GeLaBa – A Framework to Define Classes of XML Documents and to Automatically Derive Specialized Infrastructures

Benoît PIN / Georges-André SILBER
Centre de Recherche en Informatique / Mines de Paris

MINES PARIS
CRI

Why GeLaBa?

Abstract
GeLaBa means *Générateur de Langage à Balises* (in French), i.e. Markup Language Generator in English. It is a framework build upon a language called GML (GML stands for GeLaBa Markup Language) used to define classes of XML documents. Starting from a language definition in GML, GeLaBa provides a collection of tools to automatically generate a complete specialized infrastructure to handle this specific class of XML documents.

Introduction
GeLaBa has been created when our research center began to participate to a project called LHÉO funded by the french ministry of employment. The goal of LHÉO was to create an XML schema to represent and exchange informations about professional formations in an uniform way in France. To suit the needs of all the users of this schema, several things are provided by LHÉO: a DTD, a W3C XML Schema, a complete documentation, a specialized API, etc. The problem of maintaining coherence between all those components while insuring an easy evolution of the language was very tricky. Because no existing tool suited our needs, we created a new language (GML) based on XML and a set of generic tools to automate the creation and maintenance of LHÉO. Those tools and GML became the core of GeLaBa. LHÉO is now completely managed with GeLaBa, including its web site.

GML

The foundation of GeLaBa is GML, a class of XML documents that can describe a subset of all the classes of XML documents. It can represent fewer XML structures than the other schema languages (DTD, W3C XML Schema, Relax NG) but it is more regular, leading to simpler content models. This regular structure is sufficient in many cases and leads to simple but powerful tools. The two main constructions that we allow in GML are: 1) an ordered sequence of unique elements that can be repeated several times (in DTD syntax, content of the form `(a,b+,d?,c*)`), and 2) an element chosen in a restricted set of elements (content of the form `(a|b|c)`).

GML has features that are not present in any schema language. Elements and attributes can be typed with usual data types with various sizes (defined by the user). GML also supports the definition of dictionaries, whose keys can be used as enumerated types in elements or attributes. A dictionary is a simple list of entries, where each entry is a couple (key, value). Documentation is a fundamental aspect of GML: a multilingual documentation can be embedded in every component of the language definition (elements, attributes, dictionaries). Components can also have properties defined by the schema creator, to add extra informations not defined in the GML schema.

```
<definition name="adresse">
  <fullname xml:lang="fr">Adresse</fullname>
  <doc xml:lang="fr">...</doc>
  <element_type>
    <sequence>
      <element name="lignead" min="1" max="3"/>
      <element name="codepostal"/>
      <element name="ville"/>
      <element name="departement"/>
      <element name="region" min="0" max="1"/>
      <element name="pays" min="0" max="1"/>
    </sequence>
  </element_type>
</definition>
```

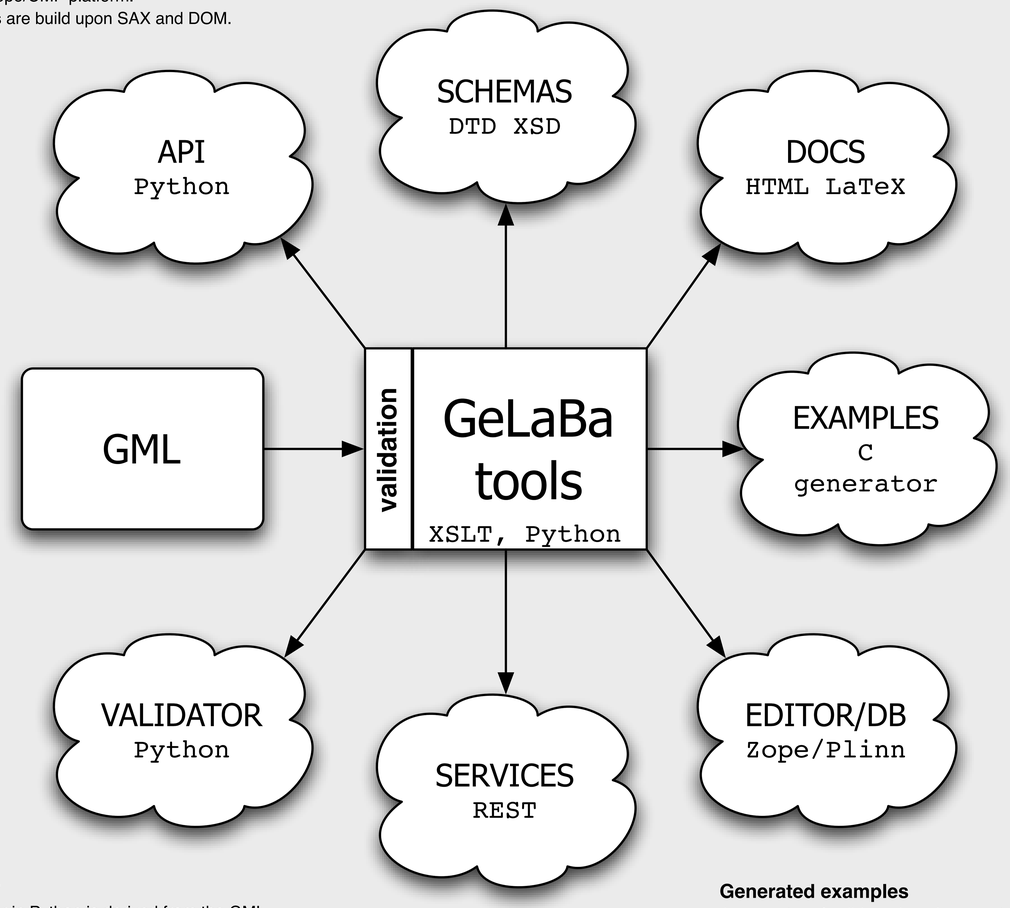
A small example of GML

GeLaBa Tools

API
GeLaBa can automatically create an object oriented Application Programming Interface (API) with a specialized parser to read, modify and write documents in the defined language. The API validates the document and ensure that a modified document follows the schema before writing a file. This API is available under the form of Python classes, compatible with the Zope/CMF platform.
All classes are build upon SAX and DOM.

SCHEMAS
Usual schemas are derived from the schema represented in GML: DTD, W3C XML Schema, Relax NG. Those schemas have not the same power, but they can be used for several level of validations. For instance, the generated DTD can only express unbounded repetitions whereas the generated W3C XML Schemas can express a maximal number of repetitions. The validation can also be done by our ad-hoc validator.

Documentations
Documentations for the defined language are generated from the GML definition, in HTML and LaTeX (PDF).



Validator
An validator in Python is derived from the GML definition. This validator is more powerful and more strict than the generated schemas. It can use the dictionary services to validate dictionary keys contained in documents.

Services
The dictionaries are used to create web services that can be used with a “RESTful” approach.

Editor, Database
GeLaBa generates a complete solution on top of a complete Zope/Plinn application server to edit and store documents in the new language. It provides a persistent storage for the documents, under the form of an Object Oriented Database.

Generated examples
A generator of valid random documents in the defined language is derived from the GML definition. This generator is in C language and very fast, allowing the production of a large amount of XML documents. The content of the elements is random and respects the types or is taken from a companion example file. The purpose of this generator is to propose a bunch of documents in the new language.

Dictionaries and Services

Consider an extract of a dictionary with two letter keys coding the country and values containing the names of the countries (ISO 3166).

```
<entries>
  <entry><key val="AO"/>
    <value xml:lang="en" val="ANGOLA"/>
  </entry>
  <entry><key val="AI"/>
    <value xml:lang="en" val="ANGUILLA"/>
  </entry>
  <entry><key val="BD"/>
    <value xml:lang="en" val="BANGLADESH"/>
  </entry>
</entries>
```

With this dictionary, GeLaBa generates a web service (in fact, a CGI program) that can be used to retrieve keys from values and values from keys. This service is used in the Zope/Plinn infrastructure.

Conclusion

GeLaBa has been implemented in XSLT and in Python for some parts. It is an open source project under a GPL licence.
An interesting observation is that GML is reflexive, i.e. GML can be defined in GML. It means that all the tools of GeLaBa can be used to manipulate GML definitions of new languages: for instance, the web editor generated from the GML definition in GML can be used to create new languages in GML.

Resources

GeLaBa
http://www.gelaba.org

Subversion repository
http://svn.cri.ensmp.fr/trac/gelaba

Plinn
http://www.plinn.org

LHÉO
http://www.lheo.org

Thermodynamic DB Project (uses Plinn/GeLaBa)
http://www.ctdp.org

Acknowledgements

This work has been funded in part by DGEFP (French Ministry of Employment) for the LHÉO project. The authors would like to thanks DGEFP people (Jean-Michel Mazouth, Alain de Lorgeril, Stéphane Akondé) for their support. The authors would also like to thank Bertrand Blanchard from Items Media Concept for fruitful discussions about GeLaBa.