UML2XML – Initial list of requirements (2001-10-09)

1. **Support of formal scheme**

   Which formal scheme for definition of XML-document needs to be supported? At least W3C Schema must be supported. Relax must at least be monitored. DTD-support might be needed depending its importance at the time of releasing this specification. The feeling is that at that time (i.e. probably at least 1 year from now) the importance of DTD will be limited. Whatever "scheme" is used will have to support the "meta-model" that will be defined in this technical specification.

2. **Support of W3C Schema features**

   Which W3C schema features will be supported? Following features can be considered:

   - **Data Types**
     - Simple types - facets to restrict and define new data types. String can be constrained to length, patterns, particular character sets. (*Required*)
       - Facets – pattern, length, min/max length, enumeration, whiteSpace, … (*Required*)
     - Complex types – derived from simple types (contain elements and attributes) (*Required*)
     - Derived types – requires use of xsi:type in instance used in XMI for extending: *(maybe useful)*
     - nil values nillable=true then xsl:nil=true on elements with no content *(used in SWIFT: Required)*
   - **Elements**
     - Global elements –defined individually and then referenced into the content of other elements required
       - Default Values for elements with no content? *(this is supported in UML)*
       - Fixed values? *(this is supported in UML)*
       - Enumerated values – code lists (*Required*)
       - Empty Elements (*Required*)
       - Lists (*Required*)
       - Union Types – zip code or state abbreviation unioned as the allowed content of an element? *(probably not required)*
     - Anonymous types *(probably not required)*/Local Element Declarations *(is this different from anonymoous – probably not required)* – not global, so not defined for use outside the declaring element
     - Substitution groups – define a head element (type) and then other elements in the group substitutionGroup=headelement then in instance element must be qualified with namespace *(can probably be solved in another way: probably not required)*
     - Abstract elements – not used in instance document *(this is supported in UML)*
     - anyElement and anyAttribute *(probably not required)*
   - **Content Models**
     - Mixed content vs element or data only content *(probably not required)*

- anyType *(probably not required)*
- nested choice or Sequence groups (*Required*)
- min/max occurrence (*Required*)
- any order (All groups) *(probably not required)*
- attribute groups *(maybe required to influence "look & feel")*
- element groups *(maybe required to influence "look & feel")*
- deriving by restriction of content of an element content *(maybe min/max ?? – rest probably not required)*

- Annotations
  - Appinfo – machine processable *(probably not required)*
  - Documentation – human processable *(required for documentation version)*
- Namespaces
  - Target namespace – no target namespace equals and XML document *(if used ==> not compliant with DTD – used by SWIFT, XMI)*
  - Unqualified locals – element and attribute form defaults *(probably not required)*
  - Required qualification – form=qualified *(probably not required)*
  - Do we separate common data into its own namespaces (manufacturing, transportation, insurance)? *(possibly required)*
  - ##any, ##local, ##other *(probably not required)*
- Uniqueness *(to be investigated)*
  - Key and xpath=*(XMI uses IDREF, …)*
  - Keys and keyref
- Schema management
  - Import *(probably not required because the XML is generated)*
  - Include – and redefine *(probably not required)*
  - Controlling creation and use of derived types – final= *(is linked to flexibility requirement)*
  - Type libraries *(possibly required to support vertical industries)*
  - SchemaLocation *(need to assess the use of it?)*

3. XML instance features

Which features of XML must be supported (i.e. how should an XML-document that is transported look like).

4. Support of UML patterns

Which patterns of UML must be supported when generating XML.

Class (*probably required*)

inheritance single & multiple

operation & method

attribute (*probably required*)

default & initial values

type

multiplicity

association, ends, and roles (*probably required*)
- types
- multiplicity
- navigability
- aggregation
- qualifiers/keys

ModelElement (*probably required*)
- name
- visiblity
- constraint
- documentation & notes
- stereotypes on classes, attributes, and packages
- tag/values

package

Behavioral (*probably not required*)
- Deployment diagrams
- Collaboration
- Use case
- State diagrams
- Activity diagrams

5. UMM-compliance

The used UML-artifacts must be in line with the UMM definitions and the XML-generation must support documents that are defined through UMM.

6. Core Component compliance

XML-generation must be based on core components and specifically on the way core components will be used in document assembly.

7. Security

Possibility to indicate that some processing is required at a specific place in a document (e.g. encryption, data authentication, enrichment, …)

8. Naming conventions

XML naming conventions: what will element names, attribute names look like. This includes the need to be able to define explicitly an XML-name in the UML model.

9. Character set

Which character set will be supported in the document? Probably it's sufficient to support UTF-8. Need to make distinction between character set for naming conventions and for content.

10. Business Rules

Need to define how and if rules will be captured in UML, in XML schema and in XML instance. Also look into the possibility to have rules in another document that the actual data and being able to reference. Look at link with XSLT.

11. Automatable / Automated support

Need to be able to generate automatically the XML from the UML. Need to be able to reference an instance back to its meaning (I.e. to the underlying model).

12. Performance

Need to make sure that run-time version can be processed in a performant way.

13. Document size

Related to performance. Ability to send partial documents (e.g. only sending what has been changed).

14. Runtime versus Documentation

See above. Also need to specify where in UML the documentation needs to be stored.

15. Generic versus specific tags

How do you model documents; how do you decide to reuse a generic component (e.g. a person) or rather a specific component (e.g. a driver). Has also to do with the use of optionality and the use of inheritance (choice). Need to verify whether the complete requirement can be captured in UML. Possible option is the use of inheritance and packages.

Needs to be supported by majority of products (parsers, …)

16. Strict rules versus flexibility

Possibility to "parameterize" the generation. Need to look with Steve whether other things would be required. Is similar or generic/specific tags. We need to make sure that we find the right balance. Example: use of extensions.

17. Element versus Attribute

When to use element and when to use attribute. Need to define guidelines & recommendations.

18. Style sheet support

Do we need design rules for style sheets as well (to get a standardized look). Style sheets can also be useful for business rules. We might need information in UML to influence the resulting style sheet (e.g. important for small & medium enterprises (SMEs) where they want to display the document).

19. Merging of UML-models (cross-model design)

How to solve possible conflicts if you have to merge models?

20. Look & Feel (e.g. Depth first?, Ordering of declarations & definitions)

What features are required in UML to be able to control the look & feel? Importance of human-readability and human-editable. This is related to how automatable it will be. Have a look at options Near & Far.

21. Maintenance

Rules for maintenance between major releases (e.g. only allowed to add optional elements).

22. Backward/Upward compatibility

Need to offer at least a minimum (= needs to be defined) of compatibility. May need different approaches for major releases. This may impact the need to edit the results.

23. Versioning

What will the versioning scheme be (minor release vs major release vs maintenance). Need to capture version of the spec according to which the XML is generated + version of the tool + version of the (modeling) elements that have been used.

24. Comments

Maybe need to define how to capture comments in UML.

25. Header & envelopes

There needs to be a way to generate the full document (header + payload). Need to look at how to model the header.

26. Change management (requirements from other industries)

see above

27. No transformation

We don't want to have transformation of documents between sender & receiver.

28. One message per schema?

Will we only support one document per schema or also schema containing multiple documents. Also look at the need to split large documents over multiple instances.