XMLlab[1] : a pluridisciplinary simulation tool based on XML and Scilab

Stéphane Mottelet[2], André Pauss[3]

Abstract

We present an XML-based simulation authoring environment. The proposed description language allows to describe mathematical objects such as systems of ordinary differential equations, systems of non-linear equations, partial differential equations in two dimensions, or simple curves and surfaces. It also allows to describe the parameters on which these objects depend. This language is independent of the software and allows to ensure the perennity of author's work, as well as collaborative work and content reuse. We also describe the architecture of a "compilation chain" allowing to transform the XML files into Scilab scripts.

Keywords : simulation, interoperability

Résumé

Nous présentons un environnement de génération automatique de simulations entièrement basé sur les technologies XML. Le langage de description proposé permet de décrire des objets mathématiques tels que des systèmes d'équations différentielles, des systèmes d'équations non-linéaires, des équations aux dérivées partielles en dimension 2, ou bien de simples courbes et surfaces. Il permet aussi de décrire les paramètres dont dépendent ces objets. Ce langage est indépendant du logiciel et permet donc de garantir la pérennité du travail des auteurs ainsi que leur mutualisation et leur réutilisation. Nous décrivons aussi l'architecture d'une "chaîne de compilation" permettant de transformer ces fichiers XML sous forme de scripts et de les faire fonctionner dans le logiciel Scilab.

Mots-clés : simulation, interopérabilité

1 Introduction

The need to use a simulation tool is in most cases an answer to simple statements : the user has some equations modeling a physical system. He wants to solve them, and if possible to be able to easily change some parameters to see how they influence the results of the simulation and finally save the parameters and the results (e.g. in a format readable by a spreadsheet application).

The educational benefit of using simulations, when an adequate tool is used, is not to be discussed here. But there are very different steps in the development of a simulation. Once the equations are stated, you firstly have to make them fit to a particular software, provided this software is adequate to the disciplinary field of the phenomenon. This first step is not time-consuming compared to the time which is always spent to develop a graphical user interface. The author will spent the greater part of his time to polish the interface, although he could have spent this time to work on another simulation. Moreover, the more the applet will be polished to fit a particular case, the less it will be reusable in another close context. The World Wide Web is a place where a lot of good quality JAVA applets can be found, but these applets are always difficult to reuse in the context of a particular course, because modifying them (when the author makes the source code available) needs abilities in a low level language (JAVA, C++, C), or a high level script language such as the one used by Matlab or Scilab ([Chancelier et al., 2001, Gomez, 1999, Motta Pires and Rogers, 2002]).

This kind of work is the concern of craftsmen, and not of an industrial approach. The author's work is not reusable in general and its perennity is not guaranteed, because the work relies on an application using a proprietary format, and last but not least, the work is exchangeable with authors using the same application (and in many cases the same version).

[1]`http://xmllab.org`

[2]Laboratoire de Mathématiques Appliquées de Compiègne, Département de Génie Informatique, Université de Technologie de Compiègne, BP 20529, 60205 COMPIEGNE CEDEX, FRANCE; `stephane.mottelet@utc.fr`

[3]UMR Génie des Procédés Industriels, Département de Génie Chimique, Université de Technologie de Compiègne, BP 20529, 60205 COMPIEGNE CEDEX, FRANCE; `andre.pauss@utc.fr`

People working on modern documentary applications have already made this reflexion, and this can be seen with the exponential growth of the number of applications of the XML markup language ([Bray et al., 2004]). In the field of simulation applets this reflexion has hardly begun. One can cite, in the field of biology and chemistry, the work of a consortium of academic people and authors of simulation software which has lead to `sbml`, and exchange markup language modeling biological and chemical systems (with kinetics) using XML ([Hucka et al., 2003, Hucka and Finney, 2003]).

Another project is `xmds`, a tool also based on XML allowing to generate Scilab, Matlab or C++ code (but without any graphical user inteface) allowing to simulate deterministic or stochastic systems ([Collecutt et al., 2001]). Compared to our approach, which will detailed in this paper, the weakness of this tool is a lack of structuration in the description language (the level of structuration is not deep enough compared to what XML allows to do).

In the scope of the XMLlab project, we have chosen to show the benefits of an approach where the content and the form are well differentiated and are the concern of different people :

The content The equations of the phenomenon, their description, the associated parameters and their thematic organization. The description of the content is the concern of the author.

The form The graphical user interface, the various "widgets" and menus which reinforce the user-friendliness of the final applet, and the visualization tools (e.g. curves and animations). This part of the applet code is the concern of a high level developer, or the concern of a tool able to generate this code automatically from the description of the content. This is the option we have chosen.

The choice of adequate numerical methods is another problem. The author is not necessarily able to make this choice himself, that's why this choice has to be done automatically, knowing which method is the most adequate to solve a given type of equation.

The purpose of the XMLlab project was not to define a description language by its own, hence we have chosen a particular "target" application in the early phases of the project. This application is Scilab, an *open source* software developed since 1990 by researchers of INRIA and ENPC. Our goal was to develop a complete "compilation chain", allowing to transform the source XML documents into executable scripts interpreted by the target application. Moreover, the choice of Scilab is motivated by the fact that this software allows to use the Tcl/Tk script language ([Ousterhout, 1998, Ousterhout, 1994]) to generate graphical user interfaces.

2 A preview of the structure of an XMLlab simulation

A simulation can be divided in a certain number of conceptual elements : parameters, mathematical models of objects (time-dependent or not) and finally a display element to output the results of the simulation.

2.1 Parameters

They are the parameters of the phenomenon and of the mathematical model. The goal is to allow the user of the simulation to make them vary by means of the interface which will be generated by the compilation chain. It has to be possible to simply specify if the value of the parameter is seen in the interface, but non modifiable by the user. There must also exist hidden parameters, for internal use only.

Scalars and matrices In an XMLlab simulation the parameters can be scalars or matrices. The minimum and maximum value of a scalar can be given, the type of "widget" to use (a slider, or a simple entry field where the user can modify the value). Each parameter must have symbolic name which can be reused in the description of the mathematical model, and a default value, which will be used for the first run of the simulation.

Parameter groups The parameters can be grouped into sections, e.g. for an ordinary differential equation, the user may want to differentiate the physical parameters of the phenomenon from the resolution parameters (final time, number of discretization steps, etc.). This logical structuration can be then used to graphically structure the interface.

Databases The user can store many instances of a parameter group. This allows, e.g. in chemistry, to build a small database of different acids and alkali, by storing their parameters (acidity constants, charge, etc.). The database can then be used to generate a menu allowing to choose a given parameter group in the interface.

2.2 Mathematical models

We now deal with the equations of the phenomenon to be simulated. There are elements of different levels.

Domains The most simple describe intervals of $\mathbb{R}$ or domains of $\mathbb{R}^2$. They are simple closed intervals of the type $[a, b]$ (where the bounds can of course depend on parameters described in the previous section) or two dimensional domains. The latter can be rectangles defined by a Cartesian product of two intervals, or general domains defined by the form of their boundary by parametric curves (we will discuss curves in the next item). The user can precise the way these domains have to be discretized, if applicable (number of discretization points, linearly or logarithmically).

Curves and surfaces Non-parametric curves can be described like this,

$$y = f(x), \quad x \in [a, b].$$

This definition reuses an interval. This way, it is possible to define many curves referring to the same interval.
Parametric curves can be also defined like this,

$$\left\{\begin{array}{rcl} x & = & f(t), \\ y & = & g(t), \\ t & \in & [a, b]. \end{array}\right.$$

The surfaces can also be of parametric or non-parametric type, namely

$$z = f(x, y), \quad (x, y) \in \mathcal{D},$$

where $\mathcal{D}$ is a domain of $\mathbb{R}^2$, or

$$\left\{\begin{array}{rcl} x & = & f(u, v), \\ y & = & g(u, v), \\ z & = & h(u, v) \\ (u, v) & \in & \mathcal{D}. \end{array}\right.$$

The parameters defined in the previous section can be used at any level, in the definition of domains or in the equations themselves.

Ordinary differential equations One can describe systems of ordinary differential equations, e.g.

$$\left\{\begin{array}{rcl} \frac{d}{dt}x(t) & = & f(x, y, t), \\ \frac{d}{dt}y(t) & = & g(x, y, t), \\ t & \in & [a, b], \end{array}\right.$$

with given initial conditions $x(a) = x_a$ and $y(a) = y_a$. XMLlab allows to keep the natural description, without having to reformulate each unknown $x(t)$ or $y(t)$ as the element of a vector $X(t)$ with $x(t) = X_1(t)$ and $y(t) = X_2(t)$. The chosen description model consists in :

- A time interval (here $[a, b]$)
- A list of "states". For each state (here x or y), its time-derivative and its initial value are given.
- A list of "outputs". They are observations which can be computed by using the states, e.g. $z(t) = x(t) + y(t)$.

Non linear equations General systems of non-linear equations can be described, namely

$$\left\{\begin{array}{rcl} f(x, y, z, t, \cdots) & = & 0, \\ h(x, y, z, t, \cdots) & = & 0, \\ & \cdots & \end{array}\right.$$

or non-linear equations depending on a parameter varying in an interval, of the form

$$\left\{\begin{array}{rcl} f(x, y) & = & 0 \\ x & \in & [a, b], \end{array}\right.$$

allowing to take into account some curves defined by an implicit equation of the type $f(x, y) = 0$. This kind of equation is used in the modeling of acid-alkali titration.

Partial differential equations XMLlab allows to describe partial differential equations of diffusion type, namely

$$\left\{\begin{array}{c} -\operatorname{div}(P \operatorname{grad} u)(x) + c(x)u(x) = f(x), \ x \in \Omega, \\ +\text{boundary conditions} \end{array}\right.$$

The domain Ω is described from its boundary (parametric curves, defined earlier). We will give some details on the numerical methods in the next section. Here again, the parameters can be used at any level, in the definition of the domain Ω or in the physical data (diffusion matrix P, source term $f(x)$, proportional coefficient $c(x)$).

2.3 Results display

We use a classical hierarchical description, using windows and systems of axes, where the user just has to precise what he wants to display by making reference to objects defined in the "Mathematical models" section.

Windows A window contains systems of axes. The user just has to specify how they have to be placed if they are more than one (the window is divided within its height and width).

System of axes The user has to specify if the system is two or three dimensional. Each system of axes contains some references to what has to be represented.

Objects to be represented Reference can be made to a curve, to a surface or to the state of an equation, by means of its symbolic name. The chosen structure allows to greatly simplify the number of different elements. For example, a surface can be referenced in a two or three dimensional system of axes. In a three dimensional system a perspective projection is used, although in two dimensions we use a pseudo-color planar representation. In both cases, the reference to the surface is made identically, only the the "parent" context is changing.

```
<?xml version="1.0"
      encoding="ISO-8859-1"?>
<!DOCTYPE simulation
          SYSTEM "simulation.dtd">
<simulation>
  <header>
   ...
  </header>
  <notes>
     ...
</notes>
  <parameters>
     ...
  </parameters>
  <compute>
     ...
  </compute>
  <display>
     ...
  </display>
</simulation>
```

FIG. 1: The outline of a simulation showing the high-level elements.

3 An example simulation

We give on the figure 1 the skeleton of a simulation document. We will now explain with details how to build this document to describe a small simulation.

We consider the pendulum depicted on figure 2. We make the hypothesis that the line connecting the sphere of mass M to the rotation axis is of negligible mass compared to M. We measure the deviation of the pendulum from the stable vertical equilibrium position by

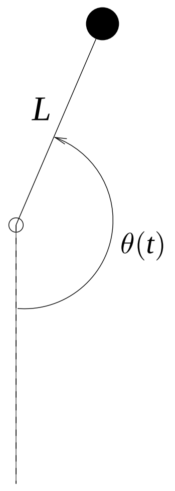


FIG. 2: The pendulum

the angle $\theta(t)$ positively measured as indicated on figure 2. If one applies the relations of dynamics for bodies under rotations, we obtain the following ordinary differential equation :

$$\begin{cases} \ddot{\theta}(t) & = & -\dfrac{g}{L}\sin\theta(t), \quad t \in [0,T] \\ \theta(0) & = & \theta_0, \\ \dot{\theta}(0) & = & 0. \end{cases}$$

The value of θ_0 gives the initial angular deviation of the pendulum, and we consider that the initial angular velocity is zero. If θ_0 is small, $\theta(t)$ can be approximated by

$$\phi(t) = \theta_0 \cos\left(\sqrt{\frac{g}{L}}t\right).$$

We will show how to traduce all this into XML elements, by proceeding in the order of the elements represented in figure 1.

3.1 Parameters, `parameters` elements

We have to take into account the physical parameters g, L, θ_0, and the final time T. The following XML code fragment allows to describe these parameters :

```
<parameters>
  <title>Parameters of the
         pendulum</title>
  <scalar label="L" unit="m">
    <name>Length of the pendulum</name>
    <value>1</value>
  </scalar>
  <scalar label="g0" unit="ms^-2">
    <name>Gravity</name>
    <value>9.81</value>
  </scalar>
  <scalar label="theta_0" max="3.141"
          min="0.01" unit="rad"
          increment="0.001"
          widget="slider">
    <name>Initial angle</name>
    <value>0.1</value>
  </scalar>
</parameters>
<parameters>
  <title>Resolution parameters
  </title>
  <scalar label="T" unit="s">
    <name>Final time</name>
    <value>2</value>
  </scalar>
</parameters>
```

Each pair of `parameter` elements allows to group parameters. The `scalar` element represents a scalar parameter, containing its full name in the `name` element and its initial value in the element `value`. A `scalar` element has a mandatory `label` attribute containing its symbolic name, which will eventually used in the `value` elements of other elements. The `widget` attribute with value `"slider"` means that the parameter will be modifiable by a slider.

3.2 Mathematical models, `compute` element

Here are the fragments corresponding to the description of the $[0, T]$ interval,

```
<defdomain1d label="t" unit="s">
  <name>time</name>
  <interval>
    <initialvalue>0</initialvalue>
    <finalvalue>T</finalvalue>
  </interval>
</defdomain1d>
```

and to the description of the differential equation :

```
<ode label="pendulum">
  <refdomain1d ref="t"/>
  <states>
    <state label="theta"
           unit="rad">
      <name>Real solution</name>
      <derivative>theta_point
      </derivative>
      <initialcond>theta_0
      </initialcond>
    </state>
    <state label="theta_point"
           unit="rad/s">
      <name>Derivative of
            the angle</name>
      <derivative>-g0/L*sin(theta)
      </derivative>
      <initialcond>0</initialcond>
    </state>
  </states>
  <outputs>
    <output label="theta_lin">
      <name>Harmonic solution</name>
      <value>theta_0*cos(sqrt(g0/L)*t)
      </value>
    </output>
  </outputs>
</ode>
```

The `ode` element (ordinary differential equation) contains an empty element `refdomain1d` referring to the $[0, T]$ interval (defined earlier by the `defdomain1d` element, referred by the attribute `ref`), and thus defining the symbolic name of the integration variable, a `states` element containing the description of each state (θ and $\dot{\theta}$) in a `state` element. Each `state` element contains the name of the state, its time-derivative `derivative` and its initial value `initialcond`. The last element `outputs` in `ode` is a list of outputs. Here, the sole output explicitly depends on time but does not depend on the states. Each state or output has a mandatory attribute `label` which will be referred in the display section.

3.3 Display of results, `display` element

We want to superimpose two curves in the same axes system :

```
<display>
  <window>
    <title>Comparison of the two
           solutions</title>
    <axis2d>
      <drawcurve2d ref="theta"/>
      <drawcurve2d ref="thetalin"/>
    </axis2d>
  </window>
<display>
```

The `display` element contains only one `window` element, containing itself a two dimensional system of axes. The two `drawcurve2d` elements within the same `axis2d` mean that the curves of θ and its harmonic version will be superimposed.

3.4 Remarks

The different structuration possibilities are constrained by a DTD (Document Type Definition), allowing an *a posteriori* validation of a simulation, or can be used to constrain the edition of a simulation by means of an XML editor. The figure 3 shows the view that the user can have of its XML file.

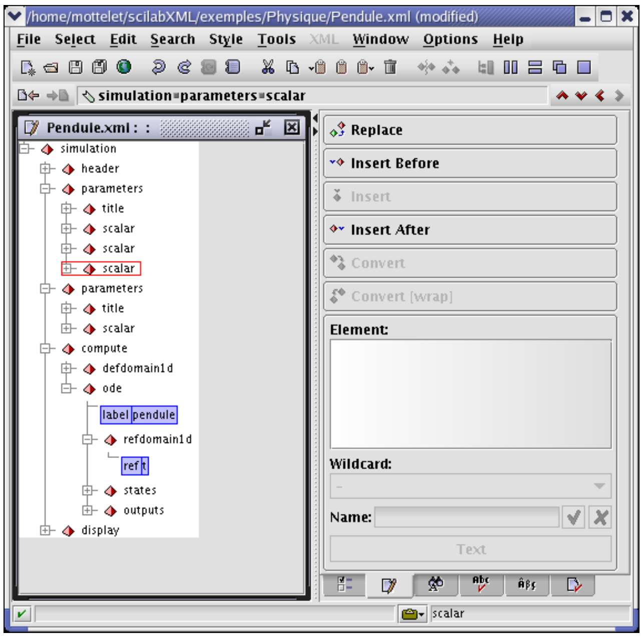


FIG. 3: The XML file describing the simulation of the pendulum, seen in the XXE editor, developed by PIXWARE, `http ://www.xmlmind.com/xmleditor`

Within all of the above mentioned elements, some have a particular status. The `name` and `title` elements can appear several times, with a different `lang` attribute (`french` or `english` in XMLlab 1.3). The goal is to be able to generate from the same XML file two different versions of the "compiled" applet, the language to use being specified as a compilation option.

The `header` element contains some meta-data such as the name of the author and some keywords. The `notes` element can appear several times with a different `lang` attribute and allows to write a few paragraphs of text allowing to describe the simulation and/or to give some help to the user.

4 The compilation chain

4.1 Some details

The compilation chain is entirely based on an XML technology : it is based on XSL transformations specified in XSL stylesheets (eXtensible Stylesheet Language). These transformations are applied to the simulation file by an "XSL processor". The XSL technology is well known to allow the display of dynamic HTML on the World Wide Web, but it is also well fitted to the automatic generation of scripts. We are here particularly interested in the script langage of Scilab or Matlab, and the script language Tcl/Tk, allowing to describe graphical user interfaces.

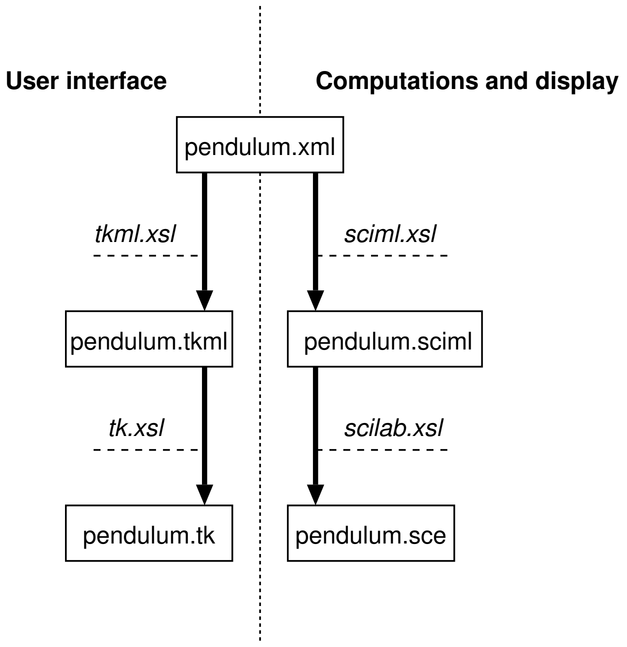


FIG. 4: Diagram of the compilation chain. The arrows represent the XSL transformations, and the italic names the associated stylesheets.

The different phases of the compilation are outlined on the diagram depicted on figure 4. In the bottom of the diagram, the `pendulum.sce` file is the Scilab script containing all the computation code, and the display of results. The `pendulum.tk` file contains the Tcl/Tk code of the interface. The diagram illustrates the fact that the transformation is not direct and needs an intermediary step ; this particular point needs an explanation.

To allow an easy maintenance of the xsl stylesheets, and especially to allow a smooth change of target languages (Scilab and Tcl/Tk), we have used “pivot” XML dialects : the code is generated in a two-step process. We use XSL transformation to translate XML input into a pseudo-Tcl/Tk and pseudo-Scilab syntax that IS XML. Then we use a second pass to serialize the pseudo-language XML into the target language. The advantage here is that the second pass captures all the complexities of formatting clean code (syntax) while the first pass concentrates on the logical aspects of the translation (semantics).

tkml As far as the interface Tcl/Tk code is concerned (left side of the diagram), the intermediary file `pendulum.tkml` is an XML file containing a logical description of the interface. This file contains the description of the different widgets (buttons, etc.) and their placement with respect to each other. Then, this intermediate file is finally translated in Tcl/Tk by means of a last XSL transformation.

sciml For the Scilab code generation, we proceed in the same manner : we first generate an intermediary file `pendulum.sciml`, written using a pseudo-Scilab markup, and then transform this file to Scilab code with a last transformation.

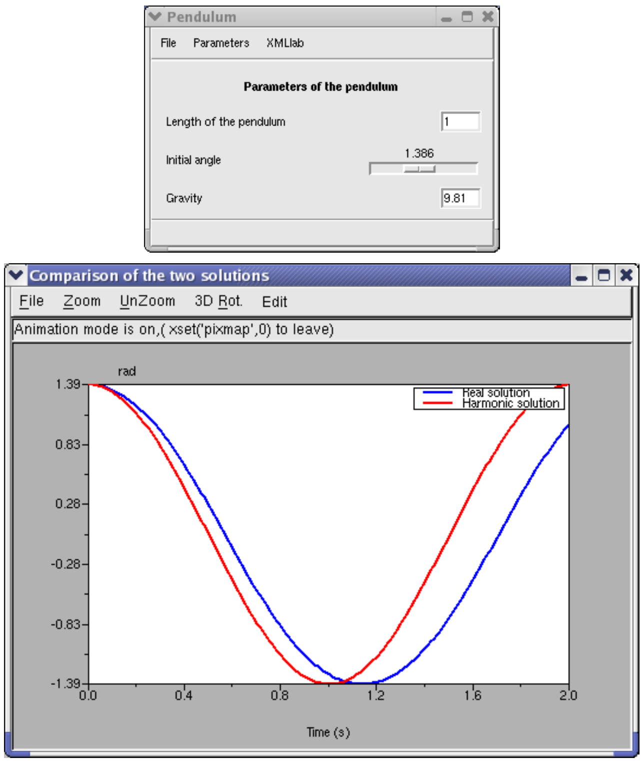
Two ways of distribution can be used : the two Tcl/Tk and Scilab files can be later used without using the XML source and the compilation chain (thus protecting the author’s work). However, it would be more profitable to the community to release the XML source.

The whole compilation chain together with Scilab (except the XML editor), uses only open-source software packages (`xsltproc` of the Gnome project, Tcl scripts), and works on any platform (Windows, MacOs X, Unix).

4.2 Comments on the example

We make some comments on the figure 5.

User interface The window of the interface has a central space where appear the widgets allowing to change parameters. The `Parameters` menu contains two items named “Parameters of the pendulum” and “Resolution parameters” corresponding to the two parameters groups specified in the XML file. The user just has to select a given item to display the corresponding parameter group.
The `File` menu contains two interesting items : “Save a session” and “Load a session”. They allow to save the values of parameters in a text file, and to load them later. It allows to resume a working session (otherwise the Scilab script always starts with the initial values of parameters).
The `XMLlab` menu gives some information on the XMLlab project (“About XMLlab” item), and some information on the simulation, extracted from the `header` element : name of the author, date and eventual notes describing the simulation (“About this simulation” item).



Animation mode is on,(xset('pixmap',0) to leave)

FIG. 5: The interface and the display window for the pendulum simulation

Graphical window The legend of the two curves is taken from the `name` elements within the states `theta` and the output `theta_lin`. The abscissa label is the name of the time variable `t`, and the ordinate label is the unit (`unit` attribute of element `<state label="theta"`).
This window belongs to Scilab, thus the user has access to the usual menus allowing to save (e.g. in EPS format) or print the figure.

The user has always the possibility to have access to all variables of the simulation from the Scilab’s command line (parameters and results of the simulation), which remains available during the simulation.

Remark on performances For this particular example (a system of two scalar ordinary differential equations), the computation time is negligible compared to the time elapsed by drawing the curves, and hence the user can see the immediate effect of the initial angle on the synchronization of the curves. For more intensive examples (e.g. resolution of a partial differential equation), the response time can be greater, but the reactivity of the system is always good, even on a lightweight system (1Ghz Pentium). For these reasons Scilab is really appreciated, because the built-in functions are well optimized (linear and nonlinear equations solving, differential equations, sparse matrix algebra, vectorization of elementary functions for arrays, etc.). Moreover, Scilab loads in a negligible time (compared to the important startup time of recent versions of Matlab, because of the use of JAVA for the user interface).

5 Trends and conclusions

The different parameters types and the mathematical objects presented in section 2 are already present in XMLlab 1.3, but XMLlab is a work in constant progress, and many extensions and improvements are necessary.

However, we think that the choices we have made are good, especially concerning the structuration of the simulations and the architecture of the compilations chain. The needed development time to add a new type of equation is always modest. For example, the `stationary-pde` element, allowing to describe an elliptic partial differential equation (extension of the DTD and associated XSL stylesheet sections), has been developed in two days (we rely on a Scilab "PDE toolbox").

Future developments The priorities are the following
- Discrete time systems simulation,
- Stochastic systems,
- Animations.

As far as the edition of the XML files is concerned, we plan to use "Cascading Stylesheets" allowing to edit XML files in a very user-friendly way in the XXE editor (see [Bos et al., 1998]). We also plan to migrate the actual DTD to an XML Schema ([Thompson et al., 2000]), to allow some enhancements in the control of validity of the different data types contained in elements and attributes.

Disciplinary fields Simulations have been written in the fields of biology (microbial and enzymatic kinetics), physics (pendulum, oscillators, two-body system, Poisson equation, etc.), chemistry (Acid/Alkali equilibriums, chemical kinetics), chemical engineering (ideal reactors). We wish to extend the scope of XMLlab to new disciplinary fields.

Use of XMLlab in the chemistry courses XMLlab has been used for 3 years in chemistry courses at the UTC (150 students by semester), under the form of demonstrations during the course, and during the labs, together with experimental acid/alkali titrations. With the help of simulation the students interpret the experimental curves and are able to answer reasoning questions. The software is also available in the UTC intranet to allow students to improve their understanding of phenomenons. A sample survey has been made by e-mail at the end of each semester and allowed us to conclude that :
- The software is easy to use, and students do not encounter major technical problems.
- The software allows a better understanding of complex phenomenon and acid/alkali equilibrium.
- High level labs assistants are needed (they must be trained on the software before the labs).
- The software is rarely used by the students outside the labs hours.

License XMLlab is available at the address `http://xmllab.org`, under the form of a Scilab toolbox under the GPL license. We hope that a lot of people will contribute and request some new features, which will help us to make XMLlab fit to new disciplinary fields.

Acknowledgments

This work has been financed by the groupment "Evaluation of New Technologies in Education" of the regional Council of Research from Picardie. IUFM's teachers and Jules Vernes University of Picardie Professors have contributed efficiently to the trandisciplinarity of this project.

References

[Bos et al., 1998] Bos, B., Wium Lie, H., Lilley, C., and Jacobs, I. E. (1998). ascading stylesheets, level 2, css2 specification. *Available via the World Wide Web at http ://www.w3.org/TR/1998/REC-CSS2-19980512.*

[Bray et al., 2004] Bray, T., Paoli, J., and Sperberg-McQueen, C. e. a. (2004). Extensible markup language (xml) 1.0. *Available via the World Wide Web at http ://www.w3.org/TR/2004/REC-xml-20040204.*

[Chancelier et al., 2001] Chancelier, J.-P., Delebecque, F., Gomez, C., Goursat, M.and Nikoukah, R., and Steer, S. (2001). *Introduction à Scilab*. Springer, Paris.

[Collecutt et al., 2001] Collecutt, G., Drummond, P., Hope, J., and Cochrane, P. (2001). Extensible multi-dimensional simulator (xmds). *Available via the World Wide Web at http ://www.physics.uq.edu.au/xmds/index.html*.

[Gomez, 1999] Gomez, C. E. (1999). *Engineering and scientific computing with scilab*. Birkauser, Boston.

[Hucka and Finney, 2003] Hucka, M. and Finney, A. (2003). Systems biology markup language : Level 2 and beyond. *Biochem. Soc. Trans.*, 31 :1472–1473.

[Hucka et al., 2003] Hucka, M., Finney, A., Sauro, H., Bolouri, H., Doyle, J., and al. (2003). The systems biology markup language (sbml) : a medium for representation and exchange of biochemical netwok models. *Bioinformatics*, 19 :524–531.

[Motta Pires and Rogers, 2002] Motta Pires, P. and Rogers, D. (2002). Free/opensource software : an alternative of engineering students. *Procceding of the 32nd ASEE/IEEE Frontiers in Education Conference, November 6 - 9*.

[Ousterhout, 1994] Ousterhout, J. (1994). *Tcl and the Tk toolkit*. Addison-Wesley.

[Ousterhout, 1998] Ousterhout, J. (1998). Scripting : Higher-level programming for the 21st century. *IEEE Computer magazine*, March :23–30.

[Thompson et al., 2000] Thompson, H., Beech, D., Maloney, M., and Mendelsohn, N. (2000). Xml schema part 1 : Structures. *Available via the World Wide Web at http ://www.w3.org/TR/xmlschema-1*.