STMML. A MARKUP LANGUAGE FOR SCIENTIFIC, TECHNICAL AND MEDICAL PUBLISHING

Peter Murray-Rust [*] [a] ***and Henry S. Rzepa*** [b]

[a] *Unilever Centre for Molecular Informatics, Department of Chemistry, University of Cambridge, Lensfield Road, Cambridge. CB2 1EW.* [b]*Department of Chemistry, Imperial College, London, SW7 2AY, England.*

Abstract:

STMML is an XML-based markup language covering many generic aspects of scientific information. It has been developed as a re-usable core for more specific markup languages. It supports data structures, data types, metadata, scientific units and some basic components of scientific narrative. The central means of adding semantic information is through dictionaries. The specification is through an XML Schema which can be used to validate STMML documents or fragments. Many examples of the language are given.

Keywords: Extensible-markup-language (XML), Chemical-markup-language (CML), Extensible-stylesheet-language-transformations (XSLT), XMLSchema, Semantic-Web, Scientific-Technical-Medical (STM), Scientific Units, Metadata, Data Types, Data Structures, Dictionaries, Terminology.

1. Introduction and history

Currently most Scientific-Technical-Medical (STM) publications are printed on paper and aimed at human readers. Where electronic versions occur, most are created as "e-paper", i.e. for ultimate viewing and interpretation by humans. The technologies for this, primarily HTML and PDF along with proprietary document formats, do not support machine processing. For example machines cannot easily extract meaningful data from these documents; they are normally limited to textual searches without a knowledge of the context in which the text occurs.

We have reviewed the need for machine-processing in an earlier article (Murray-Rust & Rzepa, 2002a). In this we argued that XML (eXtensible Markup Language) is a radical development in STM publishing, and that a "datument" (a portmanteau neologism of document+data) could be created from a number of carefully constructed information components. Each basic component is supported by a markup language (ML). For example we have developed a language termed Chemical Markup Language or CML (Murray-Rust & Rzepa, 1999; Murray-Rust & Rzepa, 2001) to manage chemical information, especially molecules. Similar markup languages (e.g. CellML, MathML, Geographical Markup Language, Reaction XML, and many more) are being developed in many STM domains (Cover, 2002). These languages are developed by a core of collaborating specialists and provide specific vocabularies for their disciplines. Normally there is a need to create specific software tools (*e.g.* for graphical input and display of molecules, maps, mice, etc). Thus considerable effort is required and such languages need careful design and management. A likely outcome is that these will become 'standard' (at least *de facto*) and will form the core method of deposition of specialist data with publishers and archivists.

However there is a large communality in the core and infrastructure of these languages, such as numbers, scientific units, metadata, simple data structures, etc. Building the required tools for this is not trivial and there is a great danger in "reinventing the wheel". Chemistry requires much of this infrastructure and we have had to implement software systems for managing basic scientific data. Our efforts have resulted in a generic approach which can be valuable as an existing infrastructure for most STM domains to build on. In this article we offer this generic core for use by the STM community as a starting point or re-usable component for their own domain. An additional benefit would be that less software needs to be developed and this may make it easier for publishers to implement systems and accept markup-language-based datuments.

In this article we assume some familiarity with basic XML (World Wide Web Consortium, n.d.) and its terminology such as *element* and *attribute*. and the syntax for these:

```
<anElement attribute1="value1" attribute2="value2">
<childElement>Some text content</childElement>
<anEmptyElement attribute3="value3"/>
</anElement>
```

We shall review in more detail the concepts behind XML namespaces and the recent XML Schema specification. Other XML protocols and tools which are likely to occur in generic XML applications are:

- XSL. A generic protocol for transforming (XSLT) and formatting (XSL-FO) documents.
- RDF. A protocol for describing metadata
- Dublin Core. A simple, extensible approach to metadata content
- XML Query. A W3C activity to define query standards for XML documents.
- XLink. A specification for hypermedia (links)
- XForms. A generic development of HTML forms to allow submission of data and structured queries in XML (and included as a modular component of XHTML 1.1).

2. Motivation for STMML

When designing CML 1.0 (Murray-Rust & Rzepa, 1999; Murray-Rust & Rzepa, 2001; Gkoutos, Murray-Rust, Rzepa & Wright (2001b) we found that many components with a "chemical content" did not require chemical concepts for their implementation. Thus the "melting point of a substance" is a numeric quantity, with error estimates and scientific units, and a name (or term) representing the concept. There is nothing specifically chemical in the abstraction of the information; the same technology could represent "rectal temperature of patient". To add the semantics (e.g. "a melting point is the temperature at which a solid turns to liquid") requires a generic mechanism - in our case through a specialised dictionary. We therefore had to include generic STM concepts in CML when it was first developed (Murray-Rust, Leach & Rzepa, 1995). XML technology has now developed to allow combination of modular components and we have therefore modularised CML 1.0 into two parts; the purely chemical core components *e.g.* representation of molecules, which becomes CMLCore (Murray-Rust & Rzepa, 2002b) and the general STM infrastructure which here is described as STMML 1.0. Since STMML is general, we offer it to the STM community as a tool for any scientific domain.

Because of the ubiquity and fundamental nature of the problem we are addressing - the description of general STM information - the community is likely to develop several (possibly independent) approaches. Although there are several groups who are addressing this problem through markup languages, we are unaware of any formally published and tested specifications.

There are also languages which address general modelling of systems, some impressive ones being:

- CellML (Cell Markup Language, n.d.) for cells and their components (but the architecture is sufficiently abstract it could be extended to other domains).
- Systems Biology Markup Language (Systems Biology Markup Language, n.d.) which again has a considerable degree of abstraction
- FieldML (Field Markup Language, n.d.) which supports coordinate information.

These cover some of the same concepts as STMML (e.g. units, metadata). They are generally at a higher level of abstraction than provided by STMML. Thus CellML has elements such as `group`, `connection`, etc. Like STMML they use a modular approach and will re-use other specifications rather than re-inventing them.

We believe that the use of markup languages will be dynamic and evolve as they fill needs and can be implemented. STMML has been deliberately kept simple and concrete. It has been proven to work in conjunction with CML and is particularly aimed at publishing heterogeneous data. It is deliberately Open, *i.e.* non-proprietary and aimed *inter alia* at supporting publication in a context where authors are not necessarily initially familiar with the concepts of markup.

3. Design

The W3C, when designing XML, produced a set of design criteria which were extremely useful for the developer community. We followed the same principle for CML and do so here for STMML. The primary design considerations are:

- STMML must be simple enough for authors to understand it
- STMML must abstract enough to allow most STM domains to use it.
- STMML must be extensible; its development may have an OpenSource-like philosophy
- STMML should avoid "reinventing the wheel"; it should interoperate with existing markup languages
- STMML should have a modular design
- It should be possible to implement STMML applications using standard XML tools

From an analysis of a number of STM articles we have derived a set of generic information components, independent of the domain. Some were explicit such as tables while others were implicit such as points on graphs, references to unpublished or supplemental data, etc. Most are amenable to representation in XML and this would allow *all* data to be included in the published datument. The reader could have access to the whole background of the publication; hopefully enough to reproduce the experiment or to analyse it further with a different set of software tools (Rzepa & Murray-Rust, 2001).

We recognise that many disciplines create very large and complex data sets which are already managed by standards (formal or *de facto* agreements) and for which authoring and processing software exists. XML has no special support for very large datasets. An example is image data from crystallographic detectors where the IUCr CIF community has created a protocol for interchange (Hall, Allen & Brown, 1991). In other cases protocols such as JCAMP for spectra have their own syntax and some variants are not human-readable. This seriously reduces the interoperability and puts a large burden on the community to write software, without the re-use that XML provides. Recognising this, some domains such as e.g. the genomic community are increasingly using XML for nucleic and protein sequence information. Even if XML is not used for the transport of the data itself, it will be valuable for the metadata, including details on what software should be used to process the data.

4. Current design

Our current analysis is unlikely to be universally applicable. It is only through the discipline of marking up real publications that further useful abstractions occur. The list below is very general; indeed many of the components will occur in business and e-commerce. At present we see a wide need for:

- **Structuring the document for publication**. This is a "solved" problem in that publishers have been using markup languages such as SGML and now XML for many years. MLs include ISO12083 and DOCBOOK (Docbook, n.d.) and STMML is designed to be used with any such structural ML. Thus a vocabulary for references/citations, abstracts, authorship, etc. can be taken directly from the ML of choice.

- **Text**. Again this is "solved"; for most purposes we find that XHTML is powerful enough to manage textual content. Its limitations are not normally in the content but the presentation (e.g. page layout, special characters and glyphs, etc.). We see XML publishing tools such as XSL-FO supporting this (Formatting Objects, n.d.). In any case, for machine processing the content rather than presentation is required. Where sophisticated presentation is required to convey meaning, it is often an indication that the semantics are poorly developed.

- **Raster/bitmap Images**. Although imperfect, the GIF, PNG and JPG standards are useful means of transporting images. Normally there is no microstructure in the image. Although the SHAPE element in XHTML can refer to areas in an image it may not have a rich enough structure to *e.g.* create a hierarchy or overlapping views. The increasing tendency to associate metadata with images is a welcome step towards adding more meaning to the otherwise presentational emphasis.

- **Vector graphics**. The increasing adoption of e.g. Scalable Vector Graphics (SVG, n.d.) is an excellent development and we recommend that graphical information is encoded this way rather than as raster images whenever possible. SVG is powerful enough to encapsulate non-graphic data, so that STMML elements can be contained by, or bound to SVG primitives such as visual areas.

- **Tables**. XHTML tables are used primarily for presentation and have no mechanism for enforcing data types. Moreover they allow row and column spanning, which although useful for presentation leads to very complex structures, sometimes a mixture of trees and tables; sometimes purely a visual layout. OASIS has developed the CALS table model for XML (Table models, n.d.) and there is also a growing interest in converting the output of statistical programs and spreadsheets to XML form. This is a fluid area and we have developed a simple `<table>` element for *STMML to support "tables for publication"*. Its semantics are strict, requiring that columns enforce datatyping on elements.

- **Graphs**. There is a very great need for a language to support graphical plots of data. This is a complex problem as it depends on dimensionality, continuity, dependent and independent variables, etc. It overlaps significantly with tables - indeed many tables are usefully plottable. We are aware of PlotML (PlotML, n.d.) which manages many of the basic concepts. For complex applications it will be necessary identify and process components of the graph (regions, contours, etc.). In particular a plotting language for STM data should support datatypes and scientific units.

- **Mathematics**. MathML (World Wide Web Consortium, n.d.) has been developed for publishing mathematics in an STM context and we fully support its use.

- **Metadata**. "Metadata" is a broad term and can be used in several general ways:

    - **Navigational**. This is primarily to locate documents and data;

    - **Constraining**. This constrains the value and the behaviour of the data;

    - **Supplemental**. This adds information to the data (e.g. a chemical element symbol could be used to locate and add the IUPAC atomic weight);

    - **Vocabulary**. This supplies "meaning" to the data.

  The W3C has developed RDF (World-Wide Web Consortium, n.d.) as an approach to metadata; it is not yet widely used but if/when it does then we expect that STM applications will make wide use of it. RDF supplies an infrastructure, and there is an additional need for representing the values, which is increasing supplied by Dublin Core. Several collections of metadata content exist and we expect generic concepts (people, organisations, places, etc.) to be supplied in this way.

- **Dictionaries**. Although part of "metadata", *STMML gives dictionaries a special place and supplies components to support the creation of STM dictionaries*. STMML usually separates the markup of a concept from its semantics, which is done through a pointer to a dictionary entry.

- **Abstract datatypes**. The W3C has identified 43 primitive datatypes in common use and included these in XML Schema (World Wide Web Consortium, n.d.). All Schema implementations must provide tools for validating these types, such as *e.g.* numbers, URLs, dates, XML components. In addition users can define lexical patterns (*e.g.* for telephone numbers). *STMML uses the W3C set of datatypes* (World Wide Web Consortium, n.d.) so that complete validation can be achieved with standard tools. Complex datatypes can also be built with XML Schema; obvious examples are "address", "person", etc. which are composed from more primitive components. It is likely that implementations of generic complex datatypes will become increasingly available.

- **Scientific units**. This is a complex area whose markup has been explored (Olken & McCarthy, 2000; Dragoset, Taylor & McLay, 2002). It is likely that several groups will develop a variety of approaches. *STMML supports units through dictionaries* so that is easy for a community to create new units. Sufficient information is held to manage dimension analysis, and to support the "dimensionless" unit in a richer manner.

- **Domain-specific components**. There will be many domains that require their own information components. In some cases these can be built from simpler ones using XML Schema (or similar) methods; often they will not need specialist software. However some disciplines require specialist software for creating, displaying and processing information components. Early examples are chemical markup language and geography (GML, 2001) Although both are represented by XML Schemas (World Wide Web Consortium, n.d.) they absolutely require additional software to make the semantics explicit.

- **General complex components**. In contrast many systems may be describable as an aggregation of their component parts, so that a general ontological system might be adequate to describe them. The "Gene Ontology" project (Gene Ontology Consortium, n.d.) which despite its name actually addresses much of biology, allows partitive ("ispartOf") and generic ("isASortOf") concepts. New concepts can be built from existing ones and this may be satisfactory for many purposes.
  As a simple means for coarse structure, *STMML provides an `<object>` element*. This implies no semantics, but is a useful container and which will normally be linked to a dictionary entry providing more information.

- **Relationships**. Many systems require relations between components. RDF (World Wide Web Consortium) is a sufficiently powerful concept to represent these as it consists of "triples" which link one object (an XML element) to another associated with a property (itself an XML element and potentially part of one or more other triples).

-
- `A...(property)...>B`
  This can represent a complex system of relationships but has no generic mechanism for adding program functionality to the properties. STM authors will need to choose between generic mechanism and developing support for specific relations in their own MLs. *STMML introduces a generic element `<link>` to support mappings.*

- **Processes**. Many STM publications describe processes (recipes, measurements, etc.) often with discrete steps and a time component. *STMML therefore provides a `<action>` element* which allows representation of discrete steps in a process.

4.1 Namespaces

Namespaces are not formally part of XML V1.0 and were introduced in 1998 (World Wide Web Consortium, n.d.). They have since become universal in all XML-based languages and are supported by XML-tools and APIs such as DOM and SAX. The namespace concept is simple: *every element and every attribute in a document can have a unique namespace associated with it*. Uniqueness is created through a namespace URI, normally containing the domain name of the "owner" of the vocabulary, though other systems such as the ISBN number could be used. Despite its URI-based syntax the namespaceURI does *not* have to exist physically and there is no requirement to connect to networks. Note that the namespace prefix is arbitrary; although "xsl" is used below, any unique prefix would suffice.

We have created an example which uses three namespaces. Two represent content (XHTML and CML) and one supports infrastructure (XSLT). There is no limit on the number or purpose of namespaces in a document; we have published an article which uses six (Murray-Rust, Rzepa and Wright, 2001).

```
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:h="http://www.w3.org/1999/xhtml"
  xmlns="http://www.xml-cml.org/schema/cml2/core"
  version="1.0">
  <xsl:template match="molecule">
    <h:h2>Entry ID:<xsl:value-of select="@id"/></h:h2>
  </xsl:template>
</xsl:stylesheet>
```

The three namespaces above are:

- `http://www.w3.org/1999/XSL/Transform`. This URI defines the XSLT V1.0 (XSL Transformation) namespace and (in this example) associates all the elements prefixed by `xsl:` with it. This is an important example, because XSLT continues to evolve (requirements for XSLT2.0 have been collected). The namespace signals to software systems what version, and therefore what functionality is required.

- `http://www.w3.org/1999/xhtml`. This denotes the namespace for XHTML, and elements prefixed (in this case) with `h:` will belong to that namespace.
- `http://www.xml-cml.org/schema/cml2/core`. A single *default namespace* without prefixes is allowed in any XML document. We illustrate this with CMLCore Schema. CML 2.0 specifies these core components, along with other proposed and individually namespaced components relating to reactions, spectra etc. Again, note the value in distinguishing versions: The DTD (Document Type Definition)-based version of CML (version 1.0) has a different syntax and a different namespace (http://www.xml-cml.org/dtd/cml_1_0_1.dtd).

There is no requirement to have a default namespace and if elements are always prefixed it may make it easy to merge document components. The namespace allows the processing systems to decide exactly what version of the software to use. At present there is no universal method for automatically associating actual software (such as through the use of *e.g.* java class libraries) with a namespace URI but this will certainly become common.

The example above represents part of an XSL stylesheet and shows how information can be transformed from one namespace (CMLCore Schema) to another (XHTML). In the next example we show how different namespaces can be combined to create a composite document:

```
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:cml="http://www.xml-cml.org/schema/cml2/core">
  <p>We can supply the following set of molecules:</p>
  <ul>
  <li>
    <cml:molecule id="p1" title="phosphine">
        <cml:atomArray>
            <cml:atom elementType="P" hydrogenCount="3"/>
        </cml:atomArray>
    </cml:molecule>
  </li>
  <li><cml:molecule id="p2" title="penguinone"/></li>
  </ul>
</html>
```

Here the default namespace is XHTML, and elements from another namespace (CMLCore) are contained in HTML elements (`li`). With XML Schemas and XSLT the components can be separately validated (see below). With DTDs this would be messy and the document probably could not be DTD-validated (the content model for `li` cannot contain CML elements). Namespaces allow us to regard fragments of the document as architecturally independent, and these fragments can be processed separately.

XPath is the syntax for addressing nodes within XML documents and is used by protocols such as XSLT and XQuery (under development). A namespace prefix in the XPath expression is arbitrary and

```
<foo xmlns:bar="http://www.xml-cml.org/schema/cml2/core">
<xsl:variable name="mols" select="bar:molecule"/>
</foo>
```

will retrieve `molecule`s from any elements mapped to the CML namespace URI. They do not have to use the prefix `cml`. Note also that other elements which happen to have the elementName `cml:molecule` mapped to a different namespace will not be selected, as the next example shows.

```
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:c="http://www.xml-cml.org/schema/cml2/core"
  xmlns:cml="http://www.penguinchem.com/schema">
  <p>We can supply the following set of molecules:</p>
  <ul>
  <li>
    <c:molecule id="p1" title="phosphine">
      <c:atomArray>
```

```
        <c:atom elementType="P" hydrogenCount="3"/>
      </c:atomArray>
     </c:molecule>
  </li>
<!-- not a CML molecule -->
  <li><cml:molecule foo="p2" bar="penguinone"/></li>
  </ul>
</html>
```

Even though the second molecule has the (rather misleading) prefix `cml` it does not belong to the CMLCore namespace and will not be retrieved by the XPath expression. This provides a mechanism (if required) for mixing different versions of a markup language (though this should always be done with great caution).

4.2 Modularization

If information can be packaged into modules the benefits are very considerable:

- The need for understanding the context often disappears. This is a major gain in re-using information and writing the software. If `cml:molecule` has the same semantics regardless of whether it is contained within a database, a primary publication, a regulatory submission, associated with a spectrum or any other modular component, then a single set of software modules will fit all circumstances.

- Complex documents can be created with components produced in parallel. If the molecular information does not have semantic collisions with (say) clinical trial data, the chemistry in a submission can be compiled independently.

- Each namespace can be viewed as consisting of a set of re-usable components with associated software. Thus SVG content can be associated with "an SVG viewer"; it is not necessary to know the details of the implementation. Similarly we have created "CML viewers" which perform independently of where they are invoked.

- Each domain can concentrate on just those components which are key to its ontology. In early versions of CML viewers, we wrote text and image handlers (because XHTML and SVG were not available). Now we can concentrate on just the chemical aspects.

- Software systems will often consist of a generic XML framework with specialist software modules (currently browser "plugins", but the technology is always changing).

- Complex documents can be analysed as a set of non-interacting components, making searches much easier to implement. If all molecules on the Web were to be namespaced as *e.g.* belonging to CML, then it would be conceptually easy to retrieve ALL such components. There would remain a need for indexes for most searches to speed up performance, but the indexing tools need only look for namespaced molecules (Gkoutos, Leach and Rzepa, 2002).

- Some complex concepts are naturally built up of separate components ("aggregation"). Thus an entry in a chemical catalog could include price and molecular information. An entry might appear as below. The CML and other components do not interact other than that the molecular information is an integral part of the entry

```
<p:entry id="p123" sku="P-123-4">
  <p:price amount="1000" unit="ml">
    <p:currency="USD">100</p:currency>
    <p:currency="GBP">66</p:currency>
  </p:price>
  <c:molecule id="p1" title="phosphine" xmlns:c="http://www.xml-cml.org/schema/cml2/core>
    <c:atomArray>
      <c:atom elementType="P" hydrogenCount="3"/>
    </c:atomArray>
  </c:molecule>
</p:entry>
```

- It often encourages the implementor to think more clearly about the semantics. A first pass at creating a markup language often benefits from being modularised, even if the modules are later found to be unnecessary, and combined for easier understanding.

Modularization is not always possible. This happens when concepts are not "atomic" **-** e.g. where A relies on information in B. This is an example of context-dependency, or coupling, and can be very difficult to model. STMML components are not coupled and can be used as independent components in a larger aggregate.

In many cases there are implicit relationships between components, and no simple way of implementing them automatically. Thus when modelling the concept of *concentration* for CML we found it necessary to create two new elements, `substance` and `substanceList` which are now part of CML 2.0 (Murray-Rust and Rzepa, 2002b). Similarly in modelling interatomic `angle`s in molecules there was an implicit assumption that all atoms belonged to a common coordinate system.

Mappings and comparisons are particularly challenging. Thus a "sequence alignment" (*e.g.* for proteins) is a mapping of 2 or more sequences. A common method is to insert "gaps" into the sequences such as:

```
ACDEF-GH
A-DEFQGH
```

The gaps, however, are meaningless without the context and could change if other sequences are added. We tackle this by creating a `link` element, which can express relationships between different information components. Our strategy is based on the XLink specification (World Wide Web Consortium) and depending on the development of generic XLink tools we may increasingly use that symbolism.

5. The need for validation

It has been estimated that a large proportion of software errors, perhaps up to 50%, arise from "invalid data", i.e. data input that does not conform to the concepts for which the software was written. Non-conformance can be due to data corruption, inputting data for a different (version of the) program, data out of range or otherwise meaningless. There are famous cases (Olken and McCarthy, 2000) where use of the wrong scientific units have been catastrophic. Perhaps the most pernicious is when a user has a different interpretation of the semantics; the data appear valid but are actually inconsistent with the program.

Schema-based *Validation* goes a long way to solving these problems. Essentially a Schema is a *machine processible contract between the program author and the user*. The programmer uses the schema to write the program, and we have found it to be a very powerful tool. In fact, for many Schemas useful code can be automatically generated from the schema. If the data can be shown to be valid, then the program should process them reliably. If this does not happen, this is a program error. If the data are invalid, it is the user's responsibility. This separation of responsibilities is far more important than is generally realised and is a major tool in increasing the quality of software.

Validity is a machine-based concept. The more that the human view of validity can be formally encoded, the more useful it becomes. In chemistry for example, is the representation "T" (often written for tritium, an isotope of hydrogen) a valid element Type? (In CMLCore Schema it is not). Software often does not define openly what values are allowed. Using XML Schemas, we can give a precise enumeration of allowed values. Is a negative crystallographic cell angle allowed? Most users and software developers would assume not, but this is not normally checked or disallowed.

XML Schemas allow three main types of validation:

- Datatyping. The form and the value of the string must conform. XML Schema gives 43 datatypes, all of which can be used in STMML. In addition patterns can be imposed. For example a ID for an information component could be required to be of the form:
-
- `[A-Z]{6}([0-9][0-9])?`
  specifying six alpha characters followed by two optional digits (e.g. AABHTZ, ABCDEF01). Note, however, that XML Schema *per se* has no support for scientific units, which is why we use STMML to add them.

- Document structure. It is possible to specify the attributes (and their datatypes and defaults) for all elements. An element's content model (the child nodes) can also be defined, with considerable power. Thus in CML 2.0, a `molecule` can contain a `formula`, and a `formula` can contain either other `formula` or `atomArray` elements.

- There are many rules than cannot be enforced by the above two schema-based approaches. However, the XSLT language, which supports the formulation of rules, is particularly good at specifying constraints, including those which are context dependent. We use this extensively in CML 2.0, where components are strongly coupled. For example, a bond must reference two distinct existing atoms in the document. In STMML the coupling is low and so this approach is not much used. However those using STMML as part of their infrastructure may need to introduce coupling.

5.1 Datatypes

STM data requires a range of datatypes, many of them representable by a character string such as "1.234E+05", "Sb", "2001-12-31". In XML V1.0 these cannot be validated, but XML Schema introduces 43 concrete datatypes for `<xsd:simpleType>`s (Figure 1).

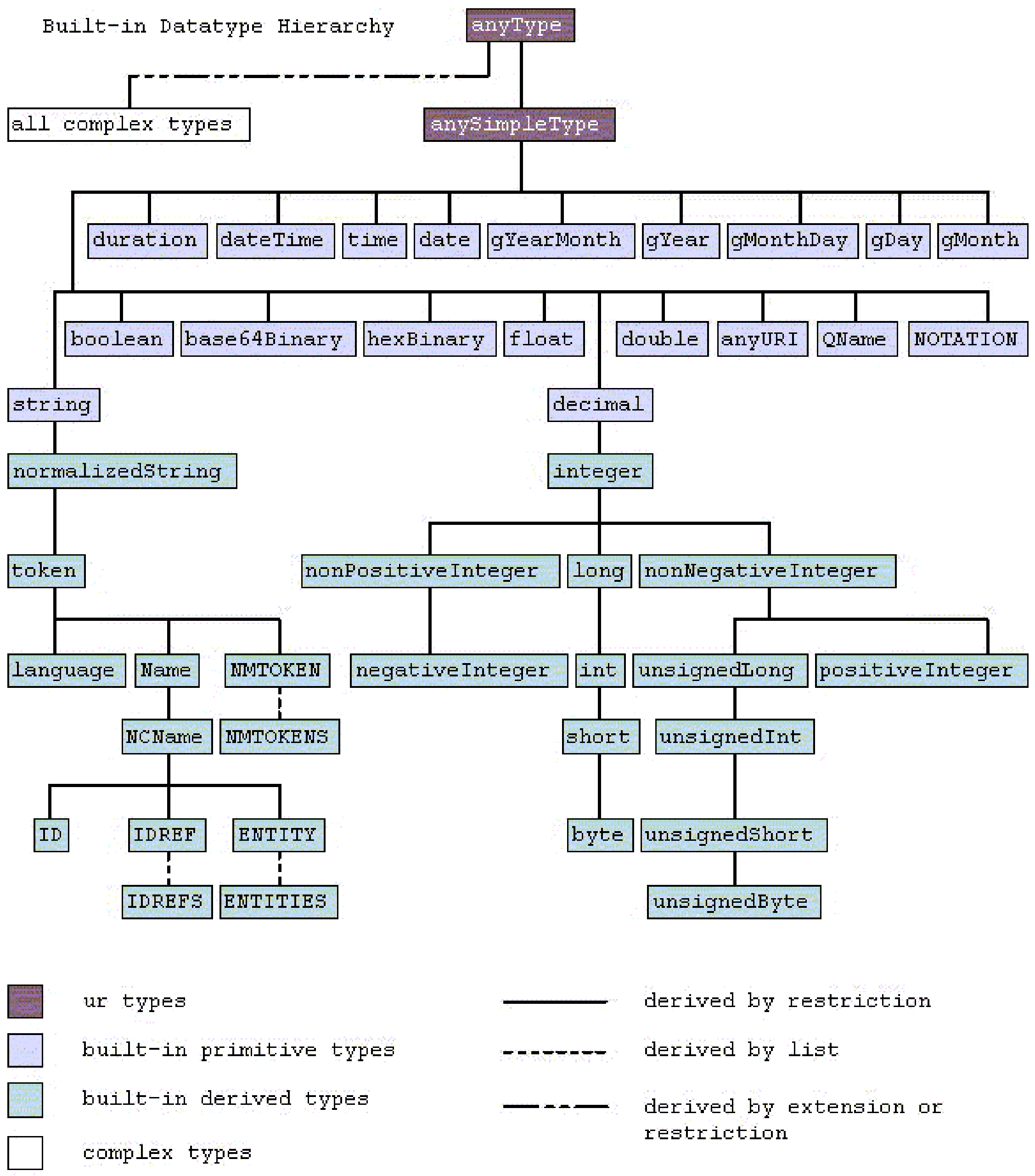

Figure 1. W3C Datatypes (taken from W3C Schema Part II datatypes, World Wide Web Consortium, n.d.).

If a value does not conform, the document is invalid. Among the constraints (some are referred to as facets) are:

- The type (number, integer, nonNegativeInteger, date, URI, etc.)
- The value. An enumerated list of values can be given; alternatively maximum and/or minimum values can be set
- The lexical pattern. This conforms to a POSIX regular expression giving character types/values, and possible occurrence counts

It is possible to construct new datatypes *derived from* XML Schema datatypes. Examples of types defined in STMML are shown in Table 1.

These datatypes are themselves extensible; thus a complex number can be represented by `coordinate2` with the addition of a `convention` attribute to emphasize the semantics. Moreover the enumerated lists in the Schema can be extended to incorporate other values; a user could define additional types of matrices (but would also have to provide the processing semantics).

Table 1. Datatypes in STMML

type	**role**	**example**
coordinate2Type	**a 2-D Coordinate**	1.2 2.3
coordinate3Type	**a 3-D Coordinate**	1.2 2.3 3.4
dataTypeType	**data types definable in a dictionary entry**	xsd:float
delimiterType	**delimiters for separating numbers**	/
errorBasisType	**basis of errors**	range
errorValueType	**type of errors**	standard error
idType	**An ID**	a123
matrixType	**matrix Type**	upperTriangular
namespaceRefType	**namespaced data**	core:mpt
unitsType	**scientific units**	units:ml

5.2 Structure

XMLSchema constrains the structure of documents in two ways:

- Specifying the attributes that an element may have. The constraints on the structure are fairly similar to those expressible by DTDs in XML 1.0, but there are additions to support inheritability of attributes.
- Specifying the content model of elements. There are several developments:
    - The & connector was defined in SGML content models to support unordered element children ("must have an A, B and C, but order doesn't matter"). This was removed in XML V1.0 as being complex to implement and ambiguous. The `all` element goes some way towards adding this functionality, but it is not used in STMML.
    - It is possible to constraint child elements by their namespace (e.g. to require that only certain CML elements are allowed as content of `<atom>`, but that elements from other namespaces may or may not be forbidden). In general all STMML elements designed as containers (`<matrix>`, `<list>`, etc. can contain any elements.
    - New complexTypes can be built and reused. STMML itself does not define complexTypes for re-use but it is likely that users will build their own from STMML components.

5.3 Rules

An XML Schema is a set of rules to which a document must conform. It has hardcoded mechanisms for datatypes and the attributes and direct content of elements. Constraints outside these cannot be expressed in XML Schema. A common need is the conditional dependence (coupling) of one part of the document on another part: "if part A has this structure and/or values, then part B must have that structure and/or values. A CML example is: "if a `molecule` has less than four `atom`s, it cannot have any `torsion` elements". This requires a knowledge of the document context which XML Schema cannot provide.

XSLT (World wide Web Consortium, n.d.) was developed as a rule-based transformation language. It allows very powerful determination of context through XPath. The constraint above could be expressed as:

```
<xsl:template match="torsion">
  <xsl:if test="count(../atomArray/atom)>3">
    <xsl:message>TORSION REQUIRES FOUR ATOMS</xsl:message>
  </xsl:if>
</xsl:template>
```

In principle, therefore, XML validation could take place through XSLT and not through Schemas. The Schematron approach (Schematron, n.d.) is essentially a mechanism for using such stylesheets to validate XML documents in place of a Schema. In practice we believe that XMLSchema is the cleanest approach for those things it does well (particularly datatypes), but that XSLT should be used for constraints that cannot be easily expressed with the current syntax. Fortunately it is easy to integrate the two approaches as XML Schema has an `appinfo` element for adding *machine-processable* documentation to an element or attribute. Certain tools can scan the `appinfo` and apply XSLT-based rules for additional validation.

There is a strong requirement for structural constraints in STM data. They often take forms like:

> *"If the boiling point is reported, the pressure must also be given".*
> *"If the data are two dimensional then the matrix is of order 3, otherwise of order 4".*
> *"If x has this value then y must be present".*

In some cases this can be represented in XMLSchemas or DTDs. For example, one of us (Murray-Rust, 2001) has converted the SELF format (Kehiaian, 2001) for physicochemical data into XML. SELF consists of many different measurement types (*e.g.* critical point, vapour pressure by temperature) with a variety of numbers of independent and dependent variables. Each data structure (one per measurement type) can be represented by a DTD. Effectively the DTD fragment is stored with the dictionary entry for the measurement type and can be used to XML-validate the data instance. We can foresee this being a useful future development for STMML.

6. The STMML Schema

XML Schemas are complex. The formal XML Schema recommendations run to over 300 pages and require a 60 page primer to get started (World Wide Web Consortium, n.d.). We therefore introduce Schemas with examples, and also append the STMML schema itself as Appendix 2 to this article.

Schemas are designed to be easily documented, and since they are in XML, they can be transformed into common formats such as XHTML and using XSL-FO, into Acrobat PDF (Gkoutos, Murray-Rust, Rzepa and Wright, 2001a). They can be searched and can be reorganised into different presentations. Documentation is available for the schema itself and for most components such as elements, attributes, attributeGroups, etc. This allows programs to extract documentation directly from the schema and associate with the component being used, perhaps on an online help system or for automatic archival. The top-level element for this is `annotation` which can contain `documentation` for human readers and `appinfo` for machine processing. The latter is a major advance in that it allows a wide variety of functionality to be added on a per-element or per-attribute basis.

STMML uses a subset of the XMLSchema vocabulary:

"Top-level" elements

- **annotation**. Manages document for the *schema*
- **element**. Defines an element (cf. `<!ELEMENT` in DTDs)
- **attribute**. Defines an attribute (cf. `<!ATTLIST` in DTDs)
- **group**,**attributeGroup**. Mainly for ease of maintenance (cf. parameter entities in DTDs)
- **simpleType**. Defines a simpleType (essentially a datatyped character string. The type can be built-in (e.g. `xsd:integer` or can be derived from a builtin type (all are ultimately derived from `xsd:string`). simpleTypes can be used either in attribute values or for text content (`#PCDATA` in DTDs)

- **complexType**. Defines a complexType (a type built up from simpler components such as other elements and/or attributes)

Content-related. These elements constrain the content of elements:

- **choice**. A series of alternatives (cf the "|" connector in DTDs)
- **sequence**. An ordered list of possible children (cf the "," connector in DTDs)
- **all**. An unordered list of possible children (cf the "&" connector in SGML DTDs)
- **any**. Any child content (with constraints on namespaces (cf `ANY` in DTDs)

Occurrence counts for elements and the above constructs now have explicit attributes (`minOccurs`, `maxOccurs` which allow for greater power in representing cardinality.

6.1 An example - part of the crystal definition in CMLCore Schema

This extract from STMML Schema shows many of the constructs that we use. It epitomises the re-use of STMML components to build a specialist element and relies on previous definition of some simpleTypes and attributeGroups. Comments are shown in italics.

```
This specifies the structure and vocabulary of the "crystal" element
<xsd:element name="crystal" id="el.crystal">
documentation...
  <xsd:annotation>
    <xsd:documentation>
      <div class="summary">A container for crystallographic
        cell parameters and spacegroup</div>
      <div class="description">
        <p>Required if
          fractional coordinates are provided for a molecule.</p>
        <p>There are precisely SIX child <tt>scalar</tt>s
          to represent the cell lengths
          and angles in that order. There are no default values; </p>
      </div>
      <div class="example"><pre>

example...
<molecule id="m1">
  <crystal spacegroup="Fm3m" z="4">
    <stm:scalar title="a" errorValue="0.001" units="angstrom">5.628</stm:scalar>
    <stm:scalar title="b" errorValue="0.001" units="angstrom">5.628</stm:scalar>
    <stm:scalar title="c" errorValue="0.001" units="angstrom">5.628</stm:scalar>
    <stm:scalar title="alpha" errorValue="0">90</stm:scalar>
    <stm:scalar title="beta" errorValue="0">90</stm:scalar>
    <stm:scalar title="gamma" errorValue="0">90</stm:scalar>
  </crystal>
  <atomArray>
    <atom id="a1" elementType="Na" formalCharge="1" xyzFract="0.0 0.0 0.0" xy2="+23.2 -21.0"/>
    <atom id="a2" elementType="Cl" formalCharge="-1" xyzFract="0.5 0.0 0.0"/>
  </atomArray>
</molecule>

        </pre></div>
    </xsd:documentation>
  </xsd:annotation>
"complexType" can have both element children and attributes...
  <xsd:complexType>
precisely 6 children allowed (the cell parameters).
    <xsd:sequence>
      <xsd:element ref="stm:scalar" minOccurs="6" maxOccurs="6"/>
    </xsd:sequence>
an attribute describing the spacegroup...
    <xsd:attribute name="spacegroup" type="xsd:string">
      <xsd:annotation>
        <xsd:documentation>
          <div class="summary">The spacegroup as a symbol</div>
          <div class="description">There is no controlled vocabulary or
                   pattern at present. Hermann-Mauguin or Hall
                   symbols are recommended
          </div>
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
another attribute (describing the number of molecules/cell).
This is restricted to be a non-negative integer...
    <xsd:attribute name="z" type="xsd:nonNegativeInteger">
      <xsd:annotation>
        <xsd:documentation>
```

```
          <p>The number of molecules per cell. Molecules
            are defined as the <tt>molecule</tt> which directly contains
            the <tt>crystal</tt> element.
          </p>
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>

more attributes (bundled in a group for ease of maintenance)...
    <xsd:attributeGroup ref="tit_id_conv_dictGroup"/>
  </xsd:complexType>
</xsd:element>
```

6.2 Components of STMML Schema

The main components are elements and attributes (Table 2). Elements are always declared as such, while attributes may be contained in elements, defined as simpleTypes, or bundled in groups of attributes (attributeGroups). This table gives a brief phrase describing each component, and some are described in more detail in the next section. Parts of the schema are primarily for ease of maintenance (simpleTypes rather than element-bound attributes, and attributeGroups) and are not elaborated here.

The whole schema (Appendix 2) is part of this "datument" publication and can be inspected or processed with specialist XMLSchema-aware tools such as Xerces (Xerces, n.d.). For example the STMML Schema itself has been transformed and styled using XSLT, and the examples included in the `appinfo` element have all themselves been validated with the STMML XMLSchema. Many technical authors will realise the benefit of a single self-consistent source of content; XSLT/XMLSchema allows it to be reworked without the likelihood of corruption through editing such as cut-and-paste or transformation in conventional document-processing tools.

Table 2. Components of STMML Schema

Data structure

array	A homogenous 1-dimensional array of similar objects.
scalar	An element to hold scalar data.
matrix	A rectangular matrix of any quantities
table	A rectangular table of any quantities
list	A generic container with no implied semantics

Data-based simpleTypes

coordinate2Type	An x/y coordinate pair
coordinate3Type	An x/y/z coordinate triple
dataTypeType	an enumerated type for all builtin allowed dataTypes in STM

Array-based simpleTypes

sizeType	The size of an array
delimiterType	A non-whitespace character used in arrays to separate components
floatArrayType	An array of floats
matrixType	Allowed matrix types
countType	A count multiplier for certain elements

General simpleTypes

General components

action	An action which might occur in scientific data or narrative.
actionList	A container for a group of actions
object	An object which might occur in scientific data or narrative
observation	An observation or occurrence

Dictionary components

dictionary	A dictionary
entry	A dictionary entry
definition	The definition for a dictionary entry, scientific units, etc.
description	Descriptive information in a dictionary entry, etc.
enumeration	An enumeration of string values associated with an entry
alternative	An alternative name for an entry
relatedEntry	An entry related in some way to a dictionary entry, scientific units, etc.
annotation	A documentation container similar to annotation in XML Schema.
documentation	Documentation in the annotation of an entry
appinfo	A container similar to appinfo in XML Schema.
dictRefGroup	A reference to a dictionary entry.

idType
A unique ID for STM elements

userTypeType
An uncontrolled vocabulary from a user

idGroup
A attribute providing a unique ID for STM elements

titleGroup
An (optional) title on most STM elements. Uncontrolled

convGroup
A reference to a convention

Numeric types

errorBasisType
The basis of an error value

maxType
The maximum INCLUSIVE value of a quantity

minType
The minimum INCLUSIVE value of a quantity

Links and References

link
An internal or external link to STMML or other object(s)

References

namespaceRefType
A string referencing a dictionary, units, convention or other metadata.

refType
A reference to an existing element

Metadata

metadata
A general container for metadata

metadataList
A general container for metadata elements

metadataType
metadataType

Scientific Units

dimension
A dimension supporting scientific units

dimensionType
Allowed values for dimension Types (for quantities).

unitList
A container for several unit entries

unitType
An element containing the type of a scientific unit

unit
A scientific unit

unitsType
Scientific units

Groups (for schema maintenance and re-use)

actionGroup
The start time

sourceGroup
An attribute linking to the source of the information

tit_id_convGroup
An group of attribute applicable to most STMML elements

tit_id_conv_dictGroup
Addition of the dictRef attribute to the tit_id_conv group.

7 Conclusion

We have shown how STMML as an XML-based markup language can cover many generic aspects of scientific information and how it has been developed as a re-usable core for more specific markup languages. The specification is through an XML Schema which can be used to validate STMML documents or fragments, and the schema itself (the "datument) has been used to validate and present through appropriate transformations all the examples included here.

8. References

Cell Markup Language: (n.d.). Home page for CellML. Retrieved from the World Wide Web July 9, 2002: http://www.cellml.org/

Cover, R. (2002), *The XML Cover Pages*. Retrieved July 9, 2002 from the World Wide Web: http://xml.coverpages.org/siteIndex.html

DocBook (n.d.). Home page for DocBook: The Definitive Guide. Retrieved April 15, 2002 from the World Wide Web: http://www.docbook.org/

Dragoset, R., Taylor, B. & McLay, M. (2002). Units of Measurement in XML: Retrieved July 15, 2002 from the NIST Website: http://units.nist.gov/

Field Markup Language (n.d.). Homepage for Field ML. Retrieved July 9, 2002 from the World Wide Web, 2002: http://www.physiome.org.nz/sites/physiome/fieldml/pages/

Formatting Objects (n.d.). Homepage for FOP. Retrieved July 9, 2002 from the *Apache XML Project* Website: http://xml.apache.org/fop/

Gene Ontology Consortium (n.d.). Homepage for the Gene Ontology Consortium. Retrieved July 9, 2002 from the World Wide Web: http://www.geneontology.org/

GML (n.d.). Homepage for Geography Markup Language. Retrieved July 9, 2002 from the World Wide Web: http://opengis.net/gml/01-029/GML2.html

Gkoutos, G. V., Leach, C & Rzepa, H. S., (2002) ChemDig: new approaches to chemically significant indexing and searching of distributed web collections *New. J. Chem.*, 656-666.

Gkoutos, G. V., Murray-Rust, P., Rzepa, H. S. & Wright, M., (2001a) The Application of XML Languages for Integrating Molecular Resources, *Internet J. Chemistry, 4,* article 13.

Gkoutos, G. V., Murray-Rust, P., Rzepa, H. S. & Wright, M., (2001b), Chemical Markup, XML and the World-Wide Web. Part III: Towards a signed semanticcChemical web of trust, *J. Chem. Inf. Comp. Sci.*, *41*, 1124.

Hall, S. R., Allen, F. H. & Brown, I. D., (1991), *Acta Cryst*., *A47*, 655-685.

Kehiaian, H. (2001), Homepage for DataExplorer. Retrieved July 9, 2002 from the World Wide Web: http://www.fiz-karlsruhe.de/dataexplorer/index.html

Murray-Rust, P., (2001), SELFML, presented at the IUCOSPED/CODATA meeting, Paris, April. Retrieved July 9, 2002 from the World Wide Web: http://www-cenerg.ensmp.fr/iupac_paris.wshop/html/meetingweb.htm

Murray-Rust, P., Leach, C. & Rzepa, H. S., (1995), Chemical Markup Language, *Abstr. Pap. Am. Chem. S., 210,* 40-COMP Part 1.

Murray-Rust, P. & Rzepa, H. S., (1999), Chemical markup Language and XML Part I. Basic principles. *J. Chem. Inf. Comp. Sci. 39*, 928.

Murray-Rust, P. & Rzepa, H. S., (2001), Chemical Markup, XML and the World-Wide Web. Part II: Information Objects and the CMLDOM. *J. Chem. Inf. Comp. Sci. 41*, 1113.

Murray-Rust, P. & Rzepa, H. S., (2002a) Scientific publications in XML - towards a global knowledge base. *Data Science 1*, 84-98.

Murray-Rust, P. & Rzepa, H. S., (2002b) Chemical Markup Language and XML Part IV. Schema Validation. *J. Chem. Inf. Comp. Sci., 42*, submitted for publication.

Murray-Rust, P., Rzepa, H. S., & Wright, M. (2001) Development of chemical markup language (CML) as a system for handling complex chemical content, *New J. Chem.*, 618-634.

Olken F. & McCarthy J. (2000) *What XML Schema Designers Need to Know About Measurement Units*. Retrieved July 9, 2002 from the Lawrence Berkeley National Laboratory Website: http://pueblo.lbl.gov/~olken/mendel/units/xtech_units_slides.ppt

PlotML (n.d.). *Homepage for the Ptolemy Project - Ptolemy II*. Retrieved July 9, 2002 from the University of California at Berkeley Department of Electrical Egnineering and Computer Sciences Website: http://ptolemy.eecs.berkeley.edu/ptolemyII/ptII1.0/

Rzepa, H. S & Murray-Rust, P., (2001), A New Publishing Paradigm: STM Articles as part of the Semantic Web, *Learned Publishing 14*, 177.

Schematron (n.d.). Homepage for the Schematron Document Schema Definition Language. Retrieved July 9, 2002 from the World Wide Web : http://xml.ascc.net/xml/resource/schematron/schematron.html

SelfML (n.d.). Homepage for the SelFML Language. Retrieved July 9, 2002 from the World Wide Web: http://www.xml-cml.org/self/

Systems Biology Markup Language (n.d.). Home page for Systems Biology Workbench Development Group. Retrieved July 9, 2002 from the Californian Institute of Technology Website: http://www.cds.caltech.edu/erato/

SVG (n.d.). Home page for Scaleable vector graphics. Retrieved July 9, 2002 from the *World Wide Web Consortium* Website: http://www.w3.org/Graphics/SVG/Overview.htm8

Table models (n.d.). Homepage for specifications for Table models. Retrieved July 9, 2002 from the *OASIS* Website: http://oasis-open.org/specs/tablemodels.shtml

World Wide Web Consortium (n.d.) Homepage of the World Wide Web Consortium. Retrieved July 9, 2002 from the World Wide Web: http://www.w3c.org/

Xerces (n.d.). Home page for the Xerces validating parser. Retrieved July 9, 2002 from the *Apache XML Project* Website: http://xml.apache.org/

Appendix 1. Synopsis of Schema Details

The full Schema is part of this datument as Appendix 2 and via http://www.xml-cml.org/schema/stmml/ but because of its comprehensive nature we have added this synopsis for initial readers. Many details are omitted, but most of the significant examples are retained. All elements and the main generic attributes are described but individual attributes are usually omitted.

The synopsis is organised by the concepts:

- Data Types and Data Structure
- General Information Components
    - General Scientific Concepts
    - Dictionaries
    - Metadata
    - Scientific Units

Data Types and Data Structure

Overview

Data structure

array

A homogenous 1-dimensional array of similar objects.

```
<array size="5" title="value"
  dataType="xsd:decimal">  1.23 2.34 3.45 4.56 5.67</array>
```

the `size` attribute is not mandatory but provides a useful validity check):

```
<array size="5" title="initials" dataType="xsd:string"
  delimiter="/">/A B//C/D-E/F/</array>
```

Note that the second array-element is the empty string ''.

```
<array title="mass" size="4"
  units="unit:g"
  errorBasis="observedStandardDeviation"
  minValues="10 11 10 9"
  maxValues="12 14 12 11"
  errorValues="1 2 1 1"
  dataType="xsd:float">11 12.5 10.9 10.2
</array>
```

scalar

An element to hold scalar data.

```
<scalar
    dataType="xsd:decimal"
    errorValue="1.0"
    errorBasis="observedStandardDeviation"
    title="body weight"
    dictRef="zoo:bodywt"
    units="units:g">34.3</scalar>
```

matrix

A rectangular matrix of any quantities

```
<matrix id="m1" title="mattrix-1" dictRef="foo:bar"
  rows="3" columns="3" dataType="xsd:decimal"
  delimiter="|" matrixType="squareSymmetric" units="unit:m"
  >|1.1|1.2|1.3|1.2|2.2|2.3|1.3|2.3|3.3!</matrix>
```

table

A rectangular table of any quantities

```
<table rows="3" columns="2" title="people">
  <array title="age" dataType="xsd:integer">3 5 7</array>
  <array title="name" dataType="xsd:string">Sue Fred Sandy</array>
</table>
```

list

A generic container with no implied semantics

```
<list>
  <array title="animals">frog bear toad</array>
  <scalar title="weight" dataType="xsd:float">3.456</scalar>
</list>
```

Data-based simpleTypes

coordinate2Type

An x/y coordinate pair

```
<list>
  <array
      >1.2,3.4   3.2,4.5   6.7,23.1 </array>
  <array delimiter="/"
      >/1.2 3.4/3.2 4.5/6.7 23.1/</array>
</list>
```

coordinate3Type

An x/y/z coordinate triple

```
<list>
  <array>1.2,3.4,1.2
    3.2,4.5,7.3   6.7,23.1,5.6 </array>
  <array delimiter="/"
  >/1.2 3.4 3.3/3.2 4.5 4.5/6.7 23.1 5.6/</array>
</list>
```

dataTypeType

an enumerated type for all builtin allowed dataTypes in STM

```
<list xmlns="http://www.xml-cml.org/schema/cml2/core">
  <scalar dataType="xsd:boolean" title="she loves me">true</scalar>
  <scalar dataType="xsd:float" title="x">23.2</scalar>
  <scalar dataType="xsd:duration" title="egg timer">PM4</scalar>
  <scalar dataType="xsd:dateTime" title="current data and time">2001-02-01:00:30</scalar>
  <scalar dataType="xsd:time" title="wake up">06:00</scalar>
  <scalar dataType="xsd:date" title="where is it">1752-09-10</scalar>
  <scalar dataType="xsd:anyURI" title="CML site">http://www.xml-cml.org/</scalar>
  <scalar dataType="xsd:QName" title="CML atom">cml:atom</scalar>
  <scalar dataType="xsd:normalizedString" title="song">the mouse ran up the clock</scalar>
  <scalar dataType="xsd:language" title="UK English">en-GB</scalar>
  <scalar dataType="xsd:Name" title="atom">atom</scalar>
  <scalar dataType="xsd:ID" title="XML ID">_123</scalar>
  <scalar dataType="xsd:integer" title="the answer">42</scalar>
  <scalar dataType="xsd:nonPositiveInteger" title="zero">0</scalar>
</list>
```

Array-based simpleTypes

sizeType

The size of an array

delimiterType

A non-whitespace character used in arrays to separate components

```
<array size="4" delimiter="|">|A|B12||D and   E|</array>
The values in the array are
    "A", "B12", "" (empty string) and "D and   E"
    note the spaces
```

floatArrayType

An array of floats

```
<atomArray xmlns="http://www.xml-cml.org/schema/cml2/core"
  x2="1.2 2.3 3.4 5.6"/>
```

matrixType

Allowed matrix types

```
<matrix id="m1" title="mattrix-1" dictRef="foo:bar"
  rows="3" columns="3" dataType="xsd:decimal"
  delimiter="|" matrixType="squareSymmetric" units="unit:m"
  >|1.1|1.2|1.3|1.2|2.2|2.3|1.3|2.3|3.3!</matrix>
```

BASE: xsd:string
ENUMERATED VALUES:
* rectangular
* square
* squareSymmetric
* squareAntisymmetric
* diagonal
* upperTriangular
* lowerTriangular
* unitary
* rowEigenvectors
* rotation22
* rotationTranslation32
* homogeneous33
* rotation33
* rotationTranslation43
* homogeneous44
* square
* square

countType

A count multiplier for certain elements

```
<list>
<object title="frog" count="10"/>
<action title="step3" count="3">
  <p>Add 10 ml reagent</p>
</action>
</list>
```

General simpleTypes

idType

A unique ID for STM elements

userTypeType

An uncontrolled vocabulary from a user

```
<alternative type="userType" userType="biological name">SusScrofa</alternative>
```

idGroup

Attributes in group

id	idType

A attribute providing a unique ID for STM elements

titleGroup

Attributes in group

title	xsd:string	

An (optional) title on most STM elements. Uncontrolled

convGroup

Attributes in group

convention	namespaceRefType	[default=CML]

A reference to a convention

```
<bond convention="fooChem" order="-5"
   xmlns:fooChem="http://www.fooChem/conventions"/>
```

Numeric types

errorBasisType

The basis of an error value

```
<scalar
    dataType="xsd:decimal"
    errorValue="1.0"
    errorBasis="observedStandardDeviation"
    title="body weight"
    dictRef="zoo:bodywt"
    units="units:g">34.3</scalar>
```

BASE: xsd:string

ENUMERATED VALUES:

* observedRange
* observedStandardDeviation
* observedStandardError
* estimatedStandardDeviation
* estimatedStandardError

maxType

The maximum INCLUSIVE value of a quantity

```
<scalar dataType="xsd:float" maxValue="20" minValue="12">15</scalar>
```

minType

The minimum INCLUSIVE value of a quantity

```
<scalar dataType="xsd:float" maxValue="20" minValue="12">15</scalar>
```

Links and References

link

An internal or external link to STMML or other object(s)

References

namespaceRefType

A string referencing a dictionary, units, convention or other metadata.

```
<list>
<!-- dictRef is of namespaceRefType -->
  <scalar dictRef="chem:mpt">123</scalar>
<!-- error -->
  <scalar dictRef="mpt23">123</scalar>
</list>
```

refType

A reference to an existing element

BASE: idType

General information components

General components

action

An action which might occur in scientific data or narrative.

```
<actionList title="boiling two eggs for breakfast">
  <!-- start cooking at 9am -->
  <action title="turn on heat" start="T09:00:00" convention="xsd"/>
  <!-- human readable description of time to start action -->
  <action title="put egg into pan" startCondition="water is boiling" count="2"/>
  <!-- the duration is expressed in ISO8601 format -->
  <action title="boil eggs for 4 minutes" duration="4" units="units:min"/>
  <!-- action immediately follows last action -->
  <action title="remove egg from pan" count="1"/>
  <action title="boil second egg for a bit longer" duration="about half a minute"/>
  <!-- action immediately follows last action -->
  <action title="remove egg from pan" count="1"/>
</actionList>

<actionList title="preparation of silanols">
  <p>This is a conversion of a chemical synthesis to STMML. We
  have deliberately not marked up the chemistry in this example!</p>
  <action title="step2">
    <p>Take 1 mmol of the diol and dissolve in dioxan in
      <object title="flask">
        <scalar title="volume" units="units:ml">25</scalar>
      </object>
    </p>
  </action>
  <action title="step2">
    <p>Place flask in water bath with magnetic stirrer</p>
  </action>
  <!-- wait until certain condition -->
  <actionList endCondition="bath temperature stabilised"/>
  <action title="step3">
    <p>Add 0.5 ml 1 M H2SO4</p>
  </action>

  <!-- carry out reaction -->
  <actionList endCondition="reaction complete; no diol spot remains on TLC">
    <actionList title="check tlc">
      <!-- wait for half an hour -->
      <action duration="half an hour"/>
      <action title="tlc">
        <p>extract solution and check diol spot on TLC</p>
      </action>
    </actionList>
  </actionList>

  <!-- work up reaction -->
  <action title="step5">
    <p>Add 10 ml water to flask</p>
  </action>
  <action title="step6">
    <p>Neutralize acid with 10% NaHCO3</p>
  </action>
  <action title="step7" count="3">
    <p>Extract with 10ml ether</p>
  </action>
  <action title="step8">
    <p>Combine ether layers</p>
  </action>
  <action title="step9" count="2">
    <p>Wash ether with 10 ml water</p>
  </action>
  <action title="step10">
    <p>Wash ether with 10 ml saturated NaCl</p>
  </action>
  <action title="step11">
    <p>Dry over anhydrous Na2SO4 and remove solvent on rotary evaporator</p>
  </action>
</actionList>
```

actionList

A container for a group of actions
See examples in action

```
<!-- demonstrating parallel and sequential actions -->
<actionList order="parallel" endCondition="all food cooked">
  <!-- meat and potatoes are cooked in parallel -->
  <actionList title="meat">
    <action title="cook" endCondition="cooked">
      <p>Roast meat</p>
    </action>
    <action><p>Keep warm in oven</p></action>
  </actionList>
  <actionList title="vegetables">
    <actionList title="cookVeg" endCondition="cooked">
      <action title="boil water" endCondition="water boiling">
        <p>Heat water</p>
      </action>
      <action title="cook" endCondition="potatoes cooked">
        <p>Cook potatoes</p>
      </action>
    </actionList>
    <action><p>Keep warm in oven</p></action>
  </actionList>
</actionList>
```

object

An object which might occur in scientific data or narrative

```
<object title="frog" type="amphibian" count="5">
  <scalar dataType="xsd:float" title="length" units="unit:cm">5</scalar>
</object>
```

observation

An observation or occurrence

Dictionary components

dictionary

A dictionary

```
<stm:dictionary xmlns:stm="http://www.xml-cml.org/schema/stmml">
  <stm:entry id="a001" term="Amplitude for charge density mixing"
    dataType="stm:decimal"
    units="arbitrary">
    <stm:annotation>
      <stm:documentation>
        <div class="summary">Amplitude for charge density mixing</div>
        <div class="description">Not yet filled in...</div>
      </stm:documentation>
    </stm:annotation>
    <stm:alternative type="abbreviation">CDMixAmp</stm:alternative>
  </stm:entry>
</stm:dictionary>
```

`dictionary` can be used in an instance document to reference the dictionary used. Example:

```
<list>
  <dictionary
     dictRef="core" href="../dictionary/coreDict.xml"/>
    <!-- ... -->
    <scalar dictRef="core:frog" title="specimens">Frogs</scalar>
</list>
```

entry

A dictionary entry

```
<s:entry id="a003" term="alpha"
  dataType="float"
  minInclusive="0.0"
  maxInclusive="180.0"
  recommendedUnits="degrees"
  xmlns:s="http://www.xml-cml.org/schema/stmml">
  <s:definition>The alpha cell angle</s:definition>
</s:entry>

<entry id="a003"
  term="matrix1"
  dataType="float"
  rows="3"
  columns="4"
  unitType="unit:length"
  minInclusive="0.0"
  maxInclusive="100.0"
  recommendedUnits="unit:m"
  totalDigits="8"
  fractionDigits="3">
  <definition>A matrix of lengths</definition>
  <description>A data instance will have a matrix which points
  to this entry (e.g. dictRef="foo:matrix1"). The matrix must
  be 3*4, composed of floats in 8.3 format, of type length,
  values between 0 and 100 and with recommended units metres.
  </description>
</entry>
```

definition

The definition for a dictionary entry, scientific units, etc.

From the IUPAC Dictionary of Medicinal Chemistry

```
<entry id="a7" term="Allosteric enzyme">
  <definition>An <a href="#e3">enzyme</a>
that contains a region to which small, regulatory molecules
("effectors") may bind in addition to and separate from the
substrate binding site and thereby affect the catalytic
activity.
  </definition>
  <description>On binding the effector, the catalytic activity of the
<strong>enzyme</strong> towards the substrate may be enhanced, in
which case the effector is an activator, or reduced, in which case
it is a de-activator or inhibitor.
  </description>
</entry>
```

description

Descriptive information in a dictionary entry, etc.
From IUPAC Dictionary of Medicinal Chemistry

```
<entry id="a7" term="Allosteric enzyme">
  <definition>An <a href="#e3">enzyme</a>
that contains a region to which small, regulatory molecules
("effectors") may bind in addition to and separate from the
substrate binding site and thereby affect the catalytic
activity.
  </definition>
  <description>On binding the effector, the catalytic activity of the
<strong>enzyme</strong> towards the substrate may be enhanced, in
which case the effector is an activator, or reduced, in which case
it is a de-activator or inhibitor.
  </description>
</entry>
```

enumeration

An enumeration of string values associated with an entry

```
<entry term="crystal system" id="cryst1" dataType="string">
  <definition>A crystal system</definition>
  <enumeration value="triclinic">
    <annotation>
      <documentation>
        <div class="summary">No constraints on lengths and angles</div>
      </documentation>
    </annotation>
  </enumeration>
  <enumeration value="monoclinic">
    <annotation>
      <documentation>
        <div class="summary">Two cell angles are right angles; no other constraints</div>
      </documentation>
    </annotation>
  </enumeration>
  <enumeration value="orthorhombic">
    <annotation>
      <documentation>
        <div class="summary">All three angles are right angles; no other constraints</div>
      </documentation>
    </annotation>
  </enumeration>
  <enumeration value="tetragonal">
    <annotation>
      <documentation>
        <div class="summary">Fourfold axis of symmetry; All three angles are right angles; two equal
cell lengths; no other constraints</div>
      </documentation>
    </annotation>
  </enumeration>
  <enumeration value="trigonal">
    <annotation>
      <documentation>
        <div class="summary">Threefold axis of symmetry; Two angles are right angles; one is 120
degrees; two equal lengths; no other constraints</div>
      </documentation>
    </annotation>
  </enumeration>
  <enumeration value="hexagonal">
    <annotation>
      <documentation>
        <div class="summary">Sixfold axis of symmetry; Two angles are right angles; one is 120
degrees; two equal lengths; no other constraints</div>
      </documentation>
    </annotation>
  </enumeration>
  <enumeration value="cubic">
    <annotation>
      <documentation>
        <div class="summary">All three angles are right angles; all cell lengths are equal</div>
      </documentation>
    </annotation>
  </enumeration>
</entry>
```

alternative

An alternative name for an entry

```
 <entry term="ammonia" id="a1">
   <alternative type="synonym">Spirits of hartshorn</alternative>
 </entry>
```

relatedEntry

An entry related in some way to a dictionary entry, scientific units, etc.

```
<stm:entry id="a14" term="Autoreceptor"
```

```
  xmlns:stm="http://www.xml-cml.org/schema/cml2/core">
  <stm:definition>An <strong>autoreceptor</strong>, present at a nerve ending, is
    a <a href="#r1">receptor</a>
    that regulates, via positive or negative feedback processes, the
    synthesis and/or release of its own physiological ligand.
  </stm:definition>
  <stm:relatedEntry type="seeAlso" href="#h4">Heteroreceptor).</stm:relatedEntry>
</stm:entry>
```

annotation

A documentation container similar to `annotation` in XML Schema.

```
<entry term="matrix">
  <annotation>
    <documentation>This refers to mathematical matrices</documentation>
    <appinfo>... some code to describe and support matrices ...</appinfo>
  </annotation>
</entry>
```

documentation

Documentation in the annotation of an entry

```
<stm:documentation id="source"
xmlns:stm="http://www.xml-cml.org/schema/stmml">
Transcribed from IUPAC website
</stm:documentation>
```

appinfo

A container similar to `appinfo` in XML Schema.

An example in XSLT where an element `foo` calls a bespoke template

.

```
<s:appinfo
  xmlns:s="http://www.xml-cml.org/schema/cml2/core"
  xmlns="http://www.w3.org/1999/XSL/Transform">
  <template match="foo">
    <call-template name="processFoo"/>
  </template>
</s:appinfo>
```

dictRefGroup

Attributes in group

dictRef	namespaceRefType

A reference to a dictionary entry.

```
<scalar dataType="xsd:float" title="surfaceArea"
  dictRef="cmlPhys:surfArea"
  xmlns:cmlPhys="http://www.xml-cml.org/dict/physical"
  units="units:cm2">50</scalar>

<stm:list xmlns:stm="http://www.xml-cml.org/schema/stmml">
  <stm:observation>
    <p>We observed <object count="3" dictRef="#p1"/>
      constructing dwellings of different material</p>
  </stm:observation>
  <stm:entry id="p1" term="pig">
    <stm:definition>A domesticated animal.</stm:definition>
    <stm:description>Predators include wolves</stm:description>
    <stm:description class="scientificName">Sus scrofa</stm:description>
  </stm:entry>
</stm:list>
```

Metadata

metadata

A general container for metadata

```
<list>
  <metadataList>
    <metadata name="dc:coverage" content="Europe"/>
    <metadata name="dc:description" content="Ornithological chemistry"/>
    <metadata name="dc:identifier"  content="ISBN:1234-5678"/>
    <metadata name="dc:format" content="printed"/>
    <metadata name="dc:relation" content="abc:def123"/>
    <metadata name="dc:rights" content="licence:GPL"/>
    <metadata name="dc:subject" content="Informatics"/>
    <metadata name="dc:title" content="birds"/>
    <metadata name="dc:type" content="bird books on chemistry"/>
    <metadata name="dc:contributor" content="Tux Penguin"/>
    <metadata name="dc:creator" content="author"/>
    <metadata name="dc:publisher" content="Penguinone publishing"/>
    <metadata name="dc:source" content="penguinPub"/>
    <metadata name="dc:language" content="en-GB"/>
    <metadata name="dc:date" content="1752-09-10"/>
  </metadataList>
  <metadataList>
    <metadata name="cmlm:safety" content="mostly harmless"/>
    <metadata name="cmlm:insilico" content="electronically produced"/>
    <metadata name="cmlm:structure" content="penguinone"/>
    <metadata name="cmlm:reaction" content="synthesis of penguinone"/>
    <metadata name="cmlm:identifier" content="smiles:O=C1C=C(C)C(C)(C)C(C)=C1"/>
  </metadataList>
</list>
```

metadataList

A general container for metadata elements

```
<list>
  <metadataList>
    <metadata name="dc:coverage" content="Europe"/>
    <metadata name="dc:description" content="Ornithological chemistry"/>
    <metadata name="dc:identifier"  content="ISBN:1234-5678"/>
    <metadata name="dc:format" content="printed"/>
    <metadata name="dc:relation" content="abc:def123"/>
    <metadata name="dc:rights" content="licence:GPL"/>
    <metadata name="dc:subject" content="Informatics"/>
    <metadata name="dc:title" content="birds"/>
    <metadata name="dc:type" content="bird books on chemistry"/>
    <metadata name="dc:contributor" content="Tux Penguin"/>
    <metadata name="dc:creator" content="author"/>
    <metadata name="dc:publisher" content="Penguinone publishing"/>
    <metadata name="dc:source" content="penguinPub"/>
    <metadata name="dc:language" content="en-GB"/>
    <metadata name="dc:date" content="1752-09-10"/>
  </metadataList>
  <metadataList>
    <metadata name="cmlm:safety" content="mostly harmless"/>
    <metadata name="cmlm:insilico" content="electronically produced"/>
    <metadata name="cmlm:structure" content="penguinone"/>
    <metadata name="cmlm:reaction" content="synthesis of penguinone"/>
    <metadata name="cmlm:identifier" content="smiles:O=C1C=C(C)C(C)(C)C(C)=C1"/>
  </metadataList>
</list>
```

metadataType

Scientific Units

dimension

A dimension supporting scientific units

```
<unitType id="energy" name="energy">
```

```
  <dimension name="length"/>
  <dimension name="mass"/>
  <dimension name="time" power="-1"/>
</unitType>
```

dimensionType

Allowed values for dimension Types (for quantities).

```
<unitType id="energy" name="energy">
  <dimension name="length"/>
  <dimension name="mass"/>
  <dimension name="time" power="-1"/>
</unitType>
```

BASE: xsd:string

ENUMERATED VALUES:

* mass
* length
* time
* charge
* amount
* luminosity
* temperature
* dimensionless
* angle

unitList

A container for several unit entries

```
<stm:unitList xmlns:stm="http://www.xml-cml.org/schema/stmml">

<!-- ================================================================== -->
<!-- ======================== fundamental types ========================== -->
<!-- ================================================================== -->

<stm:unitType id="length" name="length">
  <stm:dimension name="length"/>
</stm:unitType>

<stm:unitType id="time" name="time">
  <stm:dimension name="time"/>
</stm:unitType>

<!-- ... -->

<stm:unitType id="dimensionless" name="dimensionless">
  <stm:dimension name="dimensionless"/>
</stm:unitType>

<!-- ================================================================== -->
<!-- ========================= derived types ============================= -->
<!-- ================================================================== -->

<stm:unitType id="acceleration" name="acceleration">
  <stm:dimension name="length"/>
  <stm:dimension name="time" power="-2"/>
</stm:unitType>

<!-- ... -->

<!-- ================================================================== -->
<!-- ===================== fundamental SI units ========================== -->
<!-- ================================================================== -->

<stm:unit id="second" name="second" unitType="time">
  <stm:description>The SI unit of time</stm:description>
```

```
</stm:unit>

<stm:unit id="meter" name="meter" unitType="length"
  abbreviation="m">
  <stm:description>The SI unit of length</stm:description>
</stm:unit>

<!-- ... -->

<stm:unit id="kg" name="nameless" unitType="dimensionless"
  abbreviation="nodim">
  <stm:description>A fictitious parent for dimensionless units</stm:description>
</stm:unit>

<!-- ======================================================================= -->
<!-- ===================== derived SI units ================================ -->
<!-- ======================================================================= -->

<stm:unit id="newton" name="newton" unitType="force">
  <stm:description>The SI unit of force</stm:description>
</stm:unit>

<!-- ... -->

<!-- multiples of fundamental SI units -->

<stm:unit id="g" name="gram" unitType="mass"
  parentSI="kg"
  multiplierToSI="0.001"
  abbreviation="g">
  <stm:description>0.001 kg. </stm:description>
</stm:unit>

<stm:unit id="celsius" name="Celsius" parentSI="k"
  multiplierToSI="1"
  constantToSI="273.18">
  <stm:description><p>A common unit of temperature</p></stm:description>
</stm:unit>

<!-- fundamental non-SI units -->

<stm:unit id="inch" name="inch" parentSI="meter"
   abbreviation="in"
  multiplierToSI="0.0254" >
  <stm:description>An imperial measure of length</stm:description>
</stm:unit>

<!-- derived non-SI units -->

<stm:unit id="l" name="litre" unitType="volume"
   parentSI="meterCubed"
   abbreviation="l"
   multiplierToSI="0.001">
  <stm:description>Nearly 1 dm**3 This is not quite exact</stm:description>
</stm:unit>

<!-- ... -->

<stm:unit id="fahr" name="fahrenheit" parentSI="k"
   abbreviation="F"
  multiplierToSI="0.5555555555555555"
  constantToSI="-17.77777777777777777">
  <stm:description>An obsolescent unit of temperature still used in popular
  meteorology</stm:description>
</stm:unit>

</stm:unitList>
```

Appendix 2. The STMML Schema

```
<?xml version="1.0" encoding="utf-8"?><xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.xml-cml.org/schema/stmml" xmlns:sch="http://www.ascc.net/xml/schematron"
targetNamespace="http://www.xml-cml.org/schema/stmml" elementFormDefault="qualified">

 <xsd:annotation>
    <xsd:documentation xml:lang="en">
      <div class="curation">
      <ul>
      <li>created by hand
      2001-11-20</li>
      <li>First draft 2001-11-20</li>
      </ul>
      <p>STMML supports domain-independent STM information components</p>
    </div>
    </xsd:documentation>
  </xsd:annotation>

  <xsd:annotation>
    <xsd:appinfo>
      <sch:title>Schematron validation</sch:title>
      <sch:ns uri="http://www.xml-cml.org/schema/stmml"/>
    </xsd:appinfo>
  </xsd:annotation>

<xsd:annotation>
  <xsd:documentation>
    <div class="heading">Data Types and Data Structure</div>
  </xsd:documentation>
</xsd:annotation>

<xsd:annotation>
  <xsd:documentation>
    <div class="subheading">Overview</div>
    <div class="description">
    <p>STMML defines a number of data types suited to STM. It also defines
    a number of complex data strucures such as arrays, matrices and tables.
    the constraints are sometimes created through elements and sometimes
    through attributes. We classify the general components as follows:</p>

    <p><b>Abstract Data Structures</b></p>
    <ul>
    <li><a href="el.scalar">scalar</a>. A scalar quantity, expressible as a string, but
    with many optional facets such as errors, units, ranges, etc. Most elements
    may have <a href="st.countType">countType</a> attribute to indicate more than
    one instance.</li>
    <li><a href="el.array">array</a>. An array of homogeneous <tt>scalar</tt>s
    whose size is described by <a href="st.sizeType">sizeType</a>. <a href="st.delimiterType">
    Delimiter</a>s in string representations can ve varied.</li>
    <li><a href="el.matrix">matrix</a>. A rectangular (often square) matrix of
    homogeneous <tt>scalar</tt>s. Many matrices have special functions
    (see <a href="el.matrixType">matrixType</a>) such as geometric
    transformations</li>
    <li><a href="el.table">table</a>. An table where the columns are homogeneous
    <tt>array</tt>s. </li>
    <li><a href="el.list">list</a>. A list of heterogeneous components from any
    namespace.</li>
    <li><a href="st.sizeType">sizeType</a>. Size of arrays</li>
    <li><a href="st.delimiterType">delimiterType</a>. A lexical delimiter</li>
    </ul>

    <p><b>Links and References</b></p>
    <ul>
    <li><a href="el.link">link</a>. Support for simple hyperlinks and link structures</li>
    <li><a href="st.refType">refType</a>. A reference to an element</li>
    <li><a href="st.namespaceRefType">namespaceRefType</a>. A reference to an element, including
    namespace-like prefixes</li>
    </ul>

    <p><b>Data-based simpleTypes</b></p>
    <ul>
    <li><a href="st.coordinate2Type">coordinate2Type</a>. A 2-D coordinate</li>
    <li><a href="st.coordinate3Type">coordinate3Type</a>. A 3-D coordinate</li>
    <li><a href="st.dataTypeType">dataTypeType</a>. An enumeration of
    data types (similar to those in XML Schema).</li>
    <li><a href="st.errorBasisType">errorBasisType</a>. Basis of numeric error estimates</li>
    <li><a href="st.minType">minType</a>. Minimum value</li>
    <li><a href="st.maxType">maxType</a>. Maximum value</li>
    </ul>

    <p><b>Common attribute types</b></p>
    <ul>
    <li><a href="st.idType">idType</a>. Specifies lexical patterns for IDs</li>
    <li><a href="attGp.idGroup">idGroup</a>. ID attribute (highly encouraged)</li>
    <li><a href="attGp.titleGroup">titleGroup</a>. Title attribute (highly encouraged)</li>
    <li><a href="attGp.convGroup">convGroup</a>. Convention attribute</li>
    <li><a href="st.userTypeType">userTypeType</a>. Allows use to specify their own
    attributes</li>
    </ul>
    </div>
  </xsd:documentation>
</xsd:annotation>

<xsd:annotation>
  <xsd:documentation>
```

```
    <div class="subheading">Data structure</div>
  </xsd:documentation>
</xsd:annotation>

<xsd:element name="array" id="el.array">
   <xsd:annotation>
     <xsd:documentation>
       <div class="summary">
            A homogenous 1-dimensional array of similar objects.</div>
         <div class="description">
         <p>
            <tt>array</tt> manages a homogenous 1-dimensional array of similar objects. These
            can be encoded as strings (i.e. XSD-like datatypes) and are concatenated as
            string content. The size of the array should always be &gt;= 1.
         </p>
         <p>The default delimiter is whitespace. The <tt>normalize-space()</tt> function of
         XSLT could be used to normalize all whitespace to single spaces and this would not affect
         the value of the array elements. To extract the elements <tt>java.lang.StringTokenizer</tt>
         could be used. If the elements themselves contain whitespace then a different delimiter
         must be used and is identified through the <tt>delimiter</tt> attribute. This method is
         mandatory if it is required to represent empty strings. If a delimiter is used it MUST
         start and end the array - leading and trailing whitespace is ignored. Thus <tt>size+1</tt>
         occurrences of the delimiter character are required. If non-normalized whitespace is to be
         encoded (e.g. newlines, tabs, etc) you are recommended to translate it character-wise
         to XML character entities.
         </p>
         <p>Note that normal Schema validation tools cannot validate the elements
         of <b>array</b> (they are defined as <tt>string</tt>) However if the string is
         split, a temporary schema
         can be constructed from the type and used for validation. Also the type
         can be contained in a dictionary and software could decide to retrieve this
         and use it for validation.</p>
         <p>When the elements of the <tt>array</tt> are not simple scalars
         (e.g. <a href="el.scalar">scalar</a>s with a value and an error, the
         <tt>scalar</tt>s should be used as the elements. Although this is
         verbose, it is simple to understand. If there is a demand for
         more compact representations, it will be possible to define the
         syntax in a later version.</p>
        </div>
<div class="example">
<pre>
&lt;array size="5" title="value"
  dataType="xsd:decimal"&gt;  1.23 2.34 3.45 4.56 5.67&lt;/array&gt;

</pre>
         <p>the <tt>size</tt> attribute is not mandatory but provides a useful validity
         check): </p>
        </div>
<div class="example">
<pre>
&lt;array size="5" title="initials" dataType="xsd:string"
  delimiter="/"&gt;/A B//C/D-E/F/&lt;/array&gt;

</pre>
<p>Note that the second array-element is the empty string ''.</p>
        </div>
<div class="example">
<pre>
&lt;array title="mass" size="4"
  units="unit:g"
  errorBasis="observedStandardDeviation"
  minValues="10 11 10 9"
  maxValues="12 14 12 11"
  errorValues="1 2 1 1"
  dataType="xsd:float"&gt;11 12.5 10.9 10.2
&lt;/array&gt;

</pre>
      </div>
       </xsd:documentation>
      <xsd:appinfo>
         <sch:pattern xmlns="http://www.ascc.net/xml/schematron" name="array">
         </sch:pattern>
      </xsd:appinfo>
   </xsd:annotation>
   <xsd:complexType>
     <xsd:simpleContent>
       <xsd:extension base="xsd:string">
          <xsd:attributeGroup ref="tit_id_conv_dictGroup"/>
            <xsd:attribute name="dataType" type="dataTypeType" default="xsd:string">
               <xsd:annotation>
                  <xsd:documentation>
                   <div class="summary">all elements of the array must have the same dataType</div>
                  </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="errorValues" type="floatArrayType">
               <xsd:annotation>
                  <xsd:documentation>
                   <div class="summary">an optional array of error values for numeric arrays</div>
                  </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="errorBasis" type="errorBasisType">
               <xsd:annotation>
```

```
                  <xsd:documentation>
                   <div class="summary">A string describing the basis of the errors</div>
                  </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="minValues" type="floatArrayType">
               <xsd:annotation>
                  <xsd:documentation>
                   <div class="summary">an optional array of minimum values for numeric arrays</div>
                  </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="maxValues" type="floatArrayType">
               <xsd:annotation>
                  <xsd:documentation>
                   <div class="summary">an optional array of maximum values for numeric arrays</div>
                  </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="units" type="unitsType">
               <xsd:annotation>
                  <xsd:documentation>
                   <div class="summary">a string denoting the units (recommended for numeric quantities!!).
                   All elements must have the same units</div>
                  </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="delimiter" type="delimiterType"/>
            <xsd:attribute name="size" type="xsd:positiveInteger"/>
       </xsd:extension>
     </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>

<xsd:element name="scalar" id="el.scalar">
   <xsd:annotation>
     <xsd:documentation>
       <div class="summary">An element to hold scalar data.</div>
         <div class="description">
           <p><tt>scalar</tt> holds scalar data under a single
            generic container. The semantics are usually resolved by
            linking to a dictionary.
             <b>scalar</b> defaults to a scalar string but
             has attributes which affect the type.
         </p>
         <p><tt>scalar</tt> does not necessarily reflect a physical object (for which
         <a href="el.object">object</a> should be used). It may reflect a property of an object
         such as temperature, size, etc. </p>
         <p>Note that normal Schema validation tools cannot validate the data type
         of <b>scalar</b> (it is defined as <tt>string</tt>), but that a temporary schema
         can be constructed from the type and used for validation. Also the type
         can be contained in a dictionary and software could decide to retrieve this
         and use it for validation.</p>
      </div>

      <div class="example">
        <pre>
&lt;scalar
    dataType="xsd:decimal"
    errorValue="1.0"
    errorBasis="observedStandardDeviation"
    title="body weight"
    dictRef="zoo:bodywt"
    units="units:g"&gt;34.3&lt;/scalar&gt;

        </pre>
      </div>
     </xsd:documentation>
   </xsd:annotation>
   <xsd:complexType>
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:attributeGroup ref="tit_id_conv_dictGroup"/>
            <xsd:attribute name="dataType" type="dataTypeType" default="xsd:string"/>
            <xsd:attribute name="errorValue" type="xsd:decimal"/>
            <xsd:attribute name="errorBasis" type="errorBasisType"/>
            <xsd:attribute name="minValue" type="minType"/>
            <xsd:attribute name="maxValue" type="maxType"/>
            <xsd:attribute name="units" type="unitsType"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>

<xsd:element name="matrix" id="el.matrix">
   <xsd:annotation>
      <xsd:documentation>
        <div class="summary">A rectangular matrix of any quantities</div>
        <div class="description"><p>By default <tt>matrix</tt> represents
        a rectangular matrix of any quantities
            representable as XSD or STMML dataTypes. It consists of
            <tt>rows*columns</tt> elements, where <tt>columns</tt> is the
     fasting moving index. Assuming the elements are counted from 1 they are
     ordered <tt>V[1,1],V[1,2],...V[1,columns],V[2,1],V[2,2],...V[2,columns],
     ...V[rows,1],V[rows,2],...V[rows,columns]</tt></p>
     <p>By default whitespace is used to separate matrix elements; see
     <a href="el.array">array</a> for details. There are NO characters or markup
     delimiting the end of rows; authors must be careful!. The <tt>columns</tt>
     and <tt>rows</tt> attributes have no default values; a row vector requires
     a <tt>rows</tt> attribute of 1.</p>
     <p><tt>matrix</tt> also supports many types of square matrix, but at present we
     require all elements to be given, even if the matrix is symmetric, antisymmetric
     or banded diagonal. The <tt>matrixType</tt> attribute allows software to
     validate and process the type of matrix.</p>
      </div>
        <div class="example">
        <pre>
&lt;matrix id="m1" title="mattrix-1" dictRef="foo:bar"
  rows="3" columns="3" dataType="xsd:decimal"
  delimiter="|" matrixType="squareSymmetric" units="unit:m"
  &gt;|1.1|1.2|1.3|1.2|2.2|2.3|1.3|2.3|3.3!&lt;/matrix&gt;

        </pre>
        </div>
     </xsd:documentation>
   </xsd:annotation>

   <xsd:complexType>
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:attribute name="dataType" type="dataTypeType" default="xsd:float"/>
            <xsd:attribute name="delimiter" type="delimiterType"/>
            <xsd:attribute name="rows" type="sizeType" use="required">
               <xsd:annotation>
                  <xsd:documentation>
                     <div class="summary"><p>Number of rows</p>
                  </div>
                  </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="columns" type="sizeType" use="required">
               <xsd:annotation>
                  <xsd:documentation>
                    <div class="summary"><p>Number of columns</p>
                  </div>
                  </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="units" type="unitsType">
               <xsd:annotation>
                  <xsd:documentation>
                    <div class="summary"><p>units (recommended for numeric quantities!!)</p>
                  </div>
                  </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
            <xsd:attributeGroup ref="tit_id_conv_dictGroup"/>
            <xsd:attribute name="matrixType" type="matrixType">
              <xsd:annotation>
                 <xsd:documentation>
                  <div class="summary"><p>Type of matrix (mainly square ones)</p>
               </div>
              </xsd:documentation>
            </xsd:annotation>
         </xsd:attribute>
           <xsd:attribute name="errorValues" type="floatArrayType">
              <xsd:annotation>
                 <xsd:documentation>
                  <div class="summary">an optional array of error values for numeric matrices</div>
                 </xsd:documentation>
              </xsd:annotation>
           </xsd:attribute>
           <xsd:attribute name="errorBasis" type="errorBasisType">
              <xsd:annotation>
                 <xsd:documentation>
                  <div class="summary">A string describing the basis of the errors</div>
                 </xsd:documentation>
              </xsd:annotation>
           </xsd:attribute>
           <xsd:attribute name="minValues" type="floatArrayType">
              <xsd:annotation>
                 <xsd:documentation>
                  <div class="summary">an optional array of minimum values for numeric matrices</div>
                 </xsd:documentation>
              </xsd:annotation>
           </xsd:attribute>
           <xsd:attribute name="maxValues" type="floatArrayType">
              <xsd:annotation>
                 <xsd:documentation>
                  <div class="summary">an optional array of maximum values for numeric matrices</div>
                 </xsd:documentation>
              </xsd:annotation>
           </xsd:attribute>

         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>

   <xsd:element name="table" id="el.table">
```

```
      <xsd:annotation>
         <xsd:documentation>
            <div class="summary">A rectangular table of any quantities</div>
            <div class="description"><p>By default <tt>table</tt>
            represents a rectangular table of any quantities
               representable as XSD or STMML dataTypes. The default layout is columnwise,
               with <tt>columns</tt> columns,
               where each column is a (homogeneous) <a href="el.array">array</a> of
               size <tt>rows</tt> data. This is the "normal" orientation of data tables
               but the table display could be transposed by XSLT transformation if required.
                    Access is to columns, and thence to the data within them. DataTyping, delimiters,
                    etc are delegated to the arrays, which must all be of the same size. For
                    verification it is recommended that every array carries a <tt>size</tt>
                    attribute.
               </p>
         </div>
         <div class="example"><pre>&lt;table rows="3" columns="2" title="people"&gt;
  &lt;array title="age" dataType="xsd:integer"&gt;3 5 7&lt;/array&gt;
  &lt;array title="name" dataType="xsd:string"&gt;Sue Fred Sandy&lt;/array&gt;
&lt;/table&gt;
</pre></div>
        </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
         <xsd:sequence>
           <xsd:element ref="array" maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:attribute name="rows" type="sizeType" use="required">
            <xsd:annotation>
               <xsd:documentation>
                 <div class="description"><p>Number of rows</p>
               </div>
              </xsd:documentation>
            </xsd:annotation>
         </xsd:attribute>
         <xsd:attribute name="columns" type="sizeType" use="required">
            <xsd:annotation>
               <xsd:documentation>
                  <div class="description"><p>Number of columns</p>
               </div>
              </xsd:documentation>
            </xsd:annotation>
         </xsd:attribute>
         <xsd:attributeGroup ref="tit_id_conv_dictGroup"/>
      </xsd:complexType>
   </xsd:element>

<xsd:element name="list" id="el.list">
  <xsd:annotation>
    <xsd:documentation>
      <div class="summary">A generic container with no implied semantics</div>
      <div class="description"><p>
       A generic container with no implied semantics. It just contains
      things and can have attributes which bind conventions to it. It could often
      act as the root element in an STM document.</p>
      </div>
      <div class="example">
      <pre>
&lt;list&gt;
  &lt;array title="animals"&gt;frog bear toad&lt;/array&gt;
  &lt;scalar title="weight" dataType="xsd:float"&gt;3.456&lt;/scalar&gt;
&lt;/list&gt;

      </pre>
      </div>
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence>
      <xsd:any minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="tit_id_conv_dictGroup"/>
    <xsd:attribute name="type" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>


<xsd:annotation>
  <xsd:documentation>
    <div class="subheading">Data-based simpleTypes</div>
  </xsd:documentation>
</xsd:annotation>

<xsd:simpleType name="coordinate2Type" id="st.coordinate2Type">
   <xsd:annotation>
     <xsd:documentation>
        <div class="summary">An x/y coordinate pair</div>
         <div class="description">
         <p>An x/y coordinate pair consisting of
         two real numbers, separated by whitespace or a comma.
         In arrays and matrices, it may be useful to set a separate delimiter</p>
      </div>
      <div class="example">
        <pre>
```

```
&lt;list&gt;
  &lt;array
      &gt;1.2,3.4   3.2,4.5   6.7,23.1 &lt;/array&gt;
  &lt;array delimiter="/"
      &gt;/1.2 3.4/3.2 4.5/6.7 23.1/&lt;/array&gt;
&lt;/list&gt;

        </pre>
       </div>
     </xsd:documentation>
   </xsd:annotation>
   <xsd:restriction base="xsd:string">
      <xsd:pattern value="\s*([-]|[+])?\d*\.?\d*(\s+|[,])([-]|[+])?\d*\.?\d*\s*"/>
   </xsd:restriction>
</xsd:simpleType>

   <xsd:simpleType name="coordinate3Type" id="st.coordinate3Type">
      <xsd:annotation>
        <xsd:documentation>
          <div class="summary">An x/y/z coordinate triple</div>

          <div class="description">
            <p>An x/y/z coordinate triple consisting of
            three real numbers, separated by whitespace or commas.
            In arrays and matrices, it may be useful to set a separate delimiter</p>
         </div>

         <div class="example">
           <pre>
&lt;list&gt;
  &lt;array&gt;1.2,3.4,1.2
    3.2,4.5,7.3   6.7,23.1,5.6 &lt;/array&gt;
  &lt;array delimiter="/"
  &gt;/1.2 3.4 3.3/3.2 4.5 4.5/6.7 23.1 5.6/&lt;/array&gt;
&lt;/list&gt;

           </pre>
         </div>
        </xsd:documentation>
      </xsd:annotation>
      <xsd:restriction base="xsd:string">
         <xsd:pattern value="\s*([-]|[+])?\d*\.?\d*(\s+|[,])([-]|[+])?\d*\.?\d*(\s+|[,])([-]|[+])?\d*\.?\d*\s*"/>

      </xsd:restriction>
   </xsd:simpleType>


   <xsd:simpleType name="dataTypeType" id="st.dataTypeType">
      <xsd:annotation>
        <xsd:documentation>
          <div class="summary">
           <p>an enumerated type for all builtin allowed dataTypes in STM</p>
         </div>

         <div class="description">
            <p>
               <tt>dataTypeType</tt> represents an enumeration of allowed dataTypes
               (at present identical with those in XML-Schemas (Part2- datatypes).
               This means that implementers should be able to use standard XMLSchema-based
               tools for validation without major implementation problems.
            </p>
            <p>It will often be used an an attribute on
            <a href="el.scalar">scalar</a>,
            <a href="el.array">array</a> or
            <a href="el.matrix">matrix</a>
            elements.</p>
         </div>

         <div class="example">
           <pre>
&lt;list xmlns="http://www.xml-cml.org/schema/core"&gt;
  &lt;scalar dataType="xsd:boolean" title="she loves me"&gt;true&lt;/scalar&gt;
  &lt;scalar dataType="xsd:float" title="x"&gt;23.2&lt;/scalar&gt;
  &lt;scalar dataType="xsd:duration" title="egg timer"&gt;PM4&lt;/scalar&gt;
  &lt;scalar dataType="xsd:dateTime" title="current data and time"&gt;2001-02-01:00:30&lt;/scalar&gt;
  &lt;scalar dataType="xsd:time" title="wake up"&gt;06:00&lt;/scalar&gt;
  &lt;scalar dataType="xsd:date" title="where is it"&gt;1752-09-10&lt;/scalar&gt;
  &lt;scalar dataType="xsd:anyURI" title="CML site"&gt;http://www.xml-cml.org/&lt;/scalar&gt;
  &lt;scalar dataType="xsd:QName" title="CML atom"&gt;cml:atom&lt;/scalar&gt;
  &lt;scalar dataType="xsd:normalizedString" title="song"&gt;the mouse ran up the clock&lt;/scalar&gt;
  &lt;scalar dataType="xsd:language" title="UK English"&gt;en-GB&lt;/scalar&gt;
  &lt;scalar dataType="xsd:Name" title="atom"&gt;atom&lt;/scalar&gt;
  &lt;scalar dataType="xsd:ID" title="XML ID"&gt;_123&lt;/scalar&gt;
  &lt;scalar dataType="xsd:integer" title="the answer"&gt;42&lt;/scalar&gt;
  &lt;scalar dataType="xsd:nonPositiveInteger" title="zero"&gt;0&lt;/scalar&gt;
&lt;/list&gt;

           </pre>
         </div>
        </xsd:documentation>
      </xsd:annotation>
      <xsd:restriction base="xsd:string">
       <xsd:enumeration value="xsd:string"/>
       <xsd:enumeration value="xsd:boolean"/>
       <xsd:enumeration value="xsd:float"/>
       <xsd:enumeration value="xsd:double"/>
       <xsd:enumeration value="xsd:decimal"/>
       <xsd:enumeration value="xsd:duration"/>
       <xsd:enumeration value="xsd:dateTime"/>
       <xsd:enumeration value="xsd:time"/>
       <xsd:enumeration value="xsd:date"/>
       <xsd:enumeration value="xsd:gYearMonth"/>
       <xsd:enumeration value="xsd:gYear"/>
       <xsd:enumeration value="xsd:gMonthDay"/>
       <xsd:enumeration value="xsd:gDay"/>
       <xsd:enumeration value="xsd:gMonth"/>
       <xsd:enumeration value="xsd:hexBinary"/>
       <xsd:enumeration value="xsd:base64Binary"/>
       <xsd:enumeration value="xsd:anyURI"/>
       <xsd:enumeration value="xsd:QName"/>
       <xsd:enumeration value="xsd:NOTATION"/>
       <xsd:enumeration value="xsd:normalizedString"/>
       <xsd:enumeration value="xsd:token"/>
       <xsd:enumeration value="xsd:language"/>
       <xsd:enumeration value="xsd:IDREFS"/>
       <xsd:enumeration value="xsd:ENTITIES"/>
       <xsd:enumeration value="xsd:NMTOKEN"/>
       <xsd:enumeration value="xsd:NMTOKENS"/>
       <xsd:enumeration value="xsd:Name"/>
       <xsd:enumeration value="xsd:NCName"/>
       <xsd:enumeration value="xsd:ID"/>
       <xsd:enumeration value="xsd:IDREF"/>
       <xsd:enumeration value="xsd:ENTITY"/>
       <xsd:enumeration value="xsd:integer"/>
       <xsd:enumeration value="xsd:nonPositiveInteger"/>
       <xsd:enumeration value="xsd:negativeInteger"/>
       <xsd:enumeration value="xsd:long"/>
       <xsd:enumeration value="xsd:int"/>
       <xsd:enumeration value="xsd:short"/>
       <xsd:enumeration value="xsd:byte"/>
       <xsd:enumeration value="xsd:nonNegativeInteger"/>
       <xsd:enumeration value="xsd:unsignedLong"/>
       <xsd:enumeration value="xsd:unsignedInt"/>
       <xsd:enumeration value="xsd:unsignedShort"/>
       <xsd:enumeration value="xsd:unsignedByte"/>
       <xsd:enumeration value="xsd:positiveInteger"/>
     </xsd:restriction>
 </xsd:simpleType>

<xsd:annotation>
  <xsd:documentation>
    <div class="summary">Array-based simpleTypes</div>
  </xsd:documentation>
</xsd:annotation>

   <xsd:simpleType name="sizeType" id="st.sizeType">
      <xsd:annotation>
         <xsd:documentation>
           <div class="summary">The size of an array</div>
           <div class="description">
            <p>The size of an array. Redundant, but serves as a check
         for processing software (useful if delimiters are used)</p>
         </div>
       </xsd:documentation>
      </xsd:annotation>
      <xsd:restriction base="xsd:positiveInteger"/>
   </xsd:simpleType>

<xsd:simpleType name="delimiterType" id="st.delimiterType">
   <xsd:annotation>
     <xsd:documentation>
       <div class="summary">A non-whitespace character used in arrays to separate components</div>
      <div class="description">
        <p>Some STMML elements (such as <a href="el.array">array</a>) have
        content representing concatenated values. The default separator is
        whitespace (which can be normalised) and this should be used whenever
        possible. However in some cases the values are empty, or contain whitespace or other
        problematic punctuation, and a delimiter is required.</p>
        <p>Note that the content string MUST start and end with the delimiter so
        there is no ambiguity as to what the components are. Only printable
        characters from the ASCII character set should be used, and character
        entities should be avoided.</p>
        <p>When delimiters are used to separate precise whitespace this should always
        consist of spaces and not the other allowed whitespace characters
        (newline, tabs, etc.). If the latter are important it is probably best to redesign
        the application.</p>
      </div>
      <div class="example">
        <pre>
&lt;array size="4" delimiter="|"&gt;|A|B12||D and   E|&lt;/array&gt;

 <em>The values in the array are</em>
  "A", "B12", "" (empty string) and "D and   E"
 <em>note the spaces</em>
        </pre>
      </div>
     </xsd:documentation>
   </xsd:annotation>
   <xsd:restriction base="xsd:string">
   </xsd:restriction>
```

```
</xsd:simpleType>

   <xsd:simpleType name="floatArrayType" id="st.floatArrayType">
     <xsd:annotation>
       <xsd:documentation>
         <div class="summary">An array of floats</div>

         <div class="description">
            <p>An array of floats or other real numbers.
            Not used in STM Schema, but re-used by CML.
            There is no schema-checking of values, unfortunately.</p>
         </div>

        <div class="example">
           <pre>
&lt;atomArray xmlns="http://www.xml-cml.org/schema/core"
  x2="1.2 2.3 3.4 5.6"/&gt;

           </pre>
         </div>
        </xsd:documentation>
      </xsd:annotation>
      <xsd:restriction base="xsd:string">
      </xsd:restriction>
   </xsd:simpleType>

<xsd:simpleType name="matrixType" id="st.matrixType">
  <xsd:annotation>
    <xsd:documentation>
      <div class="summary">Allowed <a href="el.matrix">matrix</a> types</div>
       <div class="description"><p>Allowed <tt>matrix</tt> types.
       These are mainly square matrices</p></div>
      <div class="example">
      <pre>
&lt;matrix id="m1" title="mattrix-1" dictRef="foo:bar"
  rows="3" columns="3" dataType="xsd:decimal"
  delimiter="|" matrixType="squareSymmetric" units="unit:m"
  &gt;|1.1|1.2|1.3|1.2|2.2|2.3|1.3|2.3|3.3!&lt;/matrix&gt;

      </pre>
      </div>
    </xsd:documentation>
  </xsd:annotation>
  <xsd:restriction base="xsd:string">
      <xsd:enumeration value="rectangular"/>
      <xsd:enumeration value="square"/>
      <xsd:enumeration value="squareSymmetric"/>
      <xsd:enumeration value="squareAntisymmetric"/>
      <xsd:enumeration value="diagonal">
        <xsd:annotation>
          <xsd:documentation>
            <div class="description">Symmetric. Elements are zero except on the diagonal</div>
          </xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>
      <xsd:enumeration value="upperTriangular">
        <xsd:annotation>
          <xsd:documentation>
            <div class="description">Square. Elements are zero below the diagonal
            <pre>
1 2 3 4
0 3 5 6
0 0 4 8
0 0 0 2
      </pre>
            </div>
          </xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>
      <xsd:enumeration value="lowerTriangular">
        <xsd:annotation>
          <xsd:documentation>
            <div class="description">Symmetric. Elements are zero except on the diagonal</div>
          </xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>
      <xsd:enumeration value="unitary"/>
      <xsd:enumeration value="rowEigenvectors"/>
      <xsd:enumeration value="rotation22"/>
      <xsd:enumeration value="rotationTranslation32"/>
      <xsd:enumeration value="homogeneous33"/>
      <xsd:enumeration value="rotation33"/>
      <xsd:enumeration value="rotationTranslation43"/>
      <xsd:enumeration value="homogeneous44"/>
      <xsd:enumeration value="square"/>
      <xsd:enumeration value="square"/>
   </xsd:restriction>
</xsd:simpleType>

   <xsd:simpleType name="countType" id="st.countType">
      <xsd:annotation>
        <xsd:documentation>
           <div class="summary">
            <p>A count multiplier for certain elements</p>
         </div>
```

```
          <div class="description">
            <p>A count multiplier for certain elements, such as
            <tt>action</tt> or <tt>object</tt>.</p>
         </div>

         <div class="example">
           <pre>
&lt;list&gt;
&lt;object title="frog" count="10"/&gt;
&lt;action title="step3" count="3"&gt;
  &lt;p&gt;Add 10 ml reagent&lt;/p&gt;
&lt;/action&gt;
&lt;/list&gt;

           </pre>
         </div>
        </xsd:documentation>
      </xsd:annotation>
      <xsd:restriction base="xsd:integer">
        <xsd:minInclusive value="1"/>
        <xsd:maxInclusive value="999999999999"/>
      </xsd:restriction>
   </xsd:simpleType>

<xsd:annotation>
  <xsd:documentation>
    <div class="summary">General simpleTypes</div>
  </xsd:documentation>
</xsd:annotation>

<xsd:simpleType name="idType" id="st.idType">
  <xsd:annotation>
    <xsd:documentation>
      <div class="summary">A unique ID for STM elements</div>
      <div class="description">
      <p>A unique ID for STM elements. This is not formally
          of type ID (an XML NAME which must start with a letter and contain only letters,
          digits and <tt>.-_:</tt>). It is recommended that IDs start with a letter,
          and contain no punctuation or whitespace. The function <tt>generate-id()</tt>
          in XSLT will generate semantically void unique IDs</p>.
          <p>It is difficult to ensure uniqueness when documents are merged. We suggest
          namespacing IDs, perhaps using the containing elements as the base.
          Thus <tt>mol3:a1</tt> could be a useful unique ID.
          However this is still experimental.</p>
         </div>
      </xsd:documentation>
     </xsd:annotation>
   <xsd:restriction base="xsd:string">
     <xsd:pattern value="[A-Za-z0-9_-]+(:[A-Za-z0-9_-]+)?"/>
   </xsd:restriction>
</xsd:simpleType>
   <xsd:simpleType name="userTypeType" id="st.userTypeType">
      <xsd:annotation>
         <xsd:documentation>
            <div class="summary">An uncontrolled vocabulary from a user</div>
            <div class="description"><p>
               An element or attribute representing some part of
               an uncontrolled vocabulary. It will be ignored if a <tt>type</tt>
               attribute is also set on the element. The <tt>type</tt> attribute
               will normally have an enumerated set of values.
            </p>
         </div>
         <div class="example"><pre>&lt;alternative type="userType" userType="biological name"&gt;SusScrofa&lt;/alternative&gt;
</pre></div>
         </xsd:documentation>
     </xsd:annotation>
     <xsd:restriction base="xsd:string">
     </xsd:restriction>
</xsd:simpleType>

   <xsd:attributeGroup name="idGroup" id="attGp.id">
      <xsd:attribute id="att.id" name="id" type="idType">
         <xsd:annotation>
            <xsd:documentation>
              <div class="summary">A attribute prividing a unique ID for STM elements</div>
             <div class="description">
             <p>See <a href="st.idType">idType</a> for full documentation.</p>
           </div>
           </xsd:documentation>
         </xsd:annotation>
      </xsd:attribute>
   </xsd:attributeGroup>

<xsd:attributeGroup name="titleGroup" id="attGp.title">
  <xsd:attribute id="att.title" name="title" type="xsd:string">
    <xsd:annotation>
      <xsd:documentation>
        <div class="summary">An (optional) title on most STM elements. Uncontrolled</div>
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
</xsd:attributeGroup>

<xsd:attributeGroup name="convGroup" id="attGp.conv">
```

```
<xsd:attribute id="att.convention" name="convention" type="namespaceRefType" default="CML">
  <xsd:annotation>
    <xsd:documentation>
      <div class="summary">A reference to a convention</div>
      <div class="description">
       <p>A reference to a convention which is inherited by all the subelements.</p>
       <p>It may be useful to create conventions with namespaces,
         </p>
         <p>this attribute is inherited by its child elements; thus a <tt>molecule</tt>
    with a convention sets the default for its bonds and atoms. This can be overwritten
    if necessary by an explicit <tt>convention</tt>.</p>
    <p>Use of convention will normally require non-STMML semantics, and should be used with
    caution. We would expect that conventions prefixed with "ISO" would be useful,
    such as ISO8601 for dateTimes.</p>
   </div>

   <div class="example">
     <pre>
&lt;bond convention="fooChem" order="-5"
   xmlns:fooChem="http://www.fooChem/conventions"/&gt;

     </pre>
   </div>
   </xsd:documentation>
  </xsd:annotation>
</xsd:attribute>
</xsd:attributeGroup>

<xsd:annotation>
  <xsd:documentation>
    <div class="summary">Numeric types</div>
  </xsd:documentation>
</xsd:annotation>

   <xsd:simpleType name="errorBasisType" id="st.errorBasisType">
      <xsd:annotation>
        <xsd:documentation>
          <div class="summary">The basis of an error value</div>
          <div class="description">
         <p>Errors in values can be of several types and this simpleType
         provides a small controlled vocabulary</p>
         </div>

         <div class="example">
           <pre>
&lt;scalar
    dataType="xsd:decimal"
    errorValue="1.0"
    errorBasis="observedStandardDeviation"
    title="body weight"
    dictRef="zoo:bodywt"
    units="units:g"&gt;34.3&lt;/scalar&gt;

           </pre>
         </div>
        </xsd:documentation>
      </xsd:annotation>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="observedRange"/>
        <xsd:enumeration value="observedStandardDeviation"/>
        <xsd:enumeration value="observedStandardError"/>
        <xsd:enumeration value="estimatedStandardDeviation"/>
        <xsd:enumeration value="estimatedStandardError"/>
      </xsd:restriction>
   </xsd:simpleType>

 <xsd:simpleType name="maxType" id="st.maxType">
    <xsd:annotation>
       <xsd:documentation>
          <div class="summary">The maximum INCLUSIVE value of a quantity</div>
          <div class="description"><p>The maximum INCLUSIVE value of a sortable quantity such as
          numeric, date or string. It should be ignored for dataTypes such as URL.
          The use of <tt>min</tt> and
        <tt>max</tt> attributes can be used to give a range for the quantity.
       The statistical basis of this range is not defined. The value of <tt>max</tt>
       is usually an observed
       quantity (or calculated from observations). To restrict a value, the <tt>
       maxExclusive</tt> type in a dictionary should be used.</p>
       <p>The type of the maximum is the same as the quantity to which it refers - numeric,
       date and string are currently allowed</p>
       </div>
       <div class="example"><pre>&lt;scalar dataType="xsd:float" maxValue="20" minValue="12"&gt;15&lt;/scalar&gt;
</pre></div>
      </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
    </xsd:restriction>
 </xsd:simpleType>

   <xsd:simpleType name="minType" id="st.minType">
      <xsd:annotation>
         <xsd:documentation>
            <div class="summary">The minimum INCLUSIVE value of a quantity</div>
            <div class="description"><p>The minimum INCLUSIVE value of a sortable quantity such as
            numeric, date or string. It should be ignored for dataTypes such as URL.
            The use of <tt>min</tt> and
          <tt>min</tt> attributes can be used to give a range for the quantity.
         The statistical basis of this range is not defined. The value of <tt>min</tt>
         is usually an observed
         quantity (or calculated from observations). To restrict a value, the <tt>
         minExclusive</tt> type in a dictionary should be used.</p>
         <p>The type of the minimum is the same as the quantity to which it refers - numeric,
         date and string are currently allowed</p>
         </div>
          <div class="example"><pre>&lt;scalar dataType="xsd:float" maxValue="20" minValue="12"&gt;15&lt;/scalar&gt;
</pre></div>
        </xsd:documentation>
      </xsd:annotation>
      <xsd:restriction base="xsd:string">
      </xsd:restriction>
   </xsd:simpleType>


<xsd:annotation>
  <xsd:documentation>
    <div class="subheading">Links and References</div>
  </xsd:documentation>
</xsd:annotation>

<xsd:element name="link" id="el.link">
   <xsd:annotation>
     <xsd:documentation>
      <div class="summary">An internal or external link to STMML or other object(s)</div>
      <div class="description">
         <p><tt>link</tt> is an internal or external link to STMML or other object(s). </p>
         <p><b>Semantics are similar to XLink, but simpler and only a subset is implemented.</b>
         This is intended to make the instances easy to create and read, and software
         relatively easy to implement. The architecture is:</p>
         <ul>
        <li><b>A single element (<tt>link</tt>) used for all linking purposes</b>. </li>
        <li><b>The link types are determined by the <tt>type</tt> attribute and can be:</b>.
        <ul>
        <li><b>locator</b>. This points to a single target and must carry
        either a <tt>ref</tt> or <tt>href</tt> attribute.
        <tt>locator</tt> links are usually children of an extended link.
        <li><b>arc</b>. This is a 1:1 link with both ends (<tt>from</tt> and <tt>to</tt>) defined.</li>
        <li><b>extended</b>. This is usually a parent of several locator links and serves
        to create a grouping of link ends (i.e. a list of references in documents).</li>
        Many-many links can be built up from arcs linking extended elements</li>
        </ul>
        <p>All links can have optional <tt>role</tt> attributes. The semantics of this are not defined;
        you are encouraged to use a URI as described in the XLink specification.</p>
        <p>There are two address spaces: </p>
        <ul>
        <li>The <tt>href</tt> attribute on locators behaves in the same way as <tt>href</tt> in
        HTML and is of type <tt>xsd:anyURI</tt>. Its primary use is to use XPointer to reference
        elements outside the document.</li>
        <li>The <tt>ref</tt> attribute on locators and the <tt>from</tt> and <tt>to</tt>
        attributes on <tt>arc</tt>s refer to IDs (<em>without</em> the '#' syntax).</li>
        </ul>

        <p>Note: several other specific linking mechanisms are defined elsewhere in STM.
        <a href="el.relatedEntry">relatedEntry</a>
         should be used in dictionaries, and <a href="st.dictRef">dictRef</a>
        should be used to link to dictionaries. There are no required uses of <tt>link</tt> in STMML
        but we have used it to map atoms, electrons and bonds in reactions in CML</p>
        </li>
        </ul>
                  <p><b>Relation to XLink</b>.
         At present (2002) we are not aware of generic XLink
         processors from which we would benefit, so the complete implementation brings little
         extra value.
         Among the simplifications from Xlink are:</p>
         <ul>
         <li><tt>type</tt> supports only <tt>extended</tt>, <tt>locator</tt> and <tt>arc</tt></li>
         <li><tt>label</tt> is not supported and <tt>id</tt>s are used as targets of links.</li>
         <li><tt>show</tt> and <tt>actuate</tt> are not supported.</li>
         <li><tt>xlink:title</tt> is not supported (all STM elements can have a <tt>title</tt>
         attribute).</li>
         <li><tt>xlink:role</tt> supports any string (i.e. does not have to be a namespaced resource).
         This mechanism can, of course, still be used and we shall promote it where STM
         benefits from it</li>
         <li>The <tt>to</tt> and <tt>from</tt> attributes point to IDs rather than labels</li>
         <li>The xlink namespace is not used</li>
         <li>It is not intended to create independent linkbases, although some collections of
         links may have this property and stand outside the documents they link to</li>
         </ul>
      </div>
     </xsd:documentation>
   </xsd:annotation>
 <xsd:complexType>
    <xsd:sequence>
       <xsd:any minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="tit_id_conv_dictGroup"/>
    <xsd:attribute name="from" type="namespaceRefType">
      <xsd:annotation>
        <xsd:documentation>
          <div class="summary">The starting point of an arc</div>
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="to" type="namespaceRefType">
      <xsd:annotation>
        <xsd:documentation>
          <div class="summary">The endpoint of an arc</div>
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="ref" type="namespaceRefType">
      <xsd:annotation>
        <xsd:documentation>
          <div class="summary">An ID referenced within a locator</div>
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="role" type="xsd:string">
      <xsd:annotation>
        <xsd:documentation>
          <div class="summary">The role of the link. Xlink adds semantics through a
          URI; we shall not be this strict. We shall not normally use this mechanism
          and use dictionaries instead</div>
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="href" type="xsd:anyURI">
      <xsd:annotation>
        <xsd:documentation>
          <div class="summary">The target of the (locator) link, outside the document</div>
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="type">
      <xsd:annotation>
        <xsd:documentation>
          <div class="summary">The type of the link</div>
        </xsd:documentation>
      </xsd:annotation>
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="extended">
            <xsd:annotation>
              <xsd:documentation>
                <div class="summary">A container for locators</div>
              </xsd:documentation>
            </xsd:annotation>
          </xsd:enumeration>
          <xsd:enumeration value="locator">
            <xsd:annotation>
              <xsd:documentation>
                <div class="summary">A link to an element</div>
              </xsd:documentation>
            </xsd:annotation>
          </xsd:enumeration>
          <xsd:enumeration value="arc">
            <xsd:annotation>
              <xsd:documentation>
                <div class="summary">A labelled link</div>
              </xsd:documentation>
            </xsd:annotation>
          </xsd:enumeration>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
</xsd:element>

<xsd:annotation>
  <xsd:documentation>
    <div class="summary">References</div>
  </xsd:documentation>
</xsd:annotation>

<xsd:simpleType name="namespaceRefType" id="st.namespaceRefType">
   <xsd:annotation>
      <xsd:documentation>
        <div class="summary">
         A string referencing a dictionary, units, convention or other metadata.
      </div>
        <div class="description">
         <p>A string referencing a dictionary, units, convention or other
      metadata. The namespace is optional but recommended where possible</p>
      <p>Note: this convention is only used within STMML and related languages;
      it is NOT a URI.</p>
      </div>
        <div class="example">
        <pre>
&lt;list&gt;
&lt;!-- dictRef is of namespaceRefType --&gt;
  &lt;scalar dictRef="chem:mpt"&gt;123&lt;/scalar&gt;
&lt;!-- error --&gt;
  &lt;scalar dictRef="mpt23"&gt;123&lt;/scalar&gt;
&lt;/list&gt;

        </pre>
      </div>
     </xsd:documentation>
   </xsd:annotation>
   <xsd:restriction base="xsd:string">
      <xsd:pattern value="[A-Za-z][A-Za-z0-9_]*(:[A-Za-z][A-Za-z0-9_]*)?"/>
   </xsd:restriction>
</xsd:simpleType>

   <xsd:simpleType name="refType" id="st.refType">
      <xsd:annotation>
         <xsd:documentation>
           <div class="summary">A reference to an existing element</div>
           <div class="description"><p>A reference to an existing element in the document.
            The target of the
            ref attribute (which may be namepsaced) must exist. The test for validity
            will normally occur in the element's <tt>appinfo</tt>
            </p>
            <p>Any DOM Node created from this element will normally be a <i>reference</i>
            to another Node, so that if the target node is modified a the dereferenced
            content is modified. At present there are no deep copy semantics hardcoded
            into the schema.</p>
         </div>
        </xsd:documentation>
       </xsd:annotation>
      <xsd:restriction base="idType"/>
   </xsd:simpleType>

<xsd:annotation>
  <xsd:documentation>
    <div class="heading">General information components</div>
  </xsd:documentation>
</xsd:annotation>

<xsd:annotation>
  <xsd:documentation>
    <div class="heading">General components</div>
    <div class="description">
    <p>STMML provides a very small number of abstract elements to capture
    frequently encountered concepts in STM documents. There are no predetermined
    semantics or ontology; it is expected that descriptive metadata
    will be added through dictionaries.
    </p>
    <p>All elements can contain any element children and can carry the common
    STM attributes. Currently there are the following:</p>
    <ul>
    <li><a href="el.object">object</a>. Almost anything - concrete, abstract, representable by
    a noun. Objects can have properties added through <a href="el.scalar">scalar</a>, etc.
    </li>
    <li><a href="el.action">action</a>. Represents an action performed during a scientific narrative.
    It has attributes describing a time-line and conditions so that a procedure could be replayed.
    It has a container <a href="el.actionList">actionList</a> which shares these attributes
    and which can describe sets of actions.
    </li>
    <li><a href="el.observation">observation</a>. Contains narrative or other elements describing
    an observation, planned or unplanned</li>
    </ul>
    </div>
  </xsd:documentation>
</xsd:annotation>

<xsd:element name="action" id="el.action">
  <xsd:annotation>
    <xsd:documentation>
       <div class="summary">An action which might occur in scientific data or narrative.</div>
      <div class="description">
      <p>An action which might occur in scientific data or narrative.
      The definition is deliberately vague, intending to collect examples of
      possible usage. Thus an action could be addition of materials,
      measurement, application of heat or radiation.
      The content model is unrestricted. <tt>action</tt> iself is
      normally a child of <a href="#el.actionList">actionList</a></p>
      <p>The start, end and duration attributes should be interpreted as
      </p>
      <ul>
      <li>XSD dateTimes and XSD durations. This allows precise recording of
      time of day, etc, or duration
      after start of actionList. A <tt>convention="xsd"</tt> attribute should be used
      to enforce XSD.</li>
      <li>a numerical value, with a units attribute linked to a dictionary.</li>
      <li>a human-readable string (unlikely to be machine processable)</li>
      </ul>
      <p>
      <tt>startCondition</tt> and <tt>endCondition</tt>
      values are not constrained, which allows XSL-like <tt>test</tt> attribute values.
      The semantics of the conditions are yet to be defined and at present are simply
      human readable.
      </p>
      <p>The order of the <tt>action</tt> elements in the document may,
       but will not always, define the order that they actually occur in.</p>
      <p>A delay can be shown by an <tt>action</tt> with no content. Repeated actions or
      actionLists are indicated through the count attribute.</p>
       </div>

      <div class="example">
       <pre>
```

```
&lt;actionList title="boiling two eggs for breakfast"&gt;
  &lt;!-- start cooking at 9am --&gt;
  &lt;action title="turn on heat" start="T09:00:00" convention="xsd"/&gt;
  &lt;!-- human readable description of time to start action --&gt;
  &lt;action title="put egg into pan" startCondition="water is boiling" count="2"/&gt;
  &lt;!-- the duration is expressed in ISO8601 format --&gt;
  &lt;action title="boil eggs for 4 minutes" duration="4" units="units:min"/&gt;
  &lt;!-- action immediately follows last action --&gt;
  &lt;action title="remove egg from pan" count="1"/&gt;
  &lt;action title="boil second egg for a bit longer" duration="about half a minute"/&gt;
  &lt;!-- action immediately follows last action --&gt;
  &lt;action title="remove egg from pan" count="1"/&gt;
&lt;/actionList&gt;

         </pre>
       </div>

       <div class="example">
         <pre>
&lt;actionList title="preparation of silanols"&gt;
  &lt;p&gt;This is a conversion of a chemical synthesis to STMML. We
  have deliberately not marked up the chemistry in this example!&lt;/p&gt;
  &lt;action title="step2"&gt;
    &lt;p&gt;Take 1 mmol of the diol and dissolve in dioxan in
      &lt;object title="flask"&gt;
        &lt;scalar title="volume" units="units:ml"&gt;25&lt;/scalar&gt;
      &lt;/object&gt;
    &lt;/p&gt;
  &lt;/action&gt;
  &lt;action title="step2"&gt;
    &lt;p&gt;Place flask in water bath with magnetic stirrer&lt;/p&gt;
  &lt;/action&gt;
  &lt;!-- wait until certain condition --&gt;
  &lt;actionList endCondition="bath temperature stabilised"/&gt;
  &lt;action title="step3"&gt;
    &lt;p&gt;Add 0.5 ml 1 M H2SO4&lt;/p&gt;
  &lt;/action&gt;

  &lt;!-- carry out reaction --&gt;
  &lt;actionList endCondition="reaction complete; no diol spot remains on TLC"&gt;
    &lt;actionList title="check tlc"&gt;
      &lt;!-- wait for half an hour --&gt;
      &lt;action duration="half an hour"/&gt;
      &lt;action title="tlc"&gt;
        &lt;p&gt;extract solution and check diol spot on TLC&lt;/p&gt;
      &lt;/action&gt;
    &lt;/actionList&gt;
  &lt;/actionList&gt;

  &lt;!-- work up reaction --&gt;
  &lt;action title="step5"&gt;
    &lt;p&gt;Add 10 ml water to flask&lt;/p&gt;
  &lt;/action&gt;
  &lt;action title="step6"&gt;
    &lt;p&gt;Neutralize acid with 10% NaHCO3&lt;/p&gt;
  &lt;/action&gt;
  &lt;action title="step7" count="3"&gt;
    &lt;p&gt;Extract with 10ml ether&lt;/p&gt;
  &lt;/action&gt;
  &lt;action title="step8"&gt;
    &lt;p&gt;Combine ether layers&lt;/p&gt;
  &lt;/action&gt;
  &lt;action title="step9" count="2"&gt;
    &lt;p&gt;Wash ether with 10 ml water&lt;/p&gt;
  &lt;/action&gt;
  &lt;action title="step10"&gt;
    &lt;p&gt;Wash ether with 10 ml saturated NaCl&lt;/p&gt;
  &lt;/action&gt;
  &lt;action title="step11"&gt;
    &lt;p&gt;Dry over anhydrous Na2SO4 and remove solvent on rotary evaporator&lt;/p&gt;
  &lt;/action&gt;
&lt;/actionList&gt;

         </pre>
       </div>
      </xsd:documentation>
    </xsd:annotation>

  <xsd:complexType mixed="true">
    <xsd:sequence minOccurs="0" maxOccurs="unbounded">
      <xsd:any processContents="lax"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="tit_id_conv_dictGroup"/>
    <xsd:attributeGroup ref="actionGroup"/>
    <xsd:attribute name="type" type="xsd:string">
      <xsd:annotation>
        <xsd:documentation>
          <div class="summary">The type of the action; semantics are not controlled.</div>
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
  </xsd:complexType>

</xsd:element>
```

```
  <xsd:element name="actionList" id="el.actionList">
    <xsd:annotation>
      <xsd:documentation>
        <div class="summary">A container for a group of <a href="el.action">actions</a></div>
        <div class="description">
        <p><tt>ActionList</tt> contains a series of <tt>action</tt>s or
        nested <tt>actionList</tt>s.</p>
      </div>
        <div class="example">See examples in <a href="el.action">action</a></div>
       <div class="example">
        <pre>
&lt;!-- demonstrating parallel and sequential actions --&gt;
&lt;actionList order="parallel" endCondition="all food cooked"&gt;
  &lt;!-- meat and potatoes are cooked in parallel --&gt;
  &lt;actionList title="meat"&gt;
    &lt;action title="cook" endCondition="cooked"&gt;
      &lt;p&gt;Roast meat&lt;/p&gt;
    &lt;/action&gt;
    &lt;action&gt;&lt;p&gt;Keep warm in oven&lt;/p&gt;&lt;/action&gt;
  &lt;/actionList&gt;
  &lt;actionList title="vegetables"&gt;
    &lt;actionList title="cookVeg" endCondition="cooked"&gt;
      &lt;action title="boil water" endCondition="water boiling"&gt;
        &lt;p&gt;Heat water&lt;/p&gt;
      &lt;/action&gt;
      &lt;action title="cook" endCondition="potatoes cooked"&gt;
        &lt;p&gt;Cook potatoes&lt;/p&gt;
      &lt;/action&gt;
    &lt;/actionList&gt;
    &lt;action&gt;&lt;p&gt;Keep warm in oven&lt;/p&gt;&lt;/action&gt;
  &lt;/actionList&gt;
&lt;/actionList&gt;

        </pre>
      </div>
      </xsd:documentation>
    </xsd:annotation>

    <xsd:complexType mixed="true">
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:any processContents="lax"/>
      </xsd:sequence>
      <xsd:attributeGroup ref="tit_id_conv_dictGroup"/>
      <xsd:attributeGroup ref="actionGroup"/>
      <xsd:attribute name="type" type="xsd:string">
        <xsd:annotation>
          <xsd:documentation>
           <div class="summary">The type of the actionList; no defined semantics</div>
          </xsd:documentation>
        </xsd:annotation>
     </xsd:attribute>
     <xsd:attribute name="order" default="sequential">
      <xsd:annotation>
        <xsd:documentation>
          <div class="description">Describes whether child elements are sequential or parallel</div>
        </xsd:documentation>
      </xsd:annotation>
     <xsd:simpleType>
       <xsd:restriction base="xsd:string">
         <xsd:enumeration value="sequential"/>
         <xsd:enumeration value="parallel"/>
       </xsd:restriction>
     </xsd:simpleType>
     </xsd:attribute>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="object" id="el.object">
    <xsd:annotation>
      <xsd:documentation>
        <div class="summary">An object which might occur in scientific data or narrative</div>
        <div class="description"><p>Deliberately vague.
        Thus an instrument might be built from sub component
        objects, or a program could be composed of smaller modules (objects).
        Unrestricted content model</p>
      </div>
        <div class="example"><pre>&lt;object title="frog" type="amphibian" count="5"&gt;
  &lt;scalar dataType="xsd:float" title="length" units="unit:cm"&gt;5&lt;/scalar&gt;
&lt;/object&gt;
</pre></div>
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType mixed="true">
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:any processContents="lax"/>
      </xsd:sequence>
      <xsd:attributeGroup ref="tit_id_conv_dictGroup"/>
      <xsd:attribute name="type" type="xsd:string">
        <xsd:annotation>
          <xsd:documentation>
            <div class="summary">Type of the object. Uncontrolled semantics</div>
          </xsd:documentation>
        </xsd:annotation>
      </xsd:attribute>
      <xsd:attribute name="count" type="countType"/>
    </xsd:complexType>

  </xsd:element>

  <xsd:element name="observation" id="el.observation">
    <xsd:annotation>
      <xsd:documentation>
        <div class="summary">An observation or occurrence</div>
        <div class="description"><p>A container for any events that
        need to be recorded, whether planned or not. They can include notes,
        measurements, conditions that may be referenced elsewhere, etc. There are
        no controlled semantics </p>
        <div class="example"><pre>&lt;observation type="ornithology"&gt;
  &lt;object title="sparrow" count="3"/&gt;
&lt;/observation&gt;
</pre></div>
      </div>
     </xsd:documentation>
   </xsd:annotation>
    <xsd:complexType mixed="true">
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:any processContents="lax"/>
      </xsd:sequence>
      <xsd:attributeGroup ref="tit_id_conv_dictGroup"/>
      <xsd:attribute name="type" type="xsd:string">
        <xsd:annotation>
          <xsd:documentation>
            <div class="summary">Type of observation (uncontrolled vocabulary)</div>
         </xsd:documentation>
       </xsd:annotation>
      </xsd:attribute>
      <xsd:attribute name="count" type="countType"/>
    </xsd:complexType>

  </xsd:element>

<xsd:annotation>
  <xsd:documentation>
    <div class="subheading">Dictionary components</div>
    <div class="description">
    <p>Dictionaries are a major part of STMML and supported as follows:</p>
    <p>The dictionary itself:</p>
    <ul>
    <li><a href="el.dictionary">dictionary</a>. This element defines a dictionary
    and is often the root element (though a data instance might also be combined
    with a dictionary). The dictionary play a similar role to a simple schema, by defining
    data types and other constraints (such as enumerations). By transforming a dictionary to
    schema format, schema-based tools can be used for validation. A dictionary is
    normally composed of <a href="el.entry">entry</a>s.</li>
    <li><a href="el.entry">entry</a>. An entry contanins information which <em>describes</em>
    or <em>constrains</em> elements in a data instance. The link is made through a
    <a href="st.dictRef">dictRef</a> attribute on the data element.
    Descriptive information can apply to any type of element (not necessarily
    part of or derived from the STM Schema). Constraints are similar to those in XML Schemas
    and use the same vocabulary (dataTypes, value ranges, enumerations, patterns, etc.). They normally
    apply to elements from the STM Schema or derived from it.<br/>
    In addition entrys can constrain elements to have the same
    higher-level structures and constraints defined by STM Schema. Thus entrys can require a
    data element to be a matrix, of a given type, with fixed number os rows and columns. These
    constraints are usually attributes on the entry element, which therefore maps
    directly onto the instance. Every entry has a mandatory <tt>term</tt> attribute
    which is the formal text string representing the concept. This string can contain any allowed
    XML characters (e.g. greek characters) but not markup (e.g. MathML or CML).</li>
    <li><a href="el.definition">definition</a>. An almost mandatory child element of entry, giving
    a formal definition of the term</li>
    <li><a href="el.description">description</a>. Additional
    descriptive informati&gt;on for an entry.
    This can contain any content, often HTML, but also MathML, CML for description of
    equations, chemical formulae, etc.</li>
    <li><a href="el.alternative">alternative</a>. Alternative strings for describing the
    concept. These can be any of the stnadard lexical and terminological data categories
    such as synonyms, abbreviations, homonyms, etc. (see ISO12620 for a full range).</li>
    <li><a href="el.enumeration">enumeration</a>. A list of allowed values for the data
    element (or elements in arrays, matrices).</li>
    <li><a href="el.relatedEntry">relatedEntry</a>. A related entry. Sometimes this is
    descriptive (e.g. "seeAlso" provides additional information on related concepts).
    It can also be used for constraints, and there is a small controlled vocabulary
    of relationships, but no universal syntax. We
    support parentage (e.g. through "partitiveParent" = "partOf"). In principle this can
    be used with <a href="el.appinfo">appinfo</a> to provide algorithmically constructed
    relationships.
    </li>
    <li><em>attributes</em>. A wide range of constraints is provided through attributes,
    several being similar to facets on XML Schema datatypes:
    <ul>
    <li><b>rows</b> and <b>columns</b>, the structure of the data element.</li>
    <li><b>recommendedUnits</b>, <b>units</b> and <b>unitType</b>,
    the units of the data element.</li>
    <li><b>minExclusive</b>, <b>minInclusive</b>, <b>maxExclusive</b>
    and <b>maxInclusive</b>,
    the value of the data element.</li>
    <li><b>totalDigits</b>, <b>fractionDigits</b>, <b>length</b>,
     <b>maxLength</b>, <b>minLength</b>
    and <b>pattern</b>. The lexical form of the  data element.</li>
    </ul>
    </li>
    <li><a href="el.annotation">annotation</a>. Similar to XML Schema, this has children
    <a href="el.documentation">documentation</a> for information about the entry (normally
    curatorial) and <a href="el.appinfo">appinfo</a> to describe entries and constraints
    in machine-processable fashion.
    .</li>
    </ul>
    </div>
  </xsd:documentation>
</xsd:annotation>

  <xsd:element name="dictionary" id="el.dictionary">
    <xsd:annotation>
      <xsd:documentation>
         <div class="summary">A dictionary</div>
         <div class="description">
         <p>A dictionary is a container for <a href="el.entry">entry</a>
        elements. Dictionaries can also contain unit-related information.</p>
         <p>The dictRef attribute on a <tt>dictionary</tt> element
         sets a namespace-like prefix allowing the dictionary to be referenced from
         within the document. In general dictionaries are
         referenced from an element using the
         <a href="gp.dictRefGroup">dictRef</a> attribute.</p>

        </div>
        <div class="example">
        <pre>
&lt;stm:dictionary xmlns:stm="http://www.xml-cml.org/schema/stmml"&gt;
  &lt;stm:entry id="a001" term="Amplitude for charge density mixing"
    dataType="stm:decimal"
    units="arbitrary"&gt;
    &lt;stm:annotation&gt;
      &lt;stm:documentation&gt;
        &lt;div class="summary"&gt;Amplitude for charge density mixing&lt;/div&gt;
        &lt;div class="description"&gt;Not yet filled in...&lt;/div&gt;
      &lt;/stm:documentation&gt;
    &lt;/stm:annotation&gt;
    &lt;stm:alternative type="abbreviation"&gt;CDMixAmp&lt;/stm:alternative&gt;
  &lt;/stm:entry&gt;
&lt;/stm:dictionary&gt;

        </pre>
       </div>
         <div class="example">
       <p><tt>dictionary</tt> can be used in an instance
        document to reference the dictionary used. Example:</p>
        <pre>
&lt;list&gt;
  &lt;dictionary
     dictRef="core" href="../dictionary/coreDict.xml"/&gt;
    &lt;!-- ... --&gt;
    &lt;scalar dictRef="core:frog" title="specimens"&gt;Frogs&lt;/scalar&gt;
&lt;/list&gt;

        </pre>
       </div>
      </xsd:documentation>
    </xsd:annotation>

    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="unitList" minOccurs="0" maxOccurs="unbounded">
          <xsd:annotation>
            <xsd:documentation>
              <div class="description"><p>This is so that a reference to a UnitList can be put in a dictionary.</p>
            </div>
            </xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element ref="annotation" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="description" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="entry" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attributeGroup ref="tit_id_conv_dictGroup"/>

      <xsd:attribute name="href" type="xsd:anyURI">
        <xsd:annotation>
          <xsd:documentation>
            <div class="summary">URI giving the location of the document. Mandatory if <tt>dictRef</tt>
            present.
          </div>
          </xsd:documentation>
        </xsd:annotation>
      </xsd:attribute>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="entry" id="el.entry">
    <xsd:annotation>
      <xsd:documentation>
       <div class="summary">A dictionary entry</div>
       <div class="example">
        <pre>
&lt;s:entry id="a003" term="alpha"
  dataType="float"
  minInclusive="0.0"
  maxInclusive="180.0"
  recommendedUnits="degrees"
  xmlns:s="http://www.xml-cml.org/schema/stmml"&gt;
  &lt;s:definition&gt;The alpha cell angle&lt;/s:definition&gt;
&lt;/s:entry&gt;

         </pre>
      </div>
       <div class="example">
        <pre>
&lt;entry id="a003"
  term="matrix1"
  dataType="float"
  rows="3"
  columns="4"
  unitType="unit:length"
  minInclusive="0.0"
  maxInclusive="100.0"
  recommendedUnits="unit:m"
  totalDigits="8"
  fractionDigits="3"&gt;
  &lt;definition&gt;A matrix of lengths&lt;/definition&gt;
  &lt;description&gt;A data instance will have a matrix which points
  to this entry (e.g. dictRef="foo:matrix1"). The matrix must
  be 3*4, composed of floats in 8.3 format, of type length,
  values between 0 and 100 and with recommended units metres.
  &lt;/description&gt;
&lt;/entry&gt;

         </pre>
      </div>
</xsd:documentation>
    </xsd:annotation>

    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="definition" minOccurs="0"/>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
          <xsd:element ref="alternative"/>
          <xsd:element ref="annotation"/>
          <xsd:element ref="definition"/>
          <xsd:element ref="description"/>
          <xsd:element ref="enumeration"/>
          <xsd:element ref="relatedEntry"/>
        </xsd:choice>
      </xsd:sequence>
      <xsd:attributeGroup ref="tit_id_convGroup"/>
      <xsd:attribute name="dataType" type="xsd:string"/>
      <xsd:attribute name="rows" type="xsd:positiveInteger" default="1"/>
      <xsd:attribute name="columns" type="xsd:positiveInteger" default="1"/>
      <xsd:attribute name="recommendedUnits" type="unitsType"/>
      <xsd:attribute name="unitType" type="xsd:string"/>
      <xsd:attribute name="minExclusive" type="xsd:decimal"/>
      <xsd:attribute name="minInclusive" type="xsd:decimal"/>
      <xsd:attribute name="maxExclusive" type="xsd:decimal"/>
      <xsd:attribute name="maxInclusive" type="xsd:decimal"/>
      <xsd:attribute name="totalDigits" type="xsd:positiveInteger"/>
      <xsd:attribute name="fractionDigits" type="xsd:nonNegativeInteger"/>
      <xsd:attribute name="length" type="xsd:nonNegativeInteger"/>
      <xsd:attribute name="minLength" type="xsd:nonNegativeInteger"/>
      <xsd:attribute name="maxLength" type="xsd:positiveInteger"/>
      <xsd:attribute name="units" type="unitsType"/>
      <xsd:attribute name="whiteSpace" type="xsd:string"/>
      <xsd:attribute name="pattern" type="xsd:string"/>
      <xsd:attribute name="term" type="xsd:string" use="required"/>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="definition" id="el.definition">
    <xsd:annotation>
      <xsd:documentation>
        <div class="summary">
        <p>The definition for a dictionary entry, scientific units, etc.</p>
         </div>

        <div class="description">
         <p>The definition should be a short nounal phrase defininining the
         subject of the entry. Definitions should not include commentary, implementations,
         equations or formulae (unless the subject is one of these) or examples. The
         <tt>description</tt> element can be used for these.</p>
        <p>The definition can be in any markup language, but normally XHTML will be used,
        perhaps with links to other XML namespaces such as CML for chemistry.</p>
         </div>

         <div class="example">
         <em>From the IUPAC Dictionary of Medicinal Chemistry</em><br/>
           <pre>
&lt;entry id="a7" term="Allosteric enzyme"&gt;
  &lt;definition&gt;An &lt;a href="#e3"&gt;enzyme&lt;/a&gt;
that contains a region to which small, regulatory molecules
("effectors") may bind in addition to and separate from the
substrate binding site and thereby affect the catalytic
activity.
  &lt;/definition&gt;
  &lt;description&gt;On binding the effector, the catalytic activity of the
&lt;strong&gt;enzyme&lt;/strong&gt; towards the substrate may be enhanced, in
which case the effector is an activator, or reduced, in which case
it is a de-activator or inhibitor.
  &lt;/description&gt;
&lt;/entry&gt;

           </pre>
         </div>
         </xsd:documentation>
       </xsd:annotation>
     <xsd:complexType mixed="true">
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:any processContents="lax"/>
      </xsd:sequence>
      <xsd:attributeGroup ref="sourceGroup"/>
    </xsd:complexType>

  </xsd:element>

  <xsd:element name="description" id="el.description">
    <xsd:annotation>
      <xsd:documentation>
        <div class="summary">Descriptive information in a dictionary entry, etc.</div>

        <div class="description">
         <p>Entries should have at least one separate <a href="el.definition">definition</a>s.
         <tt>description</tt> is then used for most of the other information, including
         examples. The <tt>class</tt> attribute has an uncontrolled vocabulary and
         can be used to clarify the purposes of the <tt>description</tt>
         elements.</p>
         </div>

        <div class="example">
         <em>From IUPAC Dictionary of Medicinal Chemistry</em>
           <pre>
&lt;entry id="a7" term="Allosteric enzyme"&gt;
  &lt;definition&gt;An &lt;a href="#e3"&gt;enzyme&lt;/a&gt;
that contains a region to which small, regulatory molecules
("effectors") may bind in addition to and separate from the
substrate binding site and thereby affect the catalytic
activity.
  &lt;/definition&gt;
  &lt;description&gt;On binding the effector, the catalytic activity of the
&lt;strong&gt;enzyme&lt;/strong&gt; towards the substrate may be enhanced, in
which case the effector is an activator, or reduced, in which case
it is a de-activator or inhibitor.
  &lt;/description&gt;
&lt;/entry&gt;

           </pre>
         </div>
        </xsd:documentation>
       </xsd:annotation>

    <xsd:complexType mixed="true">
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:any processContents="lax"/>
      </xsd:sequence>
      <xsd:attributeGroup ref="sourceGroup"/>
      <xsd:attribute name="class" type="xsd:string">
        <xsd:annotation>
          <xsd:documentation>
            <div class="description"><p>The type of this information. This is not controlled, but examples
            might include:</p>
            <ul>
              <li>description</li>
              <li>summary</li>
              <li>note</li>
              <li>usage</li>
              <li>qualifier</li>
            </ul>
          </div>
         </xsd:documentation>
       </xsd:annotation>
      </xsd:attribute>
    </xsd:complexType>

  </xsd:element>

  <xsd:element name="enumeration" id="el.enumeration">
    <xsd:annotation>
      <xsd:documentation>
        <div class="summary">An enumeration of string values associated
        with an <a href="el.entry">entry</a></div>

        <div class="description">
        <p>An enumeration of string values. Used where a dictionary entry constrains
        the possible values in a document instance. The dataTypes (if any) must all be
        identical and are defined by the dataType of the containing element.</p>
         </div>

        <div class="example">
           <pre>
&lt;entry term="crystal system" id="cryst1" dataType="string"&gt;
```

```
  &lt;definition&gt;A crystal system&lt;/definition&gt;
  &lt;enumeration value="triclinic"&gt;
    &lt;annotation&gt;
      &lt;documentation&gt;
        &lt;div class="summary"&gt;No constraints on lengths and angles&lt;/div&gt;
      &lt;/documentation&gt;
    &lt;/annotation&gt;
  &lt;/enumeration&gt;
  &lt;enumeration value="monoclinic"&gt;
    &lt;annotation&gt;
      &lt;documentation&gt;
        &lt;div class="summary"&gt;Two cell angles are right angles; no other constraints&lt;/div&gt;
      &lt;/documentation&gt;
    &lt;/annotation&gt;
  &lt;/enumeration&gt;
  &lt;enumeration value="orthorhombic"&gt;
    &lt;annotation&gt;
      &lt;documentation&gt;
        &lt;div class="summary"&gt;All three angles are right angles; no other constraints&lt;/div&gt;
      &lt;/documentation&gt;
    &lt;/annotation&gt;
  &lt;/enumeration&gt;
  &lt;enumeration value="tetragonal"&gt;
    &lt;annotation&gt;
      &lt;documentation&gt;
        &lt;div class="summary"&gt;Fourfold axis of symmetry; All three angles are right angles; two equal cell lengths; no other
constraints&lt;/div&gt;
      &lt;/documentation&gt;
    &lt;/annotation&gt;
  &lt;/enumeration&gt;
  &lt;enumeration value="trigonal"&gt;
    &lt;annotation&gt;
      &lt;documentation&gt;
        &lt;div class="summary"&gt;Threefold axis of symmetry; Two angles are right angles; one is 120 degrees; two equal lengths; no
other constraints&lt;/div&gt;
      &lt;/documentation&gt;
    &lt;/annotation&gt;
  &lt;/enumeration&gt;
  &lt;enumeration value="hexagonal"&gt;
    &lt;annotation&gt;
      &lt;documentation&gt;
        &lt;div class="summary"&gt;Sixfold axis of symmetry; Two angles are right angles; one is 120 degrees; two equal lengths; no
other constraints&lt;/div&gt;
      &lt;/documentation&gt;
    &lt;/annotation&gt;
  &lt;/enumeration&gt;
  &lt;enumeration value="cubic"&gt;
    &lt;annotation&gt;
      &lt;documentation&gt;
        &lt;div class="summary"&gt;All three angles are right angles; all cell lengths are equal&lt;/div&gt;
      &lt;/documentation&gt;
    &lt;/annotation&gt;
  &lt;/enumeration&gt;
&lt;/entry&gt;

           </pre>
         </div>

        <div class="description">
         <p>An enumeration of string values. The dataTypes (if any) must all be
         identical and are defined by the dataType of the containing element.</p>
         <p>Documentation can be added through an <a href="el.enumeration">enumeration</a>
         child</p>
       </div>
       </xsd:documentation>
    </xsd:annotation>

    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="annotation" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="value" type="xsd:string">
        <xsd:annotation>
          <xsd:documentation>
            <div class="summary">The value of the enumerated element.</div>
            <div class="description">
            Must be compatible
            with the dataType of the containing element (not schema-checkable directly
            but possible if dictionary is transformed to schema).</div>
          </xsd:documentation>
        </xsd:annotation>
      </xsd:attribute>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="alternative" id="el.alternative">
    <xsd:annotation>
      <xsd:documentation>
        <div class="summary">
           <p>An alternative name for an entry</p>
         </div>

<div class="description">
        <p>At present a child of <a href="#el.entry">entry</a> which represents
        an alternative string that refers to the concept. There is a partial controlled
        vocabulary in <tt>alternativeType</tt> with values such as :
        </p>
        <ul>
          <li>synonym</li>
          <li>acronym</li>
          <li>abbreviation</li>
        </ul>
         </div>

<div class="example">
           <pre>
 &lt;entry term="ammonia" id="a1"&gt;
   &lt;alternative type="synonym"&gt;Spirits of hartshorn&lt;/alternative&gt;
 &lt;/entry&gt;

           </pre>
         </div>
</xsd:documentation>
              </xsd:annotation>

    <xsd:complexType mixed="true">

      <xsd:attribute name="type" default="userType">
      <xsd:annotation>
        <xsd:documentation>
           <div class="summary">the type of an
               <a href="#el.alternative">alternative</a> element.</div>
           <div class="description">
            <p>
               <tt>alternativeType</tt> represents a the type of an
               <a href="#el.alternative">alternative</a> element.
            </p>
         </div>
         </xsd:documentation>
      </xsd:annotation>
      <xsd:simpleType>
     <xsd:restriction base="xsd:string">
       <xsd:enumeration value="synonym"/>
       <xsd:enumeration value="quasi-synonym"/>
       <xsd:enumeration value="acronym"/>
       <xsd:enumeration value="abbreviation"/>
       <xsd:enumeration value="homonym"/>
       <xsd:enumeration value="identifier"/>

       <xsd:enumeration value="userType"/>
     </xsd:restriction>
      </xsd:simpleType>
      </xsd:attribute>

      <xsd:attribute name="userType" type="userTypeType"/>

    </xsd:complexType>

  </xsd:element>

  <xsd:element name="relatedEntry" id="el.relatedEntry">
    <xsd:annotation>
      <xsd:documentation>
       <div class="summary">An entry related in some way to a dictionary entry, scientific units, etc.</div>
        <div class="description"><p>The range of relationships is not restricted
        but should include parents, aggregation, seeAlso
        etc. dataCategories from ISO12620 can be referenced through the <tt>userType</tt>
        attribute.</p></div>
        <div class="example">
               <pre>
&lt;stm:entry id="a14" term="Autoreceptor"
  xmlns:stm="http://www.xml-cml.org/schema/core"&gt;
  &lt;stm:definition&gt;An &lt;strong&gt;autoreceptor&lt;/strong&gt;, present at a nerve ending, is
    a &lt;a href="#r1"&gt;receptor&lt;/a&gt;
    that regulates, via positive or negative feedback processes, the
    synthesis and/or release of its own physiological ligand.
  &lt;/stm:definition&gt;
  &lt;stm:relatedEntry type="seeAlso" href="#h4"&gt;Heteroreceptor).&lt;/stm:relatedEntry&gt;
&lt;/stm:entry&gt;

               </pre>
             </div>
           </xsd:documentation>
          </xsd:annotation>
    <xsd:complexType mixed="true">
      <xsd:attribute name="type" default="userType">
         <xsd:simpleType>
            <xsd:annotation>
               <xsd:documentation>
                 <div class="description"><p>
                     <tt>relatedEntryType</tt> represents a the type of relationship in a
                     <a href="#el.relatedEntry">relatedEntry</a> element.
                  </p>
               </div>
              </xsd:documentation>
           </xsd:annotation>
           <xsd:restriction base="xsd:string">
             <xsd:enumeration value="parent"/>
             <xsd:enumeration value="partitiveParent"/>
             <xsd:enumeration value="child"/>
             <xsd:enumeration value="partitiveChild"/>
             <xsd:enumeration value="related"/>
             <xsd:enumeration value="synonym"/>
             <xsd:enumeration value="quasi-synonym"/>
             <xsd:enumeration value="antonym"/>
             <xsd:enumeration value="homonym"/>
             <xsd:enumeration value="see"/>
             <xsd:enumeration value="seeAlso"/>
             <xsd:enumeration value="abbreviation"/>
             <xsd:enumeration value="acronym"/>
             <xsd:enumeration value="userType"/>
           </xsd:restriction>
      </xsd:simpleType>
      </xsd:attribute>

      <xsd:attribute name="userType" type="userTypeType"/>
      <xsd:attribute name="href" type="xsd:anyURI"/>
    </xsd:complexType>

  </xsd:element>

  <xsd:element name="annotation" id="el.annotation">
    <xsd:annotation>
<xsd:documentation>
<div class="summary">
<p>A documentation container similar to <tt>annotation</tt> in XML Schema.</p>
</div>

<div class="description">        <p>A documentation container similar to <tt>annotation</tt> in XML Schema.
        At present this is experimental and designed to be used for dictionaries, units, etc.
        One approach is to convert these into XML Schemas when the <tt>documentation</tt>
        and <tt>appinfo</tt> children will emerge in their correct position in the
        derived schema.</p>
        <p>It is possible that this may develop as a useful tool for annotating components
        of complex objects such as molecules. </p>
         </div>

<div class="example">
         <pre>
&lt;entry term="matrix"&gt;
  &lt;annotation&gt;
    &lt;documentation&gt;This refers to mathematical matrices&lt;/documentation&gt;
    &lt;appinfo&gt;... some code to describe and support matrices ...&lt;/appinfo&gt;
  &lt;/annotation&gt;
&lt;/entry&gt;

           </pre>
         </div>
</xsd:documentation>
              </xsd:annotation>

    <xsd:complexType mixed="true">
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element ref="documentation"/>
        <xsd:element ref="appinfo"/>
      </xsd:choice>
      <xsd:attribute name="source" type="xsd:anyURI"/>
    </xsd:complexType>

  </xsd:element>


  <xsd:element name="documentation" id="el.documentation">
    <xsd:annotation>
      <xsd:documentation>
         <div class="summary">Documentation in the <a href="el.annotation">annotation</a> of an <a href="el.entry">entry</a></div>

        <div class="description">
        <p>A container similar to <tt>documentation</tt> in XML Schema.
        This is NOT part of the textual content of an entry but is designed to
        support the transformation of dictionary entrys into schemas for validation.
        This is experimental and should only be used for dictionaries, units, etc.
        One approach is to convert these into XML Schemas when the <tt>documentation</tt>
        and <tt>appinfo</tt> children will emerge in their correct position in the
        derived schema.</p>
        <p>Do NOT confuse documentation with the <a href="el.definition">definition</a>
        or the <a href="el.definition">definition</a> which are part of the content
        of the dictionary</p>
        <p>If will probably only be used when there is significant <a href="el.appinfo">appinfo</a>
        in the entry or where the entry defines an XSD-like datatype of an element in the document.</p>
         </div>

         <div class="example">
           <pre>
&lt;stm:documentation id="source"
xmlns:stm="http://www.xml-cml.org/schema/stmml"&gt;
Transcribed from IUPAC website
&lt;/stm:documentation&gt;

           </pre>
         </div>
       </xsd:documentation>
     </xsd:annotation>

    <xsd:complexType mixed="true">
```

```
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:any processContents="lax"/>
      </xsd:sequence>
      <xsd:attributeGroup ref="sourceGroup"/>
      <xsd:attributeGroup ref="idGroup"/>
    </xsd:complexType>

  </xsd:element>

  <xsd:element name="appinfo" id="el.appinfo">
    <xsd:annotation>
<xsd:documentation>
<div class="summary">
        <p>A container similar to <tt>appinfo</tt> in XML Schema.</p>
         </div>

<div class="description">
           <p>A container for machine processable documentation for an entry.
           This is likely to be platform and/or language specific. It is possible
           that XSLT, RDF or XBL will emerge as generic languages</p>
           <p>See <a href="el.annotation">annotation</a> and <a href="el.documentation">documentation</a> for further information</p>
         </div>

<div class="example">
<p>An example in XSLT where an element <tt>foo</tt> calls a bespoke
         template</p>.
         <pre>
&lt;s:appinfo
  xmlns:s="http://www.xml-cml.org/schema/core"
  xmlns="http://www.w3.org/1999/XSL/Transform"&gt;
  &lt;template match="foo"&gt;
    &lt;call-template name="processFoo"/&gt;
  &lt;/template&gt;
&lt;/s:appinfo&gt;

         </pre>
         </div>
</xsd:documentation>
          </xsd:annotation>
    <xsd:complexType mixed="true">
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:any processContents="lax"/>
      </xsd:sequence>
      <xsd:attribute name="source" type="xsd:anyURI"/>
    </xsd:complexType>

  </xsd:element>

   <xsd:attributeGroup name="dictRefGroup" id="attGp.dictRefGroup">
      <xsd:attribute id="att.dictRef" name="dictRef" type="namespaceRefType">
         <xsd:annotation>
            <xsd:documentation>
              <div class="summary">
               <p>A reference to a dictionary entry.</p>
              </div>

           <div class="description">
         <p>Elements in data instances such as <a href="el.scalar">scalar</a> may have a
         <tt>dictRef</tt> attribute to point to an entry in a dictionary. To avoid
         excessive use of (mutable) filenames and URIs we recommend a namespace
         prefix, mapped to a namespace URI in the normal manner. In this case,
         of course, the namespace URI must point to a real XML document containing
         <a href="el.entry">entry</a> elements and validated against STMML Schema.</p>
         <p>Where there is concern about the dictionary becoming separated from the document
         the dictionary entries can be physically included as part of the data instance
         and the normal XPointer addressing mechanism can be used.</p>
         <p>This attribute can also be used on <a href="el.dictionary">dictionary</a>
         elements to define the namespace prefix</p>
           </div>

         <div class="example">
           <pre>
&lt;scalar dataType="xsd:float" title="surfaceArea"
  dictRef="cmlPhys:surfArea"
  xmlns:cmlPhys="http://www.xml-cml.org/dict/physical"
  units="units:cm2"&gt;50&lt;/scalar&gt;

           </pre>
            </div>

         <div class="example">
           <pre>
&lt;stm:list xmlns:stm="http://www.xml-cml.org/schema/stmml"&gt;
  &lt;stm:observation&gt;
    &lt;p&gt;We observed &lt;object count="3" dictRef="#p1"/&gt;
      constructing dwellings of different material&lt;/p&gt;
  &lt;/stm:observation&gt;
  &lt;stm:entry id="p1" term="pig"&gt;
    &lt;stm:definition&gt;A domesticated animal.&lt;/stm:definition&gt;
    &lt;stm:description&gt;Predators include wolves&lt;/stm:description&gt;
    &lt;stm:description class="scientificName"&gt;Sus scrofa&lt;/stm:description&gt;
  &lt;/stm:entry&gt;
&lt;/stm:list&gt;

           </pre>
```

```
            </div>
           </xsd:documentation>
         </xsd:annotation>
      </xsd:attribute>
   </xsd:attributeGroup>

<xsd:annotation>
  <xsd:documentation>
    <div class="subheading">Metadata</div>
    <div class="description">
    <p>STMML supports metadata through the element
    <a href="el.metadata">metadata</a>. If necessary
    several of these can be contained in
    a <a href="el.metadataList">metadataList</a> element.
    </p>
    </div>
  </xsd:documentation>
</xsd:annotation>

<xsd:element name="metadata" id="el.metadata">
  <xsd:annotation>
    <xsd:documentation>
      <div class="summary">A general container for metadata</div>
      <div class="description"><p>A general container for metadata, including at least
      Dublin Core (DC) and CML-specific metadata</p>
      <p>In its simple form each element provides a name and content in a similar
      fashion to the <tt>meta</tt> element in HTML. <tt>metadata</tt> may have simpleContent
      (i.e. a string for adding further information - this is not controlled).</p>
    </div>
      <div class="example"><pre>
&lt;list&gt;
  &lt;metadataList&gt;
    &lt;metadata name="dc:coverage" content="Europe"/&gt;
    &lt;metadata name="dc:description" content="Ornithological chemistry"/&gt;
    &lt;metadata name="dc:identifier"  content="ISBN:1234-5678"/&gt;
    &lt;metadata name="dc:format" content="printed"/&gt;
    &lt;metadata name="dc:relation" content="abc:def123"/&gt;
    &lt;metadata name="dc:rights" content="licence:GPL"/&gt;
    &lt;metadata name="dc:subject" content="Informatics"/&gt;
    &lt;metadata name="dc:title" content="birds"/&gt;
    &lt;metadata name="dc:type" content="bird books on chemistry"/&gt;
    &lt;metadata name="dc:contributor" content="Tux Penguin"/&gt;
    &lt;metadata name="dc:creator" content="author"/&gt;
    &lt;metadata name="dc:publisher" content="Penguinone publishing"/&gt;
    &lt;metadata name="dc:source" content="penguinPub"/&gt;
    &lt;metadata name="dc:language" content="en-GB"/&gt;
    &lt;metadata name="dc:date" content="1752-09-10"/&gt;
  &lt;/metadataList&gt;
  &lt;metadataList&gt;
    &lt;metadata name="cmlm:safety" content="mostly harmless"/&gt;
    &lt;metadata name="cmlm:insilico" content="electronically produced"/&gt;
    &lt;metadata name="cmlm:structure" content="penguinone"/&gt;
    &lt;metadata name="cmlm:reaction" content="synthesis of penguinone"/&gt;
    &lt;metadata name="cmlm:identifier" content="smiles:O=C1C=C(C)C(C)(C)C(C)=C1"/&gt;
  &lt;/metadataList&gt;
&lt;/list&gt;
</pre></div>
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="name" type="metadataType">
          <xsd:annotation>
            <xsd:documentation>
              <div class="summary">The metadata type</div>
            </xsd:documentation>
          </xsd:annotation>
        </xsd:attribute>
        <xsd:attribute name="content" type="xsd:string">
          <xsd:annotation>
            <xsd:documentation>
              <div class="summary">The metadata</div>
             </xsd:documentation>
          </xsd:annotation>
        </xsd:attribute>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>

  <xsd:element name="metadataList" id="el.metadataList">
    <xsd:annotation>
      <xsd:documentation>
        <div class="summary">A general container for metadata elements</div>
        <div class="example"><pre>&lt;list&gt;
  &lt;metadataList&gt;
    &lt;metadata name="dc:coverage" content="Europe"/&gt;
    &lt;metadata name="dc:description" content="Ornithological chemistry"/&gt;
    &lt;metadata name="dc:identifier"  content="ISBN:1234-5678"/&gt;
    &lt;metadata name="dc:format" content="printed"/&gt;
    &lt;metadata name="dc:relation" content="abc:def123"/&gt;
    &lt;metadata name="dc:rights" content="licence:GPL"/&gt;
    &lt;metadata name="dc:subject" content="Informatics"/&gt;
    &lt;metadata name="dc:title" content="birds"/&gt;
    &lt;metadata name="dc:type" content="bird books on chemistry"/&gt;
    &lt;metadata name="dc:contributor" content="Tux Penguin"/&gt;
    &lt;metadata name="dc:creator" content="author"/&gt;
    &lt;metadata name="dc:publisher" content="Penguinone publishing"/&gt;
    &lt;metadata name="dc:source" content="penguinPub"/&gt;
    &lt;metadata name="dc:language" content="en-GB"/&gt;
    &lt;metadata name="dc:date" content="1752-09-10"/&gt;
  &lt;/metadataList&gt;
  &lt;metadataList&gt;
    &lt;metadata name="cmlm:safety" content="mostly harmless"/&gt;
    &lt;metadata name="cmlm:insilico" content="electronically produced"/&gt;
    &lt;metadata name="cmlm:structure" content="penguinone"/&gt;
    &lt;metadata name="cmlm:reaction" content="synthesis of penguinone"/&gt;
    &lt;metadata name="cmlm:identifier" content="smiles:O=C1C=C(C)C(C)(C)C(C)=C1"/&gt;
  &lt;/metadataList&gt;
&lt;/list&gt;
</pre></div>
      </xsd:documentation>
    </xsd:annotation>

    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="metadata" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:simpleType name="metadataType" id="md.metadataType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="dc:coverage">
        <xsd:annotation>
          <xsd:documentation>
            <div class="definition">The extent or scope of the
            content of the resource.</div>

            <div class="description">Coverage will typically include
            spatial location (a place name or geographic
            coordinates), temporal period (a period label, date, or
            date range) or jurisdiction (such as a named
            administrative entity). Recommended best practice is to
            select a value from a controlled vocabulary (for
            example, the Thesaurus of Geographic Names [TGN]) and
            that, where appropriate, named places or time periods
            be used in preference to numeric identifiers such as
            sets of coordinates or date ranges.
          </div>
</xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>

      <xsd:enumeration value="dc:description">
        <xsd:annotation>
          <xsd:documentation>
            <div class="definition">An account of the content of the
            resource.</div>

            <div class="description">Description may include but is not
            limited to: an abstract, table of contents, reference
            to a graphical representation of content or a free-text
            account of the content.
          </div>
</xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>

      <xsd:enumeration value="dc:identifier">
        <xsd:annotation>
          <xsd:documentation>
            <div class="definition">An unambiguous reference to the
            resource within a given context.</div>

            <div class="description">Recommended best practice is to
            identify the resource by means of a string or number
            conforming to a formal identification system. Example
            formal identification systems include the Uniform
            Resource Identifier (URI) (including the Uniform
            Resource Locator (URL)), the Digital Object Identifier
            (DOI) and the International Standard Book Number
            (ISBN).
          </div>
</xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>

      <xsd:enumeration value="dc:format">
        <xsd:annotation>
          <xsd:documentation>
            <div class="definition">The physical or digital
            manifestation of the resource.</div>

            <div class="description">Typically, Format may include the
            media-type or dimensions of the resource. Format may be
            used to determine the software, hardware or other
            equipment needed to display or operate the resource.
            Examples of dimensions include size and duration.
            Recommended best practice is to select a value from a
            controlled vocabulary (for example, the list of
            Internet Media Types [MIME] defining computer media
            formats).
          </div>
</xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>

      <xsd:enumeration value="dc:relation">
        <xsd:annotation>
          <xsd:documentation>
            <div class="definition">A reference to a related
            resource.</div>

            <div class="description">Recommended best practice is to
            reference the resource by means of a string or number
            conforming to a formal identification system.
          </div>
</xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>

      <xsd:enumeration value="dc:rights">
        <xsd:annotation>
          <xsd:documentation>
            <div class="definition">Information about rights held in
            and over the resource.</div>

            <div class="description">Typically, a Rights element will
            contain a rights management statement for the resource,
            or reference a service providing such information.
            Rights information often encompasses Intellectual
            Property Rights (IPR), Copyright, and various Property
            Rights. If the Rights element is absent, no assumptions
            can be made about the status of these and other rights
            with respect to the resource.
          </div>
</xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>

      <xsd:enumeration value="dc:subject">
        <xsd:annotation>
          <xsd:documentation>
            <div class="definition">The topic of the content of the
            resource.</div>

            <div class="description">Typically, a Subject will be
            expressed as keywords, key phrases or classification
            codes that describe a topic of the resource.
            Recommended best practice is to select a value from a
            controlled vocabulary or formal classification
            scheme.
          </div>
</xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>

      <xsd:enumeration value="dc:title">
        <xsd:annotation>
          <xsd:documentation>
            <div class="definition">A name given to the resource.</div>

            <div class="description">Typically, a Title will be a name by
            which the resource is formally known.
          </div>
         </xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>

      <xsd:enumeration value="dc:type">
        <xsd:annotation>
          <xsd:documentation>
            <div class="definition">The nature or genre of the
            content of the resource.</div>

            <div class="description">Type includes terms describing
            general categories, functions, genres, or aggregation
            levels for content. Recommended best practice is to
            select a value from a controlled vocabulary (for
            example, the working draft list of Dublin Core Types
            [DCT1]). To describe the physical or digital
            manifestation of the resource, use the FORMAT
            element.
          </div>
         </xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>

      <xsd:enumeration value="dc:contributor">
        <xsd:annotation>
          <xsd:documentation>
            <div class="definition">An entity responsible for making
            contributions to the content of the resource.</div>
            <div class="description">Examples of a Contributor include a
            person, an organisation, or a service. Typically, the
            name of a Contributor should be used to indicate the
            entity.
          </div>
         </xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>

      <xsd:enumeration value="dc:creator">
        <xsd:annotation>
          <xsd:documentation>
            <div class="definition">An entity primarily responsible
            for making the content of the resource.</div>

            <div class="description">Examples of a Creator include a
            person, an organisation, or a service. Typically, the
            name of a Creator should be used to indicate the
            entity.
          </div>
          </xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>

      <xsd:enumeration value="dc:publisher">
        <xsd:annotation>
          <xsd:documentation>
            <div class="definition">An entity responsible for making
            the resource available</div>

            <div class="description">Examples of a Publisher include a
            person, an organisation, or a service. Typically, the
            name of a Publisher should be used to indicate the
            entity.
          </div>
         </xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>

      <xsd:enumeration value="dc:source">
        <xsd:annotation>
          <xsd:documentation>
            <div class="definition">A Reference to a resource from
            which the present resource is derived.</div>

            <div class="description">The present resource may be derived
            from the Source resource in whole or in part.
            Recommended best practice is to reference the resource
            by means of a string or number conforming to a formal
            identification system.
          </div>
</xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>

      <xsd:enumeration value="dc:language">
        <xsd:annotation>
          <xsd:documentation>
            <div class="definition">A language of the intellectual
            content of the resource.</div>

            <div class="description">Recommended best practice for the
            values of the Language element is defined by RFC 1766
            [RFC1766] which includes a two-letter Language Code
            (taken from the ISO 639 standard [ISO639]), followed
            optionally, by a two-letter Country Code (taken from
            the ISO 3166 standard [ISO3166]). For example, 'en' for
            English, 'fr' for French, or 'en-uk' for English used
            in the United Kingdom.
          </div>
</xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>

      <xsd:enumeration value="dc:date">
        <xsd:annotation>
          <xsd:documentation>
            <div class="definition">A date associated with an event
            in the life cycle of the resource.</div>

            <div class="description">Typically, Date will be associated
            with the creation or availability of the resource.
            Recommended best practice for encoding the date value
            is defined in a profile of ISO 8601 [W3CDTF] and
            follows the YYYY-MM-DD format.
          </div>
</xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>

      <xsd:enumeration value="cmlm:safety">
        <xsd:annotation>
          <xsd:documentation>
            <div class="definition">Entry contains information
            relating to chemical safety</div>

            <div class="description">Typically the content will be a
            reference to a handbook, MSDS, threshhold or other
            human-readable string
          </div>
</xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>

      <xsd:enumeration value="cmlm:insilico">
        <xsd:annotation>
          <xsd:documentation>
            <div class="definition">Part or whole of the information
            was computer-generated</div>

            <div class="description">Typically the content will be the
            name of a method or a program
          </div>
</xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>

      <xsd:enumeration value="cmlm:structure">
        <h2>structure</h2>

        <xsd:annotation>
          <xsd:documentation>
            <div class="definition">3D structure included</div>

            <div class="description">details included
          </div>
</xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>

      <xsd:enumeration value="cmlm:reaction">
      </xsd:enumeration>

      <xsd:enumeration value="cmlm:identifier">
      </xsd:enumeration>

      <xsd:enumeration value="other">
      </xsd:enumeration>
    </xsd:restriction>
  </xsd:simpleType>

<xsd:annotation>
  <xsd:documentation>
    <div class="heading">Scientific Units</div>
  </xsd:documentation>
</xsd:annotation>

<xsd:element name="dimension" id="el.dimension">
  <xsd:annotation>
    <xsd:documentation>
      <div class="summary">A dimension supporting scientific units</div>
      <div class="description">
        <p>This will be primarily used within the defintion of
        <a href="el.unit">units</a>s.</p>
      </div>
    <div class="example">
       <pre>
&lt;unitType id="energy" name="energy"&gt;
  &lt;dimension name="length"/&gt;
  &lt;dimension name="mass"/&gt;
  &lt;dimension name="time" power="-1"/&gt;
&lt;/unitType&gt;

       </pre>
    </div>
    </xsd:documentation>
  </xsd:annotation>

  <xsd:complexType>
    <xsd:sequence>
    </xsd:sequence>
    <xsd:attribute name="name" type="dimensionType" use="required">
      <xsd:annotation>
        <xsd:documentation>
          <div class="summary">The type of the dimension</div>
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="power" type="xsd:decimal" default="1">
      <xsd:annotation>
        <xsd:documentation>
          <div class="summary">The power to which the dimension should be raised</div>
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
  </xsd:complexType>
</xsd:element>

<xsd:simpleType name="dimensionType" id="st.dimensionType">
```

```
  <xsd:annotation>
    <xsd:documentation>
      <div class="summary">Allowed values for dimension Types (for quantities).</div>
      <div class="description">
        <p>These are the 7 types prescribed by the SI system, together
         with the "dimensionless" type. We intend to be somewhat uncoventional
         and explore enhanced values of "dimensionless", such as "angle".
         This may be heretical, but we find the present system impossible to implement
         in many cases.</p>
        <p>Used for constructing entries in a dictionary of units</p>
      </div>

      <div class="example">
        <pre>
&lt;unitType id="energy" name="energy"&gt;
  &lt;dimension name="length"/&gt;
  &lt;dimension name="mass"/&gt;
  &lt;dimension name="time" power="-1"/&gt;
&lt;/unitType&gt;

        </pre>
      </div>
    </xsd:documentation>
  </xsd:annotation>

  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="mass"/>
    <xsd:enumeration value="length"/>
    <xsd:enumeration value="time"/>
    <xsd:enumeration value="charge"/>
    <xsd:enumeration value="amount"/>
    <xsd:enumeration value="luminosity"/>
    <xsd:enumeration value="temperature"/>
    <xsd:enumeration value="dimensionless"/>
    <xsd:enumeration value="angle">
      <xsd:annotation>
        <xsd:documentation>
          <div class="summary">An angle (formally dimensionless, but useful to have units).</div>
        </xsd:documentation>
      </xsd:annotation>
    </xsd:enumeration>
  </xsd:restriction>
</xsd:simpleType>

  <xsd:element name="unitList" id="el.unitList">
    <xsd:annotation>
      <xsd:documentation>
        <div class="summary">A container for several unit entries</div>
        <div class="description">Usually forms the complete units dictionary (along with metadata)</div>
        <div class="example"><pre>&lt;stm:unitList xmlns:stm="http://www.xml-cml.org/schema/stmml"&gt;

&lt;!-- ====================================================================== --&gt;
&lt;!-- ========================= fundamental types =========================== --&gt;
&lt;!-- ====================================================================== --&gt;

&lt;stm:unitType id="length" name="length"&gt;
  &lt;stm:dimension name="length"/&gt;
&lt;/stm:unitType&gt;

&lt;stm:unitType id="time" name="time"&gt;
  &lt;stm:dimension name="time"/&gt;
&lt;/stm:unitType&gt;

&lt;!-- ... --&gt;

&lt;stm:unitType id="dimensionless" name="dimensionless"&gt;
  &lt;stm:dimension name="dimensionless"/&gt;
&lt;/stm:unitType&gt;

&lt;!-- ====================================================================== --&gt;
&lt;!-- ========================== derived types ============================== --&gt;
&lt;!-- ====================================================================== --&gt;

&lt;stm:unitType id="acceleration" name="acceleration"&gt;
  &lt;stm:dimension name="length"/&gt;
  &lt;stm:dimension name="time" power="-2"/&gt;
&lt;/stm:unitType&gt;

&lt;!-- ... --&gt;

&lt;!-- ====================================================================== --&gt;
&lt;!-- ====================== fundamental SI units =========================== --&gt;
&lt;!-- ====================================================================== --&gt;

&lt;stm:unit id="second" name="second" unitType="time"&gt;
  &lt;stm:description&gt;The SI unit of time&lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;stm:unit id="meter" name="meter" unitType="length"
  abbreviation="m"&gt;
  &lt;stm:description&gt;The SI unit of length&lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;!-- ... --&gt;
```

```
&lt;stm:unit id="kg" name="nameless" unitType="dimensionless"
  abbreviation="nodim"&gt;
  &lt;stm:description&gt;A fictitious parent for dimensionless units&lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;!-- ====================================================================== --&gt;
&lt;!-- ===================== derived SI units ================================ --&gt;
&lt;!-- ====================================================================== --&gt;

&lt;stm:unit id="newton" name="newton" unitType="force"&gt;
  &lt;stm:description&gt;The SI unit of force&lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;!-- ... --&gt;

&lt;!-- multiples of fundamental SI units --&gt;

&lt;stm:unit id="g" name="gram" unitType="mass"
  parentSI="kg"
  multiplierToSI="0.001"
  abbreviation="g"&gt;
  &lt;stm:description&gt;0.001 kg. &lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;stm:unit id="celsius" name="Celsius" parentSI="k"
  multiplierToSI="1"
  constantToSI="273.18"&gt;
  &lt;stm:description&gt;&lt;p&gt;A common unit of temperature&lt;/p&gt;&lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;!-- fundamental non-SI units --&gt;

&lt;stm:unit id="inch" name="inch" parentSI="meter"
   abbreviation="in"
  multiplierToSI="0.0254" &gt;
  &lt;stm:description&gt;An imperial measure of length&lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;!-- derived non-SI units --&gt;

&lt;stm:unit id="l" name="litre" unitType="volume"
   parentSI="meterCubed"
   abbreviation="l"
   multiplierToSI="0.001"&gt;
  &lt;stm:description&gt;Nearly 1 dm**3 This is not quite exact&lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;!-- ... --&gt;

&lt;stm:unit id="fahr" name="fahrenheit" parentSI="k"
   abbreviation="F"
  multiplierToSI="0.55555555555555555"
  constantToSI="-17.777777777777777777"&gt;
  &lt;stm:description&gt;An obsolescent unit of temperature still used in popular
  meteorology&lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;/stm:unitList&gt;
</pre></div>
        <div class="example"><pre>&lt;stm:unitList
    xmlns:stm="http://www.xml-cml.org/schema/stmml"
    dictRef="unit" href="units.xml" /&gt;
</pre></div>
         </xsd:documentation>
       </xsd:annotation>

    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="unitType" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="unit" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attributeGroup ref="tit_id_conv_dictGroup"/>
      <xsd:attribute name="href" type="xsd:anyURI">
        <xsd:annotation>
          <xsd:documentation>
            <div class="summary">Maps a <a href="st.dictRefType">dictRef</a>
            prefix to the location of a dictionary.</div>
            <div class="description">This requires the prefix and the physical URI
            address to be contained within the same file. We can anticipate that
            better mechanisms will arise - perhaps through XMLCatalogs.
            At least it works at present.</div>
          </xsd:documentation>
       </xsd:annotation>
      </xsd:attribute>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="unitType" id="el.unitType">
    <xsd:annotation>
      <xsd:documentation>
         <div class="summary">An element containing the description of a scientific unit</div>
         <div class="description"><p>Mandatory for SI Units,
          optional for nonSI units since they should be able to obtain this
          from their parent. For complex derived units without parents it may be
          useful.</p>
          <p>Used within a unitList</p>
          <p>Distinguish carefully from <a href="st.unitsType">unitsType</a>
          which is primarily used for attributes describing the units that elements
          carry</p></div>
         <div class="example">
       <pre>&lt;stm:unitList xmlns:stm="http://www.xml-cml.org/schema/stmml"&gt;

&lt;!-- ====================================================================== --&gt;
&lt;!-- ========================== fundamental types =========================== --&gt;
&lt;!-- ====================================================================== --&gt;

&lt;stm:unitType id="length" name="length"&gt;
  &lt;stm:dimension name="length"/&gt;
&lt;/stm:unitType&gt;

&lt;stm:unitType id="time" name="time"&gt;
  &lt;stm:dimension name="time"/&gt;
&lt;/stm:unitType&gt;

&lt;!-- ... --&gt;

&lt;stm:unitType id="dimensionless" name="dimensionless"&gt;
  &lt;stm:dimension name="dimensionless"/&gt;
&lt;/stm:unitType&gt;

&lt;!-- ====================================================================== --&gt;
&lt;!-- ========================== derived types =============================== --&gt;
&lt;!-- ====================================================================== --&gt;

&lt;stm:unitType id="acceleration" name="acceleration"&gt;
  &lt;stm:dimension name="length"/&gt;
  &lt;stm:dimension name="time" power="-2"/&gt;
&lt;/stm:unitType&gt;

&lt;!-- ... --&gt;

&lt;!-- ====================================================================== --&gt;
&lt;!-- ======================= fundamental SI units =========================== --&gt;
&lt;!-- ====================================================================== --&gt;

&lt;stm:unit id="second" name="second" unitType="time"&gt;
  &lt;stm:description&gt;The SI unit of time&lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;stm:unit id="meter" name="meter" unitType="length"
  abbreviation="m"&gt;
  &lt;stm:description&gt;The SI unit of length&lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;!-- ... --&gt;

&lt;stm:unit id="kg" name="nameless" unitType="dimensionless"
  abbreviation="nodim"&gt;
  &lt;stm:description&gt;A fictitious parent for dimensionless units&lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;!-- ====================================================================== --&gt;
&lt;!-- ===================== derived SI units ================================= --&gt;
&lt;!-- ====================================================================== --&gt;

&lt;stm:unit id="newton" name="newton" unitType="force"&gt;
  &lt;stm:description&gt;The SI unit of force&lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;!-- ... --&gt;

&lt;!-- multiples of fundamental SI units --&gt;

&lt;stm:unit id="g" name="gram" unitType="mass"
  parentSI="kg"
  multiplierToSI="0.001"
  abbreviation="g"&gt;
  &lt;stm:description&gt;0.001 kg. &lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;stm:unit id="celsius" name="Celsius" parentSI="k"
  multiplierToSI="1"
  constantToSI="273.18"&gt;
  &lt;stm:description&gt;&lt;p&gt;A common unit of temperature&lt;/p&gt;&lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;!-- fundamental non-SI units --&gt;

&lt;stm:unit id="inch" name="inch" parentSI="meter"
   abbreviation="in"
  multiplierToSI="0.0254" &gt;
  &lt;stm:description&gt;An imperial measure of length&lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;!-- derived non-SI units --&gt;

&lt;stm:unit id="l" name="litre" unitType="volume"
   parentSI="meterCubed"
   abbreviation="l"
   multiplierToSI="0.001"&gt;
  &lt;stm:description&gt;Nearly 1 dm**3 This is not quite exact&lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;!-- ... --&gt;

&lt;stm:unit id="fahr" name="fahrenheit" parentSI="k"
   abbreviation="F"
  multiplierToSI="0.55555555555555555"
  constantToSI="-17.777777777777777777"&gt;
  &lt;stm:description&gt;An obsolescent unit of temperature still used in popular
  meteorology&lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;/stm:unitList&gt;
</pre>
      </div>
       </xsd:documentation>
          </xsd:annotation>

    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="dimension" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="id" type="idType" use="required">
      </xsd:attribute>
      <xsd:attribute name="name" type="xsd:string" use="required"/>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="unit" id="el.unit">
    <xsd:annotation>
      <xsd:documentation>
      <div class="summary">A scientific unit</div>
      <div class="description"><p>A scientific unit. Units are of the following types:</p>
        <ul>
        <li>SI Units. These may be one of the seven fundamental types
        (e.g. meter) or may be derived (e.g. joule). An SI unit is
        identifiable because it has no parentSI attribute and will have
        a unitType attribute.</li>
        <li>nonSI Units. These will normally have a parent SI unit
        (e.g. calorie has joule as an SI parent). </li>
        <li/>
        </ul>
        <p>Example:</p>
        <pre>
&lt;unit id="units:fahr" name="fahrenheit" parentSI="units:K"
  multiplierToSI="0.55555555555555555"
  constantToSI="-17.777777777777777777"&gt;
  &lt;description&gt;An obsolescent unit of temperature still used in popular
  meteorology&lt;/description&gt;
&lt;/unit&gt;

        </pre>
      </div>
      </xsd:documentation>
      <xsd:appinfo>

      </xsd:appinfo>
    </xsd:annotation>

    <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element ref="description"/>
        <xsd:element ref="annotation"/>
      </xsd:choice>

      <xsd:attribute name="id" type="xsd:string" use="required">
      </xsd:attribute>

      <xsd:attribute name="abbreviation" type="xsd:string">
      </xsd:attribute>

      <xsd:attribute name="name" type="xsd:string">
       </xsd:attribute>

      <xsd:attribute name="parentSI" type="xsd:string">
        <xsd:annotation>
          <xsd:documentation><div class="summary">
        A reference to the parent SI unit (forbidden for SI Units themselves).
          </div>
        </xsd:documentation>
        </xsd:annotation>
      </xsd:attribute>

      <xsd:attribute name="unitType" type="xsd:string">
        <xsd:annotation>
          <xsd:documentation><div class="summary">
        A reference to the unitType (required for SI Units).
          </div>
         </xsd:documentation>
        </xsd:annotation>
      </xsd:attribute>

      <xsd:attribute name="multiplierToSI" type="xsd:decimal" default="1">
        <xsd:annotation>
```

```
          <xsd:documentation>
        <div class="summary"><p>The factor by which the non-SI unit should be multiplied to
        convert a quantity to its representation in SI Units.
        This is applied <b>before</b> <tt>constantToSI</tt>.
        Mandatory for nonSI units; forbidden for SI units</p>
          </div>
          </xsd:documentation>
        </xsd:annotation>
      </xsd:attribute>

      <xsd:attribute name="constantToSI" type="xsd:decimal" default="0">
        <xsd:annotation>
          <xsd:documentation>
           <div class="summary"><p>The amount to add to a quantity in non-SI units to
               convert its representation to SI Units.
               This is applied <b>after</b> <tt>multiplierToSI</tt>.
               Optional for nonSI units; forbidden for SI units.
             </p>
          </div>
         </xsd:documentation>
        </xsd:annotation>
      </xsd:attribute>

    </xsd:complexType>
  </xsd:element>

   <xsd:simpleType name="unitsType" id="st.unitsType">
      <xsd:annotation>
         <xsd:documentation>
           <div class="summary">Scientific units</div>
           <div class="description"><p>Scientific units. These will be linked to dictionaries of units with
         conversion information, using namespaced references (e.g. <tt>si:m</tt>)
          </p>
          <p>Distinguish carefully from <a href="st.unitType">unitType</a>
          which is an element describing a type of a unit in a <a href="el.unitList">unitList</a>
          </p></div>
           <div class="example"><pre>&lt;stm:unitList xmlns:stm="http://www.xml-cml.org/schema/stmml"&gt;

&lt;!-- ======================================================================= --&gt;
&lt;!-- ========================= fundamental types =========================== --&gt;
&lt;!-- ======================================================================= --&gt;

&lt;stm:unitType id="length" name="length"&gt;
  &lt;stm:dimension name="length"/&gt;
&lt;/stm:unitType&gt;

&lt;stm:unitType id="time" name="time"&gt;
  &lt;stm:dimension name="time"/&gt;
&lt;/stm:unitType&gt;

&lt;!-- ... --&gt;

&lt;stm:unitType id="dimensionless" name="dimensionless"&gt;
  &lt;stm:dimension name="dimensionless"/&gt;
&lt;/stm:unitType&gt;

&lt;!-- ======================================================================= --&gt;
&lt;!-- ========================== derived types ============================== --&gt;
&lt;!-- ======================================================================= --&gt;

&lt;stm:unitType id="acceleration" name="acceleration"&gt;
  &lt;stm:dimension name="length"/&gt;
  &lt;stm:dimension name="time" power="-2"/&gt;
&lt;/stm:unitType&gt;

&lt;!-- ... --&gt;

&lt;!-- ======================================================================= --&gt;
&lt;!-- ====================== fundamental SI units =========================== --&gt;
&lt;!-- ======================================================================= --&gt;

&lt;stm:unit id="second" name="second" unitType="time"&gt;
  &lt;stm:description&gt;The SI unit of time&lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;stm:unit id="meter" name="meter" unitType="length"
  abbreviation="m"&gt;
  &lt;stm:description&gt;The SI unit of length&lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;!-- ... --&gt;

&lt;stm:unit id="kg" name="nameless" unitType="dimensionless"
  abbreviation="nodim"&gt;
  &lt;stm:description&gt;A fictitious parent for dimensionless units&lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;!-- ======================================================================= --&gt;
&lt;!-- ===================== derived SI units ================================ --&gt;
&lt;!-- ======================================================================= --&gt;

&lt;stm:unit id="newton" name="newton" unitType="force"&gt;
  &lt;stm:description&gt;The SI unit of force&lt;/stm:description&gt;
&lt;/stm:unit&gt;
```

```
&lt;!-- ... --&gt;

&lt;!-- multiples of fundamental SI units --&gt;

&lt;stm:unit id="g" name="gram" unitType="mass"
  parentSI="kg"
  multiplierToSI="0.001"
  abbreviation="g"&gt;
  &lt;stm:description&gt;0.001 kg. &lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;stm:unit id="celsius" name="Celsius" parentSI="k"
  multiplierToSI="1"
  constantToSI="273.18"&gt;
  &lt;stm:description&gt;&lt;p&gt;A common unit of temperature&lt;/p&gt;&lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;!-- fundamental non-SI units --&gt;

&lt;stm:unit id="inch" name="inch" parentSI="meter"
   abbreviation="in"
  multiplierToSI="0.0254" &gt;
  &lt;stm:description&gt;An imperial measure of length&lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;!-- derived non-SI units --&gt;

&lt;stm:unit id="l" name="litre" unitType="volume"
   parentSI="meterCubed"
   abbreviation="l"
   multiplierToSI="0.001"&gt;
  &lt;stm:description&gt;Nearly 1 dm**3 This is not quite exact&lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;!-- ... --&gt;

&lt;stm:unit id="fahr" name="fahrenheit" parentSI="k"
   abbreviation="F"
  multiplierToSI="0.55555555555555555"
  constantToSI="-17.777777777777777777"&gt;
  &lt;stm:description&gt;An obsolescent unit of temperature still used in popular
  meteorology&lt;/stm:description&gt;
&lt;/stm:unit&gt;

&lt;/stm:unitList&gt;
</pre></div>
        </xsd:documentation>
      </xsd:annotation>
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>

<xsd:annotation>
  <xsd:documentation>
    <div class="heading">Groups (for schema maintenance and re-use)</div>
  </xsd:documentation>
</xsd:annotation>

<xsd:attributeGroup name="actionGroup" id="attGp.action">
  <xsd:attribute name="start" type="xsd:string">
    <xsd:annotation>
      <xsd:documentation>
        <div class="summary">The start time</div>
        <div class="description">
          <p>The start time in any allowable XSD representation of date, time or dateTime.
          This will normally be a clock time or date.</p>
        </div>
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
  <xsd:attribute name="startCondition" type="xsd:string"/>
  <xsd:attribute name="duration" type="xsd:string">
  </xsd:attribute>
  <xsd:attribute name="end" type="xsd:string">
    <xsd:annotation>
      <xsd:documentation>
        <div class="summary">The end time</div>
        <div class="description">
          <p>The start time in any allowable XSD representation of date, time or dateTime.
          This will normally be a clock time or date.</p>
        </div>
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
  <xsd:attribute name="endCondition" type="xsd:string">
    <xsd:annotation>
      <xsd:documentation>
        <div class="summary">
          <p>The end condition</p>
        </div>
        <div class="description">
          <p>At present a human-readable string describing some condition when the
          ac tion should end. As XML develops it may be possible to add machine-processable
          semantics in this field.</p>
        </div>
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>
  <xsd:attribute name="units" type="unitsType">
    <xsd:annotation>
      <xsd:documentation>
        <div class="summary">Units for the time attributes</div>
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>

  <xsd:attribute name="count" type="countType">
    <xsd:annotation>
      <xsd:documentation>
        <div class="summary">Number of times the action should be repeated</div>
      </xsd:documentation>
    </xsd:annotation>
  </xsd:attribute>

</xsd:attributeGroup>

   <xsd:group name="dataGroup" id="gp.dataGroup">
      <xsd:annotation>
        <xsd:documentation>
          <div class="summary">
           <p>A grouping of <tt>list</tt> and <tt>scalar</tt> elements for use by
           other schemas. Experimental.</p>
         </div>

         <div class="description">
           <p>This is to allow non-STM schemas to include a generic "data" component
           in their elements. It is messy and probably should not survive. Better
           extensibility mechanisms should be found.</p>
            <p>Use only within Schemas</p>
         </div>
         </xsd:documentation>
        </xsd:annotation>
      <xsd:choice>

        <xsd:element ref="list"/>
        <xsd:element ref="scalar"/>
      </xsd:choice>
   </xsd:group>

  <xsd:attributeGroup name="sourceGroup" id="attGp.source">
      <xsd:attribute id="att.source" name="source" type="xsd:anyURI">
        <xsd:annotation>
          <xsd:documentation>
            <div class="summary">An attribute linking to the source of the information</div>
         <div class="description">
           <p>A simple way of adding metadata to a piece of information.
           Likely to be fragile since the URI may disappear.</p>
         </div>
         <div class="example">
           <pre>
&lt;list&gt;
  &lt;definition source="foo.html#a3"&gt;An animal with four legs&lt;/definition&gt;
  &lt;definition source="http://www.foo.com/index.html"&gt;
    An animal with six legs&lt;/definition&gt;
&lt;/list&gt;

           </pre>
         </div>
         </xsd:documentation>
        </xsd:annotation>
      </xsd:attribute>
   </xsd:attributeGroup>

   <xsd:attributeGroup name="tit_id_convGroup" id="attGp.tit_id_convGroup">
      <xsd:annotation>
         <xsd:documentation>
        <div class="summary">An group of attribute applicable to most STMML elements</div>
        <div class="description"><p>A group of attribute applicable to most STMML elements. It is
        recommended that IDs are used widely. Titles have no semantics but
        are useful for humans (e.g. a bond angle could be labelled "ortho"
        or "gamma". Conventions are inherited by subelements; the default is the
        current schema.</p>
         </div>
        </xsd:documentation>
      </xsd:annotation>

      <xsd:attributeGroup ref="idGroup"/>
      <xsd:attributeGroup ref="titleGroup"/>
      <xsd:attributeGroup ref="convGroup"/>

   </xsd:attributeGroup>
<xsd:attributeGroup name="tit_id_conv_dictGroup" id="attGp.tit_id_conv_dictGroup">
  <xsd:annotation>
    <xsd:documentation>
      <div class="summary">Addition of the <tt>dictRef</tt> attribute to the <tt>tit_id_conv</tt> group.</div>
    </xsd:documentation>
  </xsd:annotation>
  <xsd:attributeGroup ref="tit_id_convGroup"/>
  <xsd:attributeGroup ref="dictRefGroup"/>
</xsd:attributeGroup>
</xsd:schema>
```