be referenced by an identifier TOIF, other entities will be referenced by their names. The last 2 facts are TOIF facts that refer to the entities. The facts above are represented in SBVR Structured English. Below is the TOIF XML representation of these facts. The facts are arranged into a TOIF segment called "seg1".

Note, that the default EMF ecore representation uses positional references rather than named identifiers of facts. For the purpose of this illustration we transformed the positional identifiers into more readable named identifiers.

The segment also illustrates a single finding with a CWE identifier, an SFP identifier and a Cluster identifier.

```
<?xml version="1.0" encoding="UTF-8"?>
<toif:TOIFSegment xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:toif="http:///toif.ecore">
  <name name="seg1"/>

  <fact xsi:type="toif:Project" id="pr1">
    <name name="Tool Integration Framework (TOIF)"/>
    <description text="DHS SBIR project"/>
  </fact>

  <fact xsi:type="toif:Organization" id="o1">
    <name name="Data Access Technologies"/>
  </fact>

  <fact xsi:type="toif:Person" id="p1">
    <name name="Ed S."/>
  </fact>

  <fact xsi:type="toif:Role" id="r1">
    <name name="Prime Investigator"/>
  </fact>
```

```
  <fact xsi:type="toif:Role" id="r2">
    <name name="Director"/>
  </fact>

  <fact xsi:type="toif:PersonIsInvolvedInProjectAsRole"
project="pr1" role="r1" person="p1"/>

  <fact xsi:type="toif:PersonIsEmployedByOrganizationAsRole"
role="r2" person="p1" organization="o1"/>

  <fact xsi:type="toif:Finding" is="812">
  </fact>

  <fact xsi:type="toif:SFPIdentifier" id="22">
  <name name="SFP--1"/>
  </fact>
  <fact xsi:type="toif:CWEIdentifier" id="23">
  <name name="CWE-398"/>
  </fact>
  <fact xsi:type="toif:ClusterIdentifier" id="24">
  <name name="Observation"/>
  </fact>

  <fact finding="812" sfp="22"
xsi:type="toif:FindingHasSFPIdentifier" id="2123"/>
  <fact finding="812" cwe="23"
xsi:type="toif:FindingHasCWEIdentifier" id="2124"/>
  <fact finding="812" cluster="24"
xsi:type="toif:FindingHasClusterIdentifier" id="2125"/>

</toif:TOIFSegment>
```