SIR file format

Term Definitions

An *XML document* is a well-formed data object according to the XML specification. An XML document contains at least one *element*, and each element may have a set of *attributes* and may be nested within other elements. The (unique) element in the XML document that is not nested is called the root element.
Schemas are documents that define grammar for a class of XML documents. They are an abstract collection of metadata, consisting of a set of schema components. The schema language used for describing validity of SIR file is W3C XML Schema Language. An XML document that has an associated Schema and complies it is said to be *valid*.

SIR Description

A **SIR** (**S**tandard **I**ntermediate Program **R**epresentation) is an XML document used to represent structure of an program. SIR document contains information about the structure of code (main routine, subroutines, loops, their nesting, etc.) and the used data structures. Collected data is latter used by frontend and Periscope GUI to provide an instrumentation outline and thus ease the navigation in the code.
SIR file is generated by the instrumenter for the application which should be analyzed. Collected information forms a tree of all monitored code regions. Structure of an SIR document is described by XML Schema file and its structure is shown on figure 1.

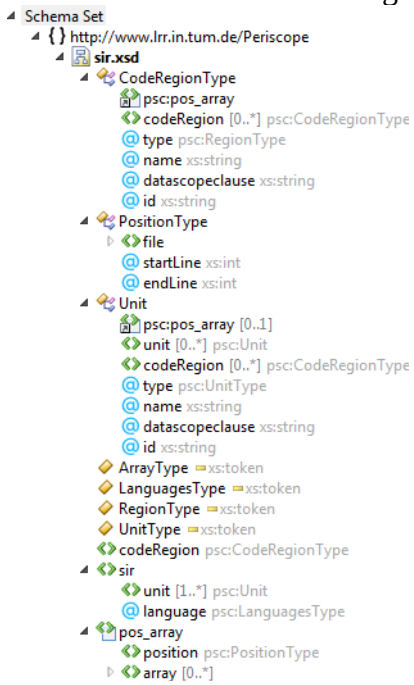

Figure 1. Structure of SIR XML Schema file

As the SIR file is an XML file it contains XML declaration on the beginning of the file. It is declared with `<?xml version="1.0" encoding="UTF-8"?>`. Xmlns is used to define namespace and it is an URI. It could be any identifier to distinguish elements with the same name from

different organizations. One XML document can have more than one namespaces defined. They are distinguished by using different prefixes. SIR file definition uses two namespaces. In the following text all element types will be named with namespace in front of them (e.g. *CodeRegionType* from psc namespace will be named *psc:CodeRegionType*).

The Element sir

The root of any SIR file is given by the *sir* element. An s*ir* element is consisted of an mandatory *unit* element of a *psc:Unit* type and a *language* attribute which is of psc*:LanguagesType* type. Number of occurrences of *unit* elements is not limited from the above. The *language* attribute is of *psc:LanguagesType* type and it can contain any name for programming language. Currently that attribute is limited just to languages supported by our instrumenters; C and Fortran.
XML Schema description for *sir* element:

```
<xs:element name="sir">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="unit" type="psc:Unit" maxOccurs="unbounded" minOccurs="1"/>
    </xs:sequence>
    <xs:attribute name="language" type="psc:LanguageType"/>
  </xs:complexType>
</xs:element>
```

Group element psc:pos_array

Group element *pos_array* is used to define basic information about an array. Information it provide for an array are name of a file, line of a file where it is located, type and size of array. It is consisted of a mandatory element *position* and optional element *array*. Element *array* has three attributes *name* and *type* of a xs:string type and a *nelts* (size) of a xs:positiveInteger type. This element is optional but number of occurrences are not limited from the above.
XML Schema description for *pos_array* group element:

```
<xs:group name="pos_array">
  <xs:sequence>
    <xs:element name="position" type="psc:PositionType" maxOccurs="1"
minOccurs="1"/>
    <xs:element name="array" maxOccurs="unbounded" minOccurs="0">
      <xs:complexType>
        <xs:attribute name="type" type="psc:ArrayType"/>
        <xs:attribute name="name" type="xs:string"/>
        <xs:attribute name="nelts" type="xs:positiveInteger"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:group>
```

SIR element types

Currently XML Schema file for SIR has three complex and four simple element types.
Complex element types are:

1. *CodeRegionType*
2. *PositionType*
3. *Unit*

Simple element types are:

1. *RegionType*
2. *UnitType*

3. *LanguagesType*
4. *ArrayType*

Complex element types

psc:CodeRegionType

Element type *CodeRegionType* is used to determine interesting parts of the source code for Periscope. It is defined with mandatory group element called *psc:pos_array* and optional element called *codeRegion* and four attributes. The optional element *codeRegion* is of a psc:CodeRegionType and is defining nested code regions. *CodeRegionType* element type provides *type* attribute of a *psc:RegionType* type and a *name*, *datascopeclause* and *id* attributes of a *xs:string* type.

XML Schema description for *CodeRegionType* complex element type:

```
<xs:complexType name="CodeRegionType">
  <xs:sequence>
    <xs:group ref="psc:pos_array" maxOccurs="1" minOccurs="1"/>
    <xs:element name="codeRegion" type="psc:CodeRegionType" maxOccurs="unbounded"
                                                          minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="type" type="psc:RegionType"/>
  <xs:attribute name="name" type="xs:string"/>
  <xs:attribute name="datascopeclause" type="xs:string"/>
  <xs:attribute name="id" type="xs:string"/>
</xs:complexType>
```

psc:PositionType

PositionType is used to define the file name and the position in a file. It is defined with an embedded element *file* with attribute *name* and two attributes *startLine* and a *endLine*. Embedded attribute is of a *xs:string* type. *startLine* and *endLine* attributes are of *xs:positiveInteger* type.

XML Schema description for *PositionType* complex element type:

```
<xs:complexType name="PositionType">
  <xs:sequence>
    <xs:element name="file">
      <xs:complexType>
        <xs:attribute name="name" type="xs:string"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="startLine" type="xs:positiveInteger"/>
  <xs:attribute name="endLine" type="xs:positiveInteger"/>
</xs:complexType>
```

psc:Unit

A complex element type *Unit* is used to define program structure. It is defined with optional group element called psc:*pos_array*, optional element called *unit*, and a mandatory element called *codeRegion* and four attributes. The element *unit* is of a type *psc:Unit*. The *codeRegion* element is of a *psc:CodeRegionType* type. *Unit* element type provides *type* attribute of a *psc:UnitType* type and a *name*, *datascopeclause* and *id* attributes of a *xs:string* type.

XML Schema description for *Unit* complex element type:

```
<xs:complexType name="Unit">
  <xs:sequence>
    <xs:group ref="psc:pos_array" maxOccurs="1" minOccurs="0"/>
    <xs:element name="unit" type="psc:Unit" maxOccurs="unbounded" minOccurs="0"/>
    <xs:element name="codeRegion" type="psc:CodeRegionType" maxOccurs="unbounded"
```

```
                                                    minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="type" type="psc:UnitType"/>
  <xs:attribute name="name" type="xs:string"/>
  <xs:attribute name="datascopeclause" type="xs:string"/>
  <xs:attribute name="id" type="xs:string"/>
</xs:complexType>
```

Simple enumerated element types

All simpleType elements uses *xs:enumeration*, which is a case sensitive list of acceptable values.

psc:RegionType

psc:RegionType is used to distinguish different types of regions available in the source code. This types is using enumerations to define all regions interesting for the Periscope.

Programming language regions

- userRegion
  Does not correspond to any standard code region. Helps user to specify regions which he want to monitor e.g. phase region.
- call
  In Fortran, corresponds to a function or subroutine call or to a statement for dynamic storage allocation or deallocation (ALLOCATE, DEALOCATE, and NULLIFY). In C, it corresponds to function call, dynamic storage allocation and deallocation.
- loop
  Corresponds to any kind of loop in the input program: for, while, do…while in C and DO, DO WHILE (but not FORALL) in Fortran.
- vect (Fortran specific)
  Corresponds to a region executed by several threads in parallel.
- forall (Fortran specific)
  Corresponds to the FORALL construct in Fortran.
- io (Fortran specific)
  Corresponds to an IO statement in Fortran (like PRINT or OPEN).

Motivated by OpenMP, there is also defined set of a parallel regions embedded in RegionType which can be applied to any similar shared-memory paradigm. Table 1 shows the mapping of OpenMP directives to the corresponding psc:RegionType.

OpenMP directive	psc:RegionType in SIR
PARALLEL	parallelRegion
DO	parallelLoop
SECTIONS	parallelSections
SECTION	parallelSection
SINGLE	parallelSingle
PARALLEL WORKSHARE	parallelWorkshare
MASTER	parallelMaster
CRITICAL	parallelCriticalSection
ATOMIC	parallelAtomic

BARRIER	parallelBarrier
ORDERED	parallelOrdered
WORKSHARE	workshare
WORKSHARE SECTIONS	workshareSections
WORKSHARE DO	workshareLoop
PARALLEL TASK	parallelTask
TASKWAIT	taskWait

Table 1. Mapping of Fortran OpenMP directives to *psc:RegionType*

OpenMP regions

- parallelRegion
  Corresponds to a region executed by several threads in parallel.
- parallelLoop
  Corresponds to a work-sharing construct that distributes the iterations of a loop among several threads.
- parallelSections
  Corresponds to a work-sharing construct that distributes the execution of several code regions among several threads.
- parallelSection
  Correspond to a construct that defines block of code that can be executed in parallel.
- parallelSingle
  Correspond to a construct used to group a sequence of code regions that must be executed by only one thread.
- parallelWorkshare
  Corresponds to a parallel work-sharing construct that distributes the execution of several code regions in parallel among several threads.
- parallelMaster
  Used to group a sequence of code regions that must be executed by only one thread.
- parallelCriticalSection
  Corresponds to a construct that restricts execution of the associated structured block to a single thread at a time.
- parallelAtomic
  Used to inform that an assignment is performed atomically.
- parallelBarrier
  Corresponds to a language construct that synchronizes all threads within the dynamic scope of a parallel region.
- parallelOrdered
  Corresponds to a construct that ensures that a sequence of code regions is executed in the order in which iteration would be executed in a sequential execution of a loop.
- workshare
  Corresponds to a work-sharing construct that distributes the execution of the enclosed structured block into separate units of work, each executed only once by one thread.
- workshareSections
  Corresponds to a combined work-sharing construct that distributes the execution of the enclosed structured block among and executed by the encountering teams of threads.

- workshareLoop
  Corresponds to a combined work-sharing construct that distributes the execution of the iterations of a loop among several threads.
- parallelTask
  Corresponds to a combined constructs that defines explicit parallel task.
- taskWait
  Corresponds to a region that specifies a wait on the completion of child tasks of the current task.

XML Schema description for *RegionType* simple element type:

```
<xs:simpleType name="RegionType">
  <xs:restriction base="xs:token">
    <xs:enumeration value="userRegion"/>
    <xs:enumeration value="call"/>
    <xs:enumeration value="loop"/>
    <xs:enumeration value="vect"/>
    <xs:enumeration value="forall"/>
    <xs:enumeration value="io"/>
    <xs:enumeration value="parallelRegion"/>
    <xs:enumeration value="parallelLoop"/>
    <xs:enumeration value="parallelSections"/>
    <xs:enumeration value="parallelSection"/>
    <xs:enumeration value="parallelSingle"/>
    <xs:enumeration value="parallelWorkshare"/>
    <xs:enumeration value="parallelMaster"/>
    <xs:enumeration value="parallelCriticalSection"/>
    <xs:enumeration value="parallelAtomic"/>
    <xs:enumeration value="parallelBarrier"/>
    <xs:enumeration value="parallelOrdered"/>
    <xs:enumeration value="workshare"/>
    <xs:enumeration value="workshareSections"/>
    <xs:enumeration value="workshareLoop"/>
    <xs:enumeration value="parallelTask"/>
    <xs:enumeration value="taskWait"/>
  </xs:restriction>
</xs:simpleType>
```

psc:UnitType

The *UnitType* type is used to distinguish different types of program units. This type is using enumerations to define all interesting regions for the Periscope. The supported program units are:

1. program
2. subroutine
3. function

XML Schema description for *UnitType* simple element type:

```
<xs:simpleType name="UnitType">
  <xs:restriction base="xs:token">
    <xs:enumeration value="program"/>
    <xs:enumeration value="subroutine"/>
    <xs:enumeration value="function"/>
  </xs:restriction>
</xs:simpleType>
```

psc:LanguageType

The *LanguageType* type is used to distinguish in which programming language input program is written in. This type is using enumerations to define all interesting languages for the Periscope. The supported languages are:

1. c
2. fortran

XML Schema description for *LanguageType* simple element type:

```
<xs:simpleType name="LanguagesType">
  <xs:restriction base="xs:token">
    <xs:enumeration value="c"/>
    <xs:enumeration value="fortran"/>
  </xs:restriction>
</xs:simpleType>
```

psc:ArrayType

The ArrayType type is used to distinguished data type of an array. This type is using enumerations to define interesting data types for Periscope. Currently it provides data types only for Fortran as only Fortran instrumenter uses them. The supported data types are:

1. INTEGER
2. REAL
3. CHARACTER
4. LOGICAL
5. COMPLEX
6. DOUBLE
7. DERIVED

XML Schema description for *ArrayType* simple element type:

```
<xs:simpleType name="ArrayType">
  <xs:restriction base="xs:token">
    <xs:enumeration value="INTEGER"/>
    <xs:enumeration value="REAL"/>
    <xs:enumeration value="CHARACTER"/>
    <xs:enumeration value="LOGICAL"/>
    <xs:enumeration value="COMPLEX"/>
    <xs:enumeration value="DOUBLE"/>
    <xs:enumeration value="DERIVED"/>
  </xs:restriction>
</xs:simpleType>
```

Example of SIR file

Example of SIR file:

```
<?xml version="1.0" encoding="UTF-8"?>
<sir xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.lrr.in.tum.de/Periscope"
xsi:schemaLocation="http://www.lrr.in.tum.de/Periscope sir.xsd" language="fortran">
  <unit type="program" name="ADD" id="42-1">
    <position startLine="1" endLine="23">
      <file name="add.f90"/>
    </position>
    <array name="A" type="DOUBLE"  nelts="10000"/>
    <codeRegion type="call" name="MPI_INIT" id="42-7">
      <position startLine="7" endLine="7">
        <file name="add.f90"/>
      </position>
```

```
    </codeRegion>
    <codeRegion type="forall" name="" id="42-9">
      <position startLine="9" endLine="17">
        <file name="add.f90"/>
      </position>
      <array name="A" type="DOUBLE"  nelts="10000"/>
      <codeRegion type="userRegion" name="" id="42-11">
        <position startLine="11" endLine="15">
          <file name="add.f90"/>
        </position>
        <array name="A" type="DOUBLE"  nelts="10000"/>
        <codeRegion type="loop" name="" id="42-12">
          <position startLine="12" endLine="14">
            <file name="add.f90"/>
          </position>
          <array name="A" type="DOUBLE"  nelts="10000"/>
        </codeRegion>
      </codeRegion>
    </codeRegion>
    <codeRegion type="call" name="MPI_FINALIZE" id="42-21">
      <position startLine="21" endLine="21">
        <file name="add.f90"/>
      </position>
    </codeRegion>
  </unit>
</sir>
```

XML Schema file for the SIR

Complete listing of an XML Schema file for the SIR:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.lrr.in.tum.de/Periscope"
            xmlns:psc="http://www.lrr.in.tum.de/Periscope"
            attributeFormDefault="unqualified" elementFormDefault="qualified">

  <xs:element name="sir">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="unit" type="psc:Unit" maxOccurs="unbounded"
minOccurs="1"/>
      </xs:sequence>
      <xs:attribute name="language" type="psc:LanguagesType"/>
    </xs:complexType>
    </xs:element>

  <xs:complexType name="Unit">
    <xs:sequence>
      <xs:group ref="psc:pos_array" maxOccurs="1" minOccurs="0"/>
      <xs:element name="unit" type="psc:Unit" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element name="codeRegion" type="psc:CodeRegionType" maxOccurs="unbounded"
                                                          minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="type" type="psc:UnitType"/>
    <xs:attribute name="name" type="xs:string"/>
    <xs:attribute name="datascopeclause" type="xs:string"/>
    <xs:attribute name="id" type="xs:string"/>
  </xs:complexType>

  <xs:complexType name="PositionType">
```

```
    <xs:sequence>
      <xs:element name="file">
        <xs:complexType>
          <xs:attribute name="name" type="xs:string"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="startLine" type="xs:positiveInteger"/>
    <xs:attribute name="endLine" type="xs:positiveInteger"/>
  </xs:complexType>

  <xs:complexType name="CodeRegionType">
    <xs:sequence>
      <xs:group ref="psc:pos_array" maxOccurs="1" minOccurs="1"/>
      <xs:element name="codeRegion" type="psc:CodeRegionType" maxOccurs="unbounded"
                                                              minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="type" type="psc:RegionType"/>
    <xs:attribute name="name" type="xs:string"/>
    <xs:attribute name="datascopeclause" type="xs:string"/>
    <xs:attribute name="id" type="xs:string"/>
  </xs:complexType>

  <xs:group name="pos_array">
    <xs:sequence>
      <xs:element name="position" type="psc:PositionType" maxOccurs="1"
minOccurs="1"/>
      <xs:element name="array" maxOccurs="unbounded" minOccurs="0">
        <xs:complexType>
          <xs:attribute name="type" type="psc:ArrayType"/>
          <xs:attribute name="name" type="xs:string"/>
          <xs:attribute name="nelts" type="xs:positiveInteger"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:element name="codeRegion" type="psc:CodeRegionType"/>

  <xs:simpleType name="RegionType">
    <xs:restriction base="xs:token">
      <xs:enumeration value="userRegion"/>
      <xs:enumeration value="call"/>
      <xs:enumeration value="loop"/>
      <xs:enumeration value="vect"/>
      <xs:enumeration value="forall"/>
      <xs:enumeration value="io"/>
      <xs:enumeration value="parallelRegion"/>
      <xs:enumeration value="parallelLoop"/>
      <xs:enumeration value="parallelSections"/>
      <xs:enumeration value="parallelSection"/>
      <xs:enumeration value="parallelSingle"/>
      <xs:enumeration value="parallelWorkshare"/>
      <xs:enumeration value="parallelMaster"/>
      <xs:enumeration value="parallelCriticalSection"/>
      <xs:enumeration value="parallelAtomic"/>
      <xs:enumeration value="parallelBarrier"/>
      <xs:enumeration value="parallelOrdered"/>
      <xs:enumeration value="workshare"/>
      <xs:enumeration value="workshareSections"/>
```

```
      <xs:enumeration value="workshareLoop"/>
      <xs:enumeration value="parallelTask"/>
      <xs:enumeration value="taskWait"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="UnitType">
    <xs:restriction base="xs:token">
      <xs:enumeration value="program"/>
      <xs:enumeration value="subroutine"/>
      <xs:enumeration value="function"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="LanguagesType">
    <xs:restriction base="xs:token">
      <xs:enumeration value="c"/>
      <xs:enumeration value="fortran"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="ArrayType">
    <xs:restriction base="xs:token">
      <xs:enumeration value="INTEGER"/>
      <xs:enumeration value="REAL"/>
      <xs:enumeration value="CHARACTER"/>
      <xs:enumeration value="LOGICAL"/>
      <xs:enumeration value="COMPLEX"/>
      <xs:enumeration value="DOUBLE"/>
      <xs:enumeration value="DERIVED"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```