INTERNATIONAL STANDARD

ISO/IEC 19757-3

First edition
2006-06-01

Information technology — Document Schema Definition Languages (DSDL) —

Part 3: Rule-based validation — Schematron

Technologies de l'information — Langages de définition de schéma de documents (DSDL) —

Partie 3: Validation de règles orientées — Schematron

Reference number
ISO/IEC 19757-3:2006(E)

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 19757-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 34, *Document description and processing languages*.

ISO/IEC 19757 consists of the following parts, under the general title *Information technology — Document Schema Definition Languages (DSDL)*:

— *Part 1: Overview*

— *Part 2: Regular-grammar-based validation — RELAX NG*

— *Part 3: Rule-based validation — Schematron*

— *Part 4: Namespace-based Validation Dispatching Language — NVDL*

The following parts are under preparation:

— *Part 5: Datatypes*

— *Part 6: Path-based integrity constraints*

— *Part 7: Character repertoire description language — CRDL*

— *Part 8: Document schema renaming language — DSRL*

— *Part 9: Datatype- and namespace-aware DTDs*

— *Part 10: Validation management*

Introduction

ISO/IEC 19757 defines a set of Document Schema Definition Languages (DSDL) that can be used to specify one or more validation processes performed against Extensible Markup Language (XML) or Standard Generalized Markup Language (SGML) documents. (XML is an application profile SGML, ISO 8879:1986.)

A document model is an expression of the constraints to be placed on the structure and content of documents to be validated with the model. A number of technologies have been developed through various formal and informal consortia since the development of Document Type Definitions (DTD) as part of ISO 8879, notably by the World Wide Web Consortium (W3C) and the Organization for the Advancement of Structured Information Standards (OASIS). A number of validation technologies are standardized in DSDL to complement those already available as standards or from industry.

To validate that a structured document conforms to specified constraints in structure and content relieves the potentially many applications acting on the document from having to duplicate the task of confirming that such requirements have been met. Historically, such tasks and expressions have been developed and utilized in isolation, without consideration of how the features and functionality available in other technologies might enhance validation objectives.

The main objective of ISO/IEC 19757 is to bring together different validation-related tasks and expressions to form a single extensible framework that allows technologies to work in series or in parallel to produce a single or a set of validation results. The extensibility of DSDL accommodates validation technologies not yet designed or specified.

In the past, different design and use criteria have led users to choose different validation technologies for different portions of their information. Bringing together information within a single XML document sometimes prevents existing document models from being used to validate sections of data. By providing an integrated suite of constraint description languages that can be applied to different subsets of a single XML document, ISO/IEC 19757 allows different validation technologies to be integrated under a well-defined validation policy.

This part of ISO/IEC 19757 is based on the Schematron[1] assertion language. The `let` element is based on XCSL[2]. Other features arise from the half-dozen early Open Source implementations of Schematron in diverse programming languages and from discussions in electronic forums by Schematron users and implementers.

The structure of this part of ISO/IEC 19757 is as follows. Clause 5 describes the syntax of an ISO Schematron schema. Clause 6 describes the semantics of a correct ISO Schematron schema; the semantics specify when a document is valid with respect to an ISO Schematron schema. Clause 7 describes conformance requirements for implementations of ISO Schematron validators. Annex A is a normative annex providing the ISO/IEC 19757-2 (RELAX NG) schema for ISO Schematron. Annex B is a normative annex providing the ISO Schematron schema for constraints in ISO Schematron that cannot be expressed by the schema of Annex A. Annex C is a normative annex providing the default query language binding to XSLT. Annex D is an informative annex providing a ISO/IEC 19757-2 (RELAX NG compact syntax) schema and corresponding ISO Schematron schema for a simple XML language Schematron Validation Report Language. Annex E is an informative annex providing motivating design requirements for ISO Schematron. Annex F is a normative annex allowing certain Schematron elements to be used in external vocabularies. Annex G is an informative annex with a simple example of a multi-lingual schema.

Considered as a document type, a Schematron schema contains natural-language assertions concerning a set of documents, marked up with various elements and attributes for testing these natural-language assertions, and for simplifying and grouping assertions.

Considered theoretically, a Schematron schema reduces to a non-chaining rule system whose terms are Boolean functions invoking an external query language on the instance and other visible XML documents, with syntactic features to reduce specification size and to allow efficient implementation.

Considered analytically, Schematron has two characteristic high-level abstractions: the pattern and the phase. These allow the representation of non-regular, non-sequential constraints that ISO/IEC 19757-2 cannot specify, and various dynamic or contingent constraints.

Information technology — Document Schema Definition Languages (DSDL) —

Part 3:
Rule-based validation — Schematron

1 Scope

This part of ISO/IEC 19757 specifies Schematron, a schema language for XML. This part of ISO/IEC 19757 establishes requirements for Schematron schemas and specifies when an XML document matches the patterns specified by a Schematron schema.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

NOTE Each of the following documents has a unique identifier that is used to cite the document in the text. The unique identifier consists of the part of the reference up to the first comma.

W3C XML 1.0, *Extensible Markup Language (XML) 1.0 (Third Edition)*, W3C Recommendation, 04 February 2004

XPath, *XML Path Language (XPath) Version 1.0*, W3C Recommendation, 16 November 1999

XSLT, *XSL Transformations (XSLT) Version 1.0*, W3C Recommendation, 16 November 1999

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1 abstract pattern

pattern in a rule that has been parameterized to enable reuse

3.2 abstract rule

collection of assertions which can be included in other rules but which does not fire itself

3.3 active pattern

pattern belonging to the active phase

3.4 active phase

one particular phase, whose patterns are used for validation

3.5 assertion

natural-language assertion with corresponding assertion test and ancillary attributes: assertions are marked up with `assert` and `report` elements

3.6 assertion test

assertion modelled or implemented by a Boolean query; an assertion test "succeeds" or "fails"

3.7 correct schema

schema that satisfies all the requirements of this part of ISO/IEC 19757

3.8 diagnostic

named natural language statements providing information to end-users of validators concerning the expected and actual values together with repair hints

3.9 elaborated rule context expression

single rule context expression which explicitly disallows items selected by lexically previous rule contexts in the same pattern

3.10 good schema

correct schema with queries which terminate and do not add constraints to those of the natural-language assertions.

NOTE It may not be possible to compute that a schema is good.

3.11 implementation

implementation of a Schematron validator

3.12 name

token with no whitespace characters

3.13 natural-language assertion

natural-language statement expressing some part of a pattern; a natural-language assertion is "met" or "unmet"

3.14 pattern

named structure in instances specified in a schema by a lexically-ordered collection of rules

3.15 phase

named, unordered collection of patterns; patterns may belong to more than one phase; two names, `#ALL` and `#DEFAULT`, are reserved with particular meanings

3.16 progressive validation

validation of constraints in stages determined or grouped to some extent by the schema author rather than, for example, entirely determined by document order

3.17 query language binding

named set, specified in a document called a Query Language Binding, of the languages and conventions used for assertion tests, rule-context expressions and so on, by a particular Schematron implementation

NOTE 1 Schematron is defined as a framework, with a default query language binding, but other query language bindings are possible.

NOTE 2 6.4 specifies the information to be required by a query language binding and Annex C defines the default query language binding for Schematron.

3.18 rule

unordered collection of assertions with a rule-context expression and ancillary attributes

3.19 rule context

element or other information item used for assertion tests; a rule is said to fire when an information item matches the rule context

3.20 rule-context expression

a query to specify subjects; a rule-context is said to match an information item when that information item has not been matched by any lexically-previous rule context expressions in the same pattern and the information item is one of the information items that the query would specify

3.21 schema

specification of a set of XML documents

3.22 subject

particular information item which corresponds to the object of interest of the natural-language assertions and typically is matched by the context expression of a rule

3.23 valid with respect to a schema

member of the set of XML documents described by the schema: an instance document is valid if no assertion tests in fired rules of active patterns fail

3.24 variable

constant value, evaluated within the parent schema, phase, pattern or rule and scoped within the parent schema, phase, pattern or rule

4 Notation

4.1 XPath

This part of ISO/IEC 19757 uses XPath to identify information items in Schematron schemas.

4.2 Predicate Logic

This part of ISO/IEC 19757 uses predicate logic to express the semantics of Schematron schema. The following symbols are defined for use in s6.3:

— ()
Grouping delimiters

— ∀
"for all". Prefix operator.

— ¬
"not". Prefix operator.

— ∈
"is member of", in set operative sense. Prefix operator.

— ,
"and" (sequence). Infix operator.

— :
"where". Such that. Infix operator.

— ⇔
"if and only if". Infix operator.

5 Syntax

5.1 Well-formedness

A Schematron schema shall be a well-formed XML document, according to the version of XML used.

5.2 Namespace

All elements shown in the grammar for Schematron are qualified with the namespace URI:

`http://purl.oclc.org/dsdl/schematron`

In subsequent clauses, the prefix `sch` is taken as bound to the Schematron namespace URI for exposition purposes. The prefix `sch` is not reserved or required by this part of ISO/IEC 19757.

Any element can also have foreign attributes in addition to the attributes shown in the grammar. A foreign attribute is an attribute with a name whose namespace URI is neither the empty string nor the Schematron namespace URI. Any non-empty element may have foreign child elements in addition to the child elements shown in the grammar. A foreign element is an element with a name whose namespace URI is not the Schematron namespace URI. There are no constraints on the relative position of foreign child elements with respect to other child elements.

5.3 Whitespace

Any element can also have as children strings that consist entirely of whitespace characters, where a whitespace character is one of U+0020, U+0009, U+000D or U+000A. There are no constraints on the relative position of whitespace string children with respect to child elements.

NOTE Leading and trailing whitespace should be stripped from attributes defined by this part. Whitespace should be collapsed in elements defined by this part that allow text. Whitespace may be stripped from elements defined by this part that do not allow text.

5.4 Core Elements

The grammar for Schematron elements is given in Annex A.

5.4.1 `active` element

The required `pattern` attribute is a reference to a pattern that is active in the current phase.

5.4.2 `assert` element

An assertion made about the context nodes. The data content is a natural-language assertion. The required `test` attribute is an assertion test evaluated in the current context. If the test evaluates positive, the assertion succeeds. The optional `diagnostics` attribute is a reference to further diagnostic information.

The natural-language assertion shall be a positive statement of a constraint.

NOTE The natural-language assertion may contain information about actual values in addition to expected values and may contain diagnostic information. Users should note, however, that the `diagnostic` element is provided for such information to encourage clear statement of the natural-language assertion.

The `icon`, `see` and `fpi` attributes allow rich interfaces and documentation. They are defined below.

The `flag` attribute allows more detailed outcomes. It is defined below.

The `role` and `subject` attributes allow explicit identification of some part of a pattern. They are defined below.

5.4.3 `extends` element

Abstract rules are named lists of assertions without a context expression. The required `rule` attribute references an abstract rule. The current rule uses all the assertions from the abstract rule it extends.

5.4.4 `include` element

The required `href` attribute references an external well-formed XML document whose document element is a Schematron element of a type which is allowed by the grammar for Schematron at the current position in the schema. The external document is inserted in place of the `include` element.

5.4.5 `let` element

A declaration of a named variable. If the let element is the child of a `rule` element, the variable is calculated and scoped to the current rule and context. Otherwise, the variable is calculated with the context of the instance document root.

The required `name` attribute is the name of the variable. The required `value` attribute is an expression evaluated in the current context.

It is an error to reference a variable that has not been defined in the current schema, phase, pattern, or rule, if the query language binding allows this to be determined reliably. It is an error for a variable to be multiply defined in the current schema, phase, pattern and rule.

The variable is substituted into assertion tests and other expressions in the same rule before the test or expression is evaluated. The query language binding specifies which lexical conventions are used to detect references to variables.

An implementation may provide a facility to override the values of top-level variables specified by `let` elements under the `schema` element. For example, an implementation may allow top-level variables to be supplied on the command line. The values provided are strings or data objects, not expressions.

5.4.6 `name` element

Provides the names of nodes from the instance document to allow clearer assertions and diagnostics. The optional `path` attribute is an expression evaluated in the current context that returns a string that is the name of a node. In the latter case, the name of the node is used.

An implementation which does not report natural-language assertions is not required to make use of this element.

5.4.7 `ns` element

Specification of a namespace prefix and URI. The required `prefix` attribute is an XML name with no colon character. The required `uri` attribute is a namespace URI.

NOTE Because the characters allowed as names may change in versions of XML subsequent to W3C XML 1.0, the ISO/IEC 19757-2 (RELAX NG Compact Syntax) schema for Schematron does not constrain the prefix to particular characters.

In an ISO Schematron schema, namespace prefixes in context expressions, assertion tests and other query expressions should use the namespace bindings provided by this element. Namespace prefixes should not use the namespace bindings in scope for element and attribute names.

5.4.8 `param` element

A name-value pair providing parameters for an abstract pattern. The required `name` attribute is an XML name with no colon. The required `value` attribute is a fragment of a query.

5.4.9 `pattern` element

A structure, simple or complex. A set of rules giving constraints that are in some way related. The `id` attribute provides a unique name for the pattern and is required for abstract patterns.

The `title` and `p` elements allow rich documentation.

The `icon`, `see` and `fpi` attributes allow rich interfaces and documentation.

When a `pattern` element has the attribute `abstract` with a value `true`, then the pattern defines an abstract pattern. An abstract pattern shall not have a `is-a` attribute and shall have an `id` attribute.

Abstract patterns allow a common definition mechanism for structures which use different names and paths, but which are at heart the same. For example, there are different table markup languages, but they all can be in large part represented as an abstract pattern where a table contains rows and rows contain entries, as defined in the following example using the default query language binding:

```
<sch:pattern abstract="true" id="table">
    <sch:rule context="$table">
        <sch:assert test="$row">
        The element <name/> is a table. Tables contain rows.
        </sch:assert>
    </sch:rule>

    <sch:rule context="$row">
        <sch:assert test="$entry">
        The element <name/> is a table row. Rows contain entries.
        </sch:assert>
    </sch:rule>

</sch:pattern>
```

When a `pattern` element has the attribute `is-a` with a value specifying the name of an abstract pattern, then the pattern is an instance of an abstract pattern. Such a pattern shall not contain any `rule` elements, but shall have `param` elements for all parameters used in the abstract pattern.

The following example uses the abstract pattern for tables given above to create three patterns for tables with different names or structures.

```
<sch:pattern is-a="table" id="HTML_Table">
    <sch:param name="table" value="table"/>
    <sch:param name="row"   value="tr"/>
    <sch:param name="entry" value="td|th"/>
</sch:pattern>

<sch:pattern is-a="table" id="CALS_Table">
    <sch:param name="table" value="table"/>
    <sch:param name="row"   value=".//row"/>
    <sch:param name="entry" value="cell"/>
</sch:pattern>

<sch:pattern is-a="table" id="calendar">
```

```
    <sch:param name="table" value="calendar/year"/>
    <sch:param name="row"   value="week"/>
    <sch:param name="entry" value="day"/>
</sch:pattern>
```

When creating an instance of an abstract pattern, the parameter values supplied by the `param` element replace the parameter references used in the abstract patterns. The examples above use the default query language binding in which the character **$** is used as the delimiter for parameter references.

Thus, given the abstract patterns defined earlier in this clause, the patterns defined above are equivalent to the following, with the `id` elements shown expanded:

```
<sch:pattern id="HTML_table">
    <sch:rule context="table">
        <sch:assert test="tr">
        The element table is a table. Tables containing rows.
        </sch:assert>
    </sch:rule>

    <sch:rule context="tr">
        <sch:assert test="td|th">
        The element tr is a table row. Rows contain entries.
        </sch:assert>
    </sch:rule>

</sch:pattern>

<sch:pattern id="CALS_table">
    <sch:rule context="table">
        <sch:assert test=".//row">
        The element table is a table. Tables containing rows.
        </sch:assert>
    </sch:rule>

    <sch:rule context=".//row">
        <sch:assert test="cell">
        The element row is a table row. Rows contain entries.
        </sch:assert>
    </sch:rule>

</sch:pattern>

<sch:pattern id="calendar">
    <sch:rule context="calendar/year">
        <sch:assert test="week">
        The element year is a table. Tables containing rows.
        </sch:assert>
    </sch:rule>

    <sch:rule context="week">
        <sch:assert test="day">
        The element week is a table row. Rows contain entries.
        </sch:assert>
    </sch:rule>

</sch:pattern>
```

5.4.10 phase element

A grouping of patterns, to name and declare variations in schemas, for example, to support progressive validation. The required id attribute is the name of the phase. The implementation determines which phase to use for validating documents, for example by user command.

Two names, #ALL and #DEFAULT, have special meanings. The name #ALL is reserved and available for use by implementations to denote that all patterns are active. The name #DEFAULT is reserved and available for use by implementations to denote that the name given in the defaultPhase attribute on the schema element should be used. If no defaultPhase is specified, then all patterns are active.

NOTE The names #ALL and #DEFAULT shall not be used in a Schematron schema. They are for use when invoking or configuring schema validation, for example as a command-line parameter.

The icon, see and fpi attributes allow rich interfaces and documentation.

5.4.11 report element

An assertion made about the context nodes. The data content is a natural-language assertion. The required test attribute is an assertion test evaluated in the current context. If the test evaluates positive, the report succeeds. The optional diagnostics attribute is a reference to further diagnostic information.

The natural-language assertion shall be a positive statement of a found pattern or a negative statement of a constraint.

NOTE The natural-language assertion may contain information about actual values in addition to expected values and may contain diagnostic information. Users should note, however, that the diagnostic element is provided for such information to encourage clear statement of the natural-language assertion.

The icon, see and fpi attributes allow rich interfaces and documentation. They are defined below.

The flag attribute allows more detailed outcomes. It is defined below.

The role and subject attributes allow explicit identification of some part of a pattern. They are defined below.

5.4.12 rule element

A list of assertions tested within the context specified by the required context attribute. The context attribute specifies the rule context expression.

NOTE It is not an error if a rule never fires in a document. In order to test that a document always has some context, a new pattern should be created from the context of the document, with an assertion requiring the element or attribute.

The icon, see and fpi attributes allow rich interfaces and documentation.

The flag attribute allows more detailed outcomes. It is defined below.

The role and subject attributes allow explicit identification of some part of a pattern as part of the validation outcome. They are defined below.

When the rule element has the attribute abstract with a value true, then the rule is an abstract rule. An abstract rule shall not have a context attribute. An abstract rule is a list of assertions that will be invoked by other rules belonging to the same pattern using the extends element. Abstract rules provide a mechanism for reducing schema size.

5.4.13 schema element

The top-level element of a Schematron schema.

The optional schemaVersion attribute gives the version of the schema. Its allowed values are not defined by this part of ISO/IEC 19757 and its use is implementation-dependent.

The optional `queryBinding` attribute provides the short name of the query language binding in use. If this attribute is specified, it is an error if it has a value that the current implementation does not support.

The `defaultPhase` attribute may be used to indicate the phase to use in the absence of explicit user-supplied information.

The `title` and p elements allow rich documentation.

The `icon`, `see` and `fpi` attributes allow rich interfaces and documentation.

5.4.14 `value-of` element

Finds or calculates values from the instance document to allow clearer assertions and diagnostics. The required `select` attribute is an expression evaluated in the current context that returns a string.

Variable references in the `select` attribute are resolved in the scope of the current schema, phase, pattern and rule.

An implementation which does not report natural-language assertions is not required to make use of this element.

5.5 Ancillary Elements and Attributes

5.5.1 `diagnostic` element

A natural-language message giving more specific details concerning a failed assertion, such as found *versus* expected values and repair hints.

NOTE Diagnostics in multiple languages may be supported by using a different `diagnostic` element for each language, with the appropriate `xml:lang` language attribute, and referencing all the unique identifiers of the `diagnostic` elements in the `diagnostics` attribute of the assertion. Annex G gives a simple example of a multi-lingual schema.

An implementation is not required to make use of this element.

5.5.2 `diagnostics` element

A section containing individual diagnostic elements.

An implementation is not required to make use of this element.

5.5.3 `dir` element

A section of natural-language text with a direction specified by the `value` attribute. The value `ltr` indicates left-to-right text; the value `rtl` indicates right-to-left text.

An implementation is not required to make use of this element.

5.5.4 `emph` element

A portion of text that should be rendered with some emphasis.

An implementation is not required to make use of this element.

5.5.5 `flag` attribute

The name of a Boolean flag variable. A flag is implicitly declared by an assertion or rule having a `flag` attribute with that name. The value of a flag becomes true when an assertion with that flag fails or a rule with that flag fires.

The purpose of flags is to convey state or severity information to a subsequent process.

An implementation is not required to make use of this attribute.

5.5.6 `fpi` attribute

A formal public identifier for the schema, phase or other element.

An implementation is not required to make use of this attribute.

5.5.7 `icon` attribute

The location of a graphics file containing some visible representation of the severity, significance or other grouping of the associated element.

An implementation is not required to make use of this attribute.

5.5.8 `p` element

A paragraph of natural language text containing maintainer and user information about the parent element. The schema can nominate paragraphs that should be rendered in a distinct way, keyed with the `class` attribute.

An implementation is not required to make use of this element.

5.5.9 `role` attribute

A name describing the function of the assertion or context node in the pattern. If the assertion has a `subject` attribute, then the role labels the arc between the context node and any nodes which match the path expression given by the `subject` attribute.

An implementation is not required to make use of this attribute.

5.5.10 `see` attribute

The URI of external information of interest to maintainers and users of the schema.

An implementation is not required to make use of this attribute.

5.5.11 `span` element

A portion of some paragraph that should be rendered in a distinct way, keyed with the `class` attribute.

An implementation is not required to make use of this element.

5.5.12 `subject` attribute

A path allowing more precise specification of nodes. The path expression is evaluated in the context of the context node of the current rule. If no subject attribute is specified, the current subject node may be used.

NOTE The `subject` attribute is required because the rule context may have been selected for reasons of convenience or performance, in association with the particular assertion tests. In such cases, the rule context may not be useful to identify the subject, and the nodes located by the `subject` attribute may be more useful. Similarly, it may not be possible to determine from an assertion test which nodes the assertion test has tested. In such a case, the nodes located by the `subject` attribute may be more useful.

An implementation is not required to make use of this element.

5.5.13 `title` element

A summary of the purpose or role of the schema or pattern, for the purpose of documentation or a rich user interface.

An implementation is not required to make use of this element.

6 Semantics

6.1 Validation Function

A general Schematron validator is a function returning "valid", "invalid" or "error". The function notionally performs two steps: transforming the schema into a minimal syntax, then testing the instance against the minimal syntax.

NOTE This part of ISO/IEC 19757 does not constrain other information provided by an implementation nor other uses of Schematron schemas. However, it is the intent of this part of ISO/IEC 19757 to support implementations to provide rich, specific diagnostics customised with values that assist in detecting and rectifying problems.

A Schematron validator is a function over the following:

— a query language binding

— a schema document

— an instance to be validated

— external XML documents addressed using information in the instance or schema

— a phase name, or `#ALL` if all patterns shall be active patterns, or `#DEFAULT` if the `defaultPhase` attribute on the schema element shall be used

— a list of name-value pairs, if the schema uses external variables.

6.2 Minimal Syntax

To simplify the specification of semantics later, the following transformation steps are first applied to a schema, resulting in a schema in the minimal syntax:

1. Resolve all inclusions by replacing the `include` element by the resource linked to.

2. Resolve all abstract patterns by replacing parameter references with actual parameter values in all enclosed attributes that contain queries.

3. Resolve all abstract rules in the schema by replacing the `extends` elements with the contents of the abstract rule identified.

4. Negate all `report` elements into `assert` elements.

5. Remove elements used for diagnostics and documentation.

The resulting minimal syntax is also a valid Schematron instance in the full syntax. The minimal syntax differs from the complex syntax by not containing the following XPaths:

— `//sch:include`

— `//sch:pattern[@abstract="true"]`

— `//sch:pattern[@is-a]`

— `//sch:rule[@abstract="true"]`

— `//sch:extends`

— `//sch:report`

— `//sch:diagnostics`

— `//sch:p`

— `//sch:title`

6.3 Schema Semantics

This clause gives the semantics of a good schema that has been resolved into the minimal form.

This predicate treats an instance as a set of contexts, a schema as a set of phases, a phase as a set of patterns, a pattern as a set of rules, and a rule as a set of assertions.

A document is valid in respect to a schema in a phase when the following predicate is satisfied:

```
∀ ( context, pattern, rule, assertion):
    ( context ∈ instance,
      active-phase ∈ schema,
      pattern ∈ active-phase,
      rule ∈ pattern,
      assertion ∈ rule :
        ( match ( rule, context, instance ),
          ∀ (previous-rule) :
            ( previous-rule ∈ pattern,
              position (previous-rule) < position( rule):
                  ¬ match(previous-rule, context, instance)
            )
        )
    )
    ⇔ assert ( assertion, context, instance) = true
```

where

— `position( rule )` is the position of the member in the set, a cardinal number,

— `match( rule, context, instance)` is a predicate supplied by the query language binding, and

— `assert( assertion, context, instance )` is a predicate supplied by the query language binding.

NOTE In natural language, a document is valid against a schema in a phase if: *There exists an instance, schema and active-phase combination where, for each context, pattern, rule and assertion (the context being a member of that instance, the active phase being a member of the schema, the pattern being a member of that active phase, the rule being a member of that pattern, the assertion being a member of that rule), the following is true: if the context of an instance matches the rule, and that context has not been matched by a previous rule in the same pattern, then the particular assertion evaluates to true when evaluated with the particular context and instance.*

6.4 Query Language Binding

A query language binding shall provide the following:

— The general query language used. A name token which identifies the query language. The data model.

— The rule context query language. The rule context scope.

— The assertion test, a function which returns a data value coercible into Boolean.

NOTE The following query language names are reserved without further definition. Implementations which use different query language bindings are encouraged to use one of these names if appropriate : `stx`, `xslt`, `xslt1.1`, `exslt`, `xslt2`, `xpath`, `xpath2`, `xquery`.

A schema language binding may also provide the following:

— The data models of the various query languages, the conversion between data models, and the treatment of information items: which information items are stripped or ignored, which information items are errors, and which information items are used.

— The `name` query language, a function which returns a data value coercible into a string.

— The `value-of` query language, a function which returns a data value coercible into a string.

— The `let` value query language, a function which returns a data value.

— The variable delimiter convention, a lexical convention such as a delimiter by which the use of a variable in a query expression shall be recognized.

— The abstract pattern parameter convention, a lexical convention such as a delimiter by which the parameters of abstract patterns inside query expressions shall be recognised.

The query language binding may also specify the element types in other namespaces which provide query-language-specific ancillary information or pragmatic hints.

The query language binding shall specify any whitespace processing required on queries.

The query language binding shall specify any restrictions to the value of name tokens.

A Schematron implementation which does not support the query language binding, specified in a schema with the `queryBinding` attribute, shall fail with an error.

6.5 Order and side-effects

The order in which elements are validated is implementation-dependent, without altering the validity of the instance.

The order in which patterns are used is implementation-dependent, without altering the validity of the instance.

The order in which assertions are tested is implementation-dependent, without altering the validity of the instance.

The only elements for which order is significant are the `rule` and `let` elements.

A `rule` element acts as an if-then-else statement within each pattern. An implementation may make order non-significant by converting rules context expressions to elaborated rule context expressions.

NOTE 1 The behaviour of the `rule` element allows constraints that would require a complex context expression to be factored into simpler expressions in different rules.

A `let` element may use lexically previous variables within the same rule or global variables.

NOTE 2 A wide variety of implementation strategies are therefore possible.

All queries shall act as pure functions. Queries shall not alter the instance in any way visible to other queries. This part of ISO/IEC 19757 does not specify any outcome augmentation of the instance being validated.

7 Conformance

7.1 Simple Conformance

A simple-conformance implementation shall be able to report for any XML document that its structure does not conform to that of a valid Schematron schema.

— A valid schema conforms to the constraints of Annex A, the normative ISO/IEC 19757-2 (RELAX NG Compact Syntax) schema of this part of ISO/IEC 19757.

A simple-conformance implementation shall be able to determine for any XML document and for any good schema whether the document is valid with respect to the schema.

NOTE 1 It is not a requirement of this part of ISO/IEC 19757 that a simple-conformance implementation shall be able to determine whether validation will terminate or whether the queries are feasible against some other schema for the instance. The ability to determine these depends on the query language used. Where the query language allows incorrectness to be established, implementations are encouraged to report this information as part of validation.

NOTE 2 It is not a requirement of this part of ISO/IEC 19757 that a simple-conformance implementation shall be able to generate validation reports in the Schematron Validation Report Language, defined in Annex D.

7.2 Full Conformance

A full-conformance implementation shall be able to determine for any XML document whether it is a correct schema.

— A valid schema conforms to the constraints of Annex A, the normative ISO/IEC 19757-2 (RELAX NG Compact Syntax) schema of this part of ISO/IEC 19757.

— A valid schema conforms to the constraints of Annex B, the normative ISO Schematron schema of this part of ISO/IEC 19757.

— A correct schema's attributes conform to the grammars specified by the query language binding in use.

— A correct schema has one definition only in scope for any variable name in any context.

— The values of the attributes `flag`, `id`, `name` and `prefix` are well-formed names in the version of XML being used.

A full-conformance implementation shall be able to determine for any XML document and for any good schema whether the document is valid with respect to the schema.

NOTE 1 It is not a requirement of this part of ISO/IEC 19757 that a full-conformance implementation shall be able to determine whether the validation will terminate or whether the queries are feasible against some other schema for the instance. The ability to determine these depends on the query language used. Where the query language allows incorrectness to be established, implementations are encouraged to report this information as part of validation.

NOTE 2 It is not a requirement of this part of ISO/IEC 19757 that a full-conformance implementation shall be able to generate validation reports in the Schematron Validation Report Language, defined in Annex D.

Annex A
(normative)

RELAX NG schema for Schematron

A correct Schematron schema shall be valid with respect to the following ISO/IEC 19757-2 (RELAX NG Compact Syntax) schema.

```
#       (c) International Organization for Standardization 2005.
#       Permission to copy in any form is granted for use with conforming
#       SGML systems and applications as defined in ISO 8879,
#       provided this notice is included in all copies.

default namespace sch = "http://purl.oclc.org/dsdl/schematron"

namespace local = ""

start = schema

# Element declarations
schema = element schema {
    attribute id { xsd:ID }?,
    rich,
    attribute schemaVersion { non-empty-string }?,
    attribute defaultPhase { xsd:IDREF }?,
    attribute queryBinding { non-empty-string }?,
    (foreign
     & inclusion*
     &  (title?,
       ns*,
       p*,
       let*,
       phase*,
       pattern+,
       p*,
       diagnostics?))
}

active = element active {
    attribute pattern { xsd:IDREF },
    (foreign & (text | dir | emph | span)*)
}

assert = element assert {
    attribute test { exprValue },
    attribute flag { flagValue }?,
    attribute id { xsd:ID }?,
    attribute diagnostics { xsd:IDREFS }?,
    rich,
    linkable,
    (foreign & (text | name | value-of | emph | dir | span)*)
}

diagnostic = element diagnostic {
    attribute id { xsd:ID },
    rich,
```

```
    (foreign & (text | value-of | emph | dir | span)*)
}

diagnostics = element diagnostics {
    foreign & inclusion* & diagnostic*
}

dir = element dir {
    attribute value { "ltr" | "rtl" }?,
    (foreign & text)
}

emph = element emph { text }

extends = element extends {
    attribute rule { xsd:IDREF },
    foreign-empty
}

let = element let {
    attribute name { nameValue },
    attribute value { string }
}

name = element name {
    attribute path { pathValue }?,
    foreign-empty
}

ns = element ns {
    attribute uri { uriValue },
    attribute prefix { nameValue },
    foreign-empty
}

p = element p {
    attribute id { xsd:ID }?,
    attribute class { classValue }?,
    attribute icon { uriValue }?,
    (foreign & (text | dir | emph | span)*)
}

param = element param {
    attribute name { nameValue },
    attribute value { non-empty-string }
}

pattern = element pattern {
    rich,
    (foreign & inclusion* &
     ( (attribute abstract { "true" }, attribute id { xsd:ID },
            title?, (p*, let*, rule*))
     |  (attribute abstract { "false" }?, attribute id { xsd:ID }?,
            title?, (p*, let*, rule*))
     | (attribute abstract { "false" }?, attribute is-a { xsd:IDREF },
            attribute id { xsd:ID }?, title?, (p*, param*))
     )
    )
}
```

```
phase = element phase {
    attribute id { xsd:ID },
    rich,
    (foreign & inclusion* & (p*, let*, active*))
}

report = element report {
    attribute test { exprValue },
    attribute flag { flagValue }?,
    attribute id { xsd:ID }?,
    attribute diagnostics { xsd:IDREFS }?,
    rich,
    linkable,
    (foreign &  (text | name | value-of | emph | dir | span)*)
}

rule = element rule {
    attribute flag { flagValue }?,
    rich,
    linkable,
    (foreign & inclusion*
     & ((attribute abstract { "true" },
            attribute id { xsd:ID }, let*, (assert | report | extends)+)
        | (attribute context { pathValue },
            attribute id { xsd:ID }?,
           attribute abstract { "false" }?,
           let*, (assert | report | extends)+)))
}

span = element span {
    attribute class { classValue },
    (foreign & text)
}

title = element title {
    (text | dir)*
}

value-of = element value-of {
    attribute select { pathValue },
    foreign-empty
}

# common declarations
inclusion = element include {
    attribute href { uriValue }
}

rich =
    attribute icon { uriValue }?,
    attribute see { uriValue }?,
    attribute fpi { fpiValue }?,
    attribute xml:lang { langValue }?,
    attribute xml:space { "preserve" | "default" }?

linkable =
    attribute role { roleValue }?,
    attribute subject { pathValue }?
```

```
foreign =
    foreign-attributes, foreign-element*

foreign-empty =
    foreign-attributes

foreign-attributes =
    attribute * - (local:* | xml:*) { text }*

foreign-element =  element * - sch:*  {
    (attribute * { text }
     | foreign-element
     | schema
     | text)*
}

# Data types

uriValue = xsd:anyURI
pathValue = string
exprValue = string
fpiValue = string
langValue = xsd:language
roleValue = string
flagValue = string
nameValue = string  # In the default query language binding, xsd:NCNAME
classValue = string

non-empty-string = xsd:token { minLength = "1" }
```

Annex B
(normative)

Schematron Schema for Additional Constraints

A correct Schematron schema shall be valid with respect to the following Schematron schema.

```
<!--
       (c) International Organization for Standardization 2005.
       Permission to copy in any form is granted for use with conforming
       SGML systems and applications as defined in ISO 8879,
       provided this notice is included in all copies.
 -->

<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron"
     xml:lang="en" >
    <sch:title>Schema for Additional Constraints in Schematron</sch:title>
    <sch:ns prefix="sch" uri="http://purl.oclc.org/dsdl/schematron" />

    <sch:p>This schema supplies some constraints in addition to those
    given in the ISO/IEC 19757-2 (RELAX NG Compact Syntax) Schema for Schematron.
    </sch:p>

    <sch:pattern>
        <sch:rule context="sch:active">
            <sch:assert
                test="//sch:pattern[@id=current()/@pattern]">
            The pattern attribute of the active element shall match the
            id attribute of a pattern.
            </sch:assert>
        </sch:rule>

        <sch:rule context="sch:pattern[@is-a]">
            <sch:assert
                test="//sch:pattern[@abstract='true'][@id=current()/@is-a]">
            The is-a attribute of a pattern element shall match
            the id attribute of an abstract pattern.
            </sch:assert>
        </sch:rule>

        <sch:rule context="sch:extends">
            <sch:assert
                test="//sch:rule[@abstract='true'][@id=current()/@rule]">
            The rule attribute of an extends element shall match
            the id attribute of an abstract rule.
            </sch:assert>
        </sch:rule>


        <sch:rule context="sch:let">
            <sch:assert
                test = "not(//sch:pattern
                    [@abstract='true']/sch:param[@name=current()/@name])">
            A variable name and an abstract pattern parameter should not
            use the same name.
```

```
            </sch:assert>
        </sch:rule>

    </sch:pattern>
</sch:schema>
```

Annex C
(normative)

Default Query Language Binding

A Schematron schema with no language binding or a `queryBinding` attribute with the value `xslt`, in any mix of upper and lower case letters, shall use the following binding:

— The query language used is the extended version of XPath specified in XSLT. Consequently, the data model used is the data model of those specifications.

— The rule context is interpreted according to the Production 1 of XSLT. The rule context may be the root node, elements, attributes, comments and processing instructions.

— The assertion test is interpreted according to Production 14 of XPath, as returning a Boolean value.

— The name query is interpreted according to Production 14 of XPath, as returning a string value. Typically, the `select` attribute contains an expression returning an element node: the name query takes the local or prefixed name of the node, not its value.

— The value-of query is interpreted according to Production 14 of XPath, as returning a string value.

— The let value is interpreted according to Production 14 of XPath, as returning a string value.

— The notation for signifying the use of parameter of an abstract pattern is to prefix the name token with the **$** character. This is a character not found as a delimiter in URLs or XPaths. The **$** character not followed by the name of an in-scope parameter shall not be treated as a parameter name delimiter. Such a character may subsequently be used as a delimiter for a variable name or as a literal character.

— A Schematron let expression is treated as an XSLT variable. The XSLT $ delimiter signifies the use of a variables in an context expression, assertion test, name query, value-of query or let expression. The **$** character not followed by the name of an in-scope variable shall be treated as a literal character.

The XSLT `key` element may be used, in the XSLT namespace, before the pattern elements.

The attributes `id`, `name` and `prefix` should follow the rules for non-colonized names for the version of XML used by the document.

Annex D
(informative)

Schematron Validation Report Language

D.1 Description

The Schematron Validation Report Language (SVRL) is a simple XML language which may be used for reporting the results of Schematron validation and for conformance suites.

The order of elements in an SVRL is implementation-dependent; different implementations may generate the same elements in a different order.

All elements shown in the grammar for Simple Validation Report Language are qualified with the namespace URI:

`http://purl.oclc.org/dsdl/svrl`

In subsequent schemas, the prefix `svrl` is taken as bound to the Simple Validation Report Language namespace URI for exposition purposes. The prefix `svrl` is not reserved or required by this part of ISO/IEC 19757.

D.2 RELAX NG Compact Syntax Schema

```
<!--
       (c) International Organization for Standardization 2005.
       Permission to copy in any form is granted for use with conforming
       SGML systems and applications as defined in ISO 8879,
       provided this notice is included in all copies.
 -->
default namespace = "http://purl.oclc.org/dsdl/svrl"

schematron-output = element schematron-output {
   attribute title { text }?,
   attribute phase { xsd:NMTOKEN }?,
   attribute schemaVersion { text }?,
    human-text*,
    ns-prefix-in-attribute-values*,
    (active-pattern,
     (fired-rule, (failed-assert | successful-report)*)+)+
  }

# only namespaces from sch:ns need to be reported
ns-prefix-in-attribute-values = element ns-prefix-in-attribute-values {
    attribute prefix { xsd:NMTOKEN },
    attribute uri { text },
    empty
  }

# only active patterns are reported
active-pattern = element active-pattern {
   attribute id { xsd:ID }?,
  attribute name { text }?,
  attribute role { xsd:NMTOKEN }?,
   empty
  }

# only rules that are fired are reported,
```

```
fired-rule =
  element fired-rule {
   attribute id { xsd:ID }?,
   attribute context { text },
   attribute role { xsd:NMTOKEN }?,
   attribute flag { xsd:NMTOKEN }?,
 empty
  }

# only references are reported, not the diagnostic
diagnostic-reference = element diagnostic-reference {
   attribute diagnostic { xsd:NMTOKEN },
   human-text
  }
# only failed assertions are reported
failed-assert = element failed-assert {
    attlist.assert-and-report,
    diagnostic-reference*,
    human-text
  }

# only successful asserts are reported
successful-report = element successful-report {
    attlist.assert-and-report,
    diagnostic-reference*,
    human-text
  }

human-text = element text { text }

attlist.assert-and-report = attribute id { xsd:ID }?,
  attribute location { text },
  attribute test { text },
  attribute role { xsd:NMTOKEN }?,
  attribute flag { xsd:NMTOKEN }?

start = schematron-output
```

D.3 Schematron Schema

The corresponding Schematron schema is:

```
<!--
       (c) International Organization for Standardization 2005.
       Permission to copy in any form is granted for use with conforming
       SGML systems and applications as defined in ISO 8879,
       provided this notice is included in all copies.
 -->

<sch:schema
    xmlns:sch="http://purl.oclc.org/dsdl/schematron"
    xml:lang="en" >

    <sch:title>Schema for Schematron Validation Report Language</sch:title>
    <sch:ns prefix="svrl" uri="http://purl.oclc.org/dsdl/svrl" />

    <sch:p>The Schematron Validation Report Language is a simple language
        for implementations to use to compare their conformance. It is
        basically a list of all the assertions that fail when validating
        a document, in any order, together with other information such
```

```
    as which rules fire.
</sch:p>
<sch:p>This schema can be used to validate SVRL documents, and provides
    examples of the use of abstract rules and abstract patterns.</sch:p>

<sch:pattern>
   <sch:title>Elements</sch:title>

   <!--Abstract Rules -->
   <sch:rule abstract="true" id="second-level">
     <sch:assert test="../svrl:schematron-output">
         The <sch:name/> element is a child of schematron-output.
     </sch:assert>
   </sch:rule>

   <sch:rule abstract="true" id="childless">
     <sch:assert test="count(*)=0">
         The <sch:name/> element should not contain any elements.
     </sch:assert>
   </sch:rule>

   <sch:rule abstract="true" id="empty">
     <sch:extends rule="childless" />
     <sch:assert test="string-length(space-normalize(.)) = 0">
         The <sch:name/> element should be empty.
     </sch:assert>
   </sch:rule>

   <!-- Rules-->
   <sch:rule context="svrl:schematron-output">
     <sch:assert test="not(../*)">
         The <sch:name/> element is the root element.
     </sch:assert>
     <sch:assert
         test="count(svrl:text)+ count(svrl:ns-prefix-in-attribute-values)+
             count(svrl:active-pattern)+
             count(svrl:fired-rule) + count(svrl:failed-assert)+
             count(svrl:successful-report) = count(*)">
         <sch:name/> may only contain the following elements:
         text, ns-prefix-in-attribute-values, active-pattern, fired-rule, failed-assert
         and successful-report.
     </sch:assert>
     <sch:assert test="svrl:active-pattern">
         <sch:name/> should have at least one active pattern.
     </sch:assert>
   </sch:rule>

   <sch:rule context="svrl:text">
       <sch:extends rule="childless" />
   </sch:rule>

   <sch:rule context="svrl:diagnostic-reference">
     <sch:extends rule="childless" />
     <sch:assert test="string-length(@diagnostic) &gt; 0">
         <sch:name/> should have a diagnostic attribute,
         giving the id of the diagnostic.
     </sch:assert>
   </sch:rule>

   <sch:rule context="svrl:ns-prefix-in-attribute-values">
     <sch:extends rule="second-level" />
```

```
        <sch:extends rule="empty" />
        <sch:assert
            test="following-sibling::svrl:active-pattern
                  or following-sibling::svrl:ns-prefix-in-attribute-value">
            A <sch:name/> comes before an active-pattern or another
            ns-prefix-in-attribute-values element.
        </sch:assert>
    </sch:rule>

    <sch:rule context="svrl:active-pattern">
        <sch:extends rule="second-level" />
        <sch:extends rule="empty" />
    </sch:rule>

    <sch:rule context="svrl:fired-rule">
        <sch:extends rule="second-level" />
        <sch:extends rule="empty" />
        <sch:assert
            test="preceding-sibling::active-pattern |
                preceding-sibling::svrl:fired-rule |
                preceding-sibling::svrl:failed-assert |
                preceding-sibling::svrl:successful-report">
            A <sch:name/> comes after an active-pattern, an empty
            fired-rule, a failed-assert or a successful report.
        </sch:assert>
        <sch:assert test="string-length(@context) &gt; 0">
            The <sch:name/> element should have a context attribute
            giving the current context, in simple XPath format.
        </sch:assert>
    </sch:rule>

    <sch:rule context="svrl:failed-assert | svrl:successful-report">
        <sch:extends rule="second-level" />
        <sch:assert
            test="count(svrl:diagnostic-reference) + count(svrl:text) = count(*)">
            The <sch:name/> element should only contain a text element
            and diagnostic reference elements.
        </sch:assert>
        <sch:assert test="count(svrl:text) = 1">
            The <sch:name/> element should only contain a text element.
        </sch:assert>
        <sch:assert test="preceding-sibling::svrl:fired-rule |
            preceding-sibling::svrl:failed-assert |
            preceding-sibling::svrl:successful-report">
            A <sch:name/> comes after a fired-rule, a failed-assert or a
            successful-report.
        </sch:assert>
    </sch:rule>

    <!-- Catch-all rule-->
    <sch:rule context="*">
        <sch:report test="true()">
            An unknown <sch:name/> element has been used.
        </sch:report>
    </sch:rule>
</sch:pattern>

<sch:pattern>
    <sch:title>Unique Ids</sch:title>
```

```
        <sch:rule context="*[@id]">
            <sch:assert test="not(preceding::*[@id=current()/@id][1]">
                Id attributes should be unique in a document.
            </sch:assert>
        </sch:rule>
    </sch:pattern>

    <sch:pattern abstract="true" id="requiredAttribute">
        <sch:title>Required Attributes</sch:title>

        <sch:rule context=" $context ">
            <sch:assert test="string-length( $attribute ) &gt; 0">
                The <sch:name/> element should have a
                <sch:value-of select="$attribute /name()" /> attribute.
            </sch:assert>
        </sch:rule>
    </sch:pattern>

    <sch:pattern is-a="requiredAttribute">
        <sch:param name="context" value="svrl:diagnostic-reference" />
        <sch:param name="attribute" value="@diagnostic" />
    </sch:pattern>

    <sch:pattern is-a="requiredAttribute">
        <sch:param name="context" value="svrl:failed-assert or svrl:successful-report" />
        <sch:param name="attribute" value="@location" />
    </sch:pattern>

    <sch:pattern is-a="requiredAttribute">
        <sch:param name="context" value="svrl:failed-assert or svrl:successful-report" />
        <sch:param name="attribute" value="@test" />
    </sch:pattern>

    <sch:pattern is-a="requiredAttribute">
        <sch:param name="context" value="svrl:ns-prefix-in-attribute-values" />
        <sch:param name="attribute" value="@uri" />
    </sch:pattern>

    <sch:pattern is-a="requiredAttribute">
        <sch:param name="context" value="svrl:ns-prefix-in-attribute-values" />
        <sch:param name="attribute" value="@prefix" />
    </sch:pattern>

</sch:schema>
```

Annex E
(informative)

Design Requirements

Motivating design requirements for the schema language with the default query language binding include:

1. Represent abstract patterns such as the *head+body* pattern.
2. Support progressive validation.
3. Include assertions or abstract rules from an external file.
4. Support a one-to-one mapping between the natural-language statements and artificial-language statements.
5. Support the generation and labelling of arcs between information items.

Motivating design requirements for the schema language with the default query language binding do not include:

— Simple specification of constraints that are simply expressed by grammar-based validation, such as ISO/IEC 19757-2 (RELAX NG) schemas.

— Replacement of any other standard schema language for XML.

— Single-pass or streamable implementation.

As well, certain outcomes are out-of-scope for this part of ISO/IEC 19757:

— Specification of a type system.

— Generation of links between multiple occurrences of some datum across multiple documents for the purpose of consistency-checking.

Annex F
(normative)

Use of Schematron as a Vocabulary

The following Schematron element types may be used as vocabulary elements by other standards and schemas. The semantics of element types used externally shall follow this part of ISO/IEC 19757.

— `schema`

— `pattern`

— `rule`

— `assert`

— `report`

These elements should use the standard Schematron namespace specified in 5.2.

When Schematron elements other than the `schema` element are used as vocabulary elements by other standards and schemas, the other standard or schema should specify mechanisms for defining information otherwise supplied by Schematron elements, such as `ns` or `let`.

Annex G
(informative)

Use of Schematron for Multi-Lingual Schemas

The following Schematron schema shows how multiple languages may be supported.

```
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron"
     xml:lang="en" >
    <sch:title>Example of Multi-Lingual Schema</sch:title>

    <sch:pattern>
        <sch:rule context="dog">
            <sch:assert test="bone" diagnostics="d1 d2">
            A dog should have a bone.
            </sch:assert>
        </sch:rule>
    </sch:pattern>

    <sch:diagnostics>
        <sch:diagnostic id="d1" xml:lang="en">
        A dog should have a bone.
        </sch:diagnostic>
        <sch:diagnostic id="d2" xml:lang="de">
        Ein Hund sollte ein Bein haben.
        </sch:diagnostic>
    </sch:diagnostics>

</sch:schema>
```

Bibliography

[1] *Resource Description for Schematron (web page)*, Rick Jelliffe, Computing Centre, Academia Sinica, Taipei, http://purl.oclc.org/dsdl/schematron

[2] *XML Constraint Specification Language*, José Carlos Leite Ramalho, Department of Informatics, School of Engineering, University of Minho, http://www.di.uminho.pt/~jcr/PROJS/xcsl-www/

ICS 35.240.30

Price based on 30 pages