UDC 004.738.5: 519.85

GENERAL FRAMEWORK FOR PUBLISHING OF MATHEMATICAL WEB APPLICATIONS

Syedov A.O.
Deputy Director of Department of Innovations and Technologies' Transfer, Ministry of Science and Education of Ukraine.
Klenov D.M.
Laboratory of development and implementation of pedagogical software Research Institute of Information Technologies Kherson State University

This system would allow an easy way to bring mathematical applications to the Internet. Although the automated generation of PhP scripts is a simple task, there is no system that would allow bringing mathematical application written in any locally based system to the web. The main part of such system is simple XML-based language that fully describes source mathematical application.

Keywords: *XML, web publishing, mathematical programming, framework.*

This project is guided by the Prof. Wolfgang Schreiner of the Research Institute for Symbolic Computation (RISC), Johannes Kepler University, who has decided the main problem of the project. W. Schreiner developed the first version of XML-based language that is the main part of entire project, and the Discrete Wave Turbulence web application, that is the best example of how the automatically generated web-applications will actually look like.

Statement of the problem in general outline and its connection with important scientific and practical tasks;

The main goal of entire project is to develop an easy way for publishing of mathematical software into the web. A source mathematical application can be written in any programming language C++/C#/Java/APS/Prolog etc. By this moment there also are a lot of systems of computer algebra like Wolfram Mathematica, Maple, MathLab, they are providing an appealing GUI(Graphical User Interface) for executing the computations and presenting a result.

However to use such systems like, for example Mathematica, you have to install it locally on your computer. Mathematical programmers mostly have deal only with specified part of such systems or programming languages that can be used to implement some methods they are working at. The main problem of such way is portability.

To solve this problem it is obviously that mathematical methods must be implemented as web-applications. Our system will allow mathematical programmers to get such web implementation without actually PhP coding. The only thing a mathematical programmer will have to do is to upload his math application written in any language he likes most, with some description of it, and he will receive an automatically generated web-application that implement his method.

Analysis of the latest researches and publications, in which solution of a problem is initiated and which serve as a background for the research;

Of course nowadays there are a lot of various ways to generate simple web pages automatically. As locally based application shall be transformed into a web service, there is no actual need of using "heavy" web applications like for example Java applets or Silverlight applications, of course if the implementation of mathematical method do not require a building of 3D models or some sort of dynamic graphical drawing. There is no problem with automated generation of simple HTML pages with minimum Java-scripting. If current implementation would require a complex graphics then system must also generate a Java applet. This can be easily done through ANTLR(Another Tool for Language Recognisation) for example that can generate any type

of Java/C++/C#/Action Scripts applications using a grammar. So there is no problem of automated generation of simple web interfaces and even Java applets automatically

Sorting out of unsolved aspects of the general problem, which the article covers;

The main problem is "How to bind the application written as locally based to that automatically generated web interface?" Not every language can create cross-platform applications.

Statement of the object of article (problem definition);

It is obvious that there must be some description of mathematical program to make the transformation into a web service possible. Such description has to include all required information and would be developed as a simple XML-based language, containing various sets of nodes divided into three sections:

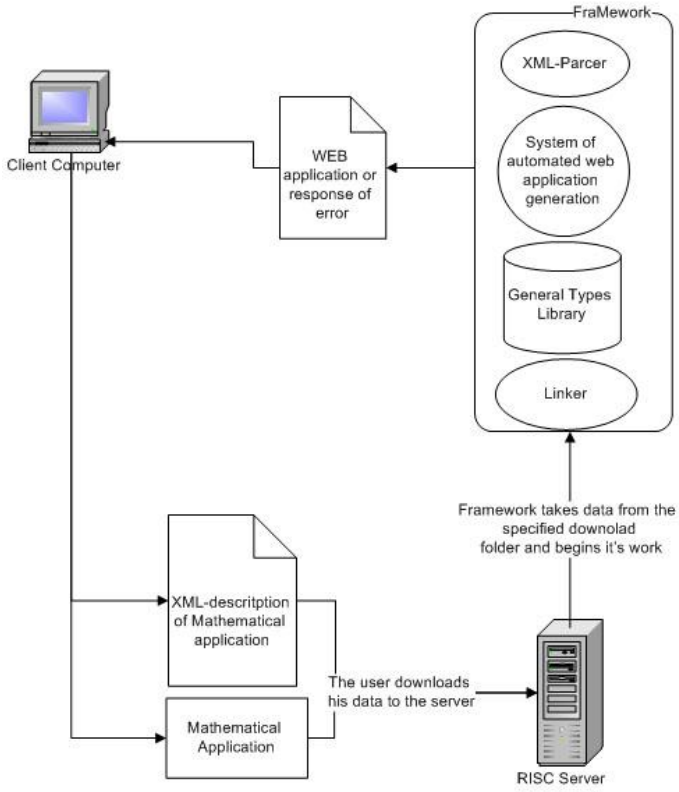
1. Nodes containing information about the language that was used to make a source mathematical application, with descriptions of all additionally included libraries (in the worst case this libraries should also be uploaded to the server alongside with application itself), source architecture description(there is no guarantee that application written under 32-bit architecture will run under 62-bit architecture in the same way, and there is almost no possibility of running 62-bit application under 32-bit system), target architecture description, and operation system version that the source mathematical application was based for(Windows/Mac/Unix).

2. Nodes containing information about all of input/output parameters, as the set of variables. Every variable will have a value of concrete type, taken from the General Type Library. Such library has to be created to make access to the variables types of various computer algebra systems or computer language we are going to support. Every time we add new language to our Framework, we'll have to update this Library. Variables will have unique names, used to create links between nodes and parts of the web interface. The second part of this section is the nodes representing the computations performed by the mathematical program. It is very important to store the computation results, to make graphical buildings much easier, and to allow an option of saving them on client computers.

3. Nodes containing a simple description of user interface.

Summary of the basic material of research with complete argumentation of the obtained scientific results;

Our system of automated web application generation must include all compilers and interpreters of language and mathematical systems we are going to support. It is obvious that if we have a Mathematica script, there have to be a Wolfram Mathematica to execute it. More than that, we've discovered that a concrete example (Discrete Wave Turbulence web application that was created manually by W.Schreiner and can be found on the CENREC portal, this web application is also the best example of how the automatically generated web applications would look like) implemented in Mathematica5.2 does not completely works with Mathematica6.0. This means that we also must have not only all the systems installations on the server, but also a lot of versions of them (this is where we also may face to the version compatibility of systems, will for example Mathematica5.2 work fine if Mathematica6.0 is also installed (I just took Mathematica as preliminary example, this computer algebra system do not face such problem under Linux)). The third part of that system is some sort of a framework that will parse the XML-based algorithm description, and send all necessary data to the automated system of web user interface generation. After the web interface is generated, framework will have to check the source mathematical application itself no matter of the system used to make it, and link this application to the web interface. In our case linking means "wrapping" of it as the web service and binding the invoking of methods to the parts of automatically generated web interface. On the Pic.1 below you can see the basic diagram of this system.



Pic.1

As we can see user must upload his mathematical application alongside with XML-based description to the specified folder on the server. This action is to be done via a simple web interface (Of course it's obvious that it would be much better to make also a web interface for generating the XML description file and some sort of WebFtp for uploading, but for the first time it will be quite enough to create this file manually). After the upload will be completed correctly the framework takes it's part, description is taken to the parser that gathers all information about the mathematical application.

The main problems will begin after the application is recognized by the system. Pre-compiled libraries (for example C++ libraries) are not always completely portable. The RISC server where the system will be based is running under the operating system Debian Linux. If this library was first compiled under any version of Microsoft Windows or MAC OS, there is no guarantee that it can be running under Linux in the same way. Of course there are a lot different possibilities of running Windows applications under Linux, for example Wine. But any of such ways may not offer 100% efficiency. In our case we'll have to use all of them. On the other hand if application is given as source code, this task will become much more simplified. Source code can be re-compiled under Linux, but still the portability problem remains, if application uses some platform-specific libraries that are available for Windows only for example. (As I said before such library should also be uploaded to the server and run under the Wine). If application was written in Microsoft Visual Studio beginning with .Net version, then we'll have to use MONO (.NET Framework implemented under Unix). Every variable declared in XML is description specifies one input/output parameter of the algorithm. Each variable is of a concrete type. This type is taken from the General Types Library that is the part of a Framework. This Library contains description of all types available in supported languages and mathematical systems.

System of automated web interface generation is the most easy to develop part of entire project. It only works with the user interface section of XML description and generates HTML (or probably some times with minimum java-script) web interface according to it. The main problem is that every User Interface element must have an attribute "name" that is the name of a variable created before.

Final part of the Framework is Linker. This will be the most complex scripts-based environment whose task is to link the mathematical application uploaded by the user to the automatically generated web interface. Linker uses variables only of those types that are described in the General Types Library.

All of the computations are done on the RISC server, and in future the possibility of saving the computation results on the client computer is also has to be implemented.

Conclusions of the given study;

As we can see the most important part of a project is the XML description, containing all information about mathematical algorithm, in a simple meta-language. I have already described three types of nodes that are anyway to be implemented. Below you can see a preliminary example of such language.

```
--main information nodes
<SMWML >
<language=”C++” compiler=”gcc” platform=”Windows”>
<library>
     DiscreteWaveTurbulence`SolutionSet
  </library>
</language>
--varset nodes
<workflow>
<varset id="input">
  <var id="domain" format="integer" />
 </varset>
</workflow>
--user interface
<GUI>
<page>
 <frame name="inputarea">
  ...
  <form id="input"> % form bound to varset input
    <input name="domain" > % bound to var input
    <input type="submit" value="Create Solution Set">
  </form> ...
  <frame name="set"> % inline frame bound to varset set
</page>
</GUI>
</SMWML>
```

The entire document is to be bracketed between the nodes <SMWML></SMWML> that specifies that this document contains description of mathematical algorithm. There is a possibility that there may be not a single XML file in the specified folder, the mathematical application itself may use XML for some inner purposes. But only one XML document must be tagged with SMWML upper tag. This means Simple Mathematical Workflows Description Language as this language also describes the flows of data between the parts of mathematical application in some point of view. The second level nodes are tagged with <language></language> <workflow></workflow> <GUI></GUI>. The first section describes the general information about the mathematical application: what language was used to write an application, what system was used to make it. If application was written for example in Mathematica than we must say what was the version of Mathematica. And of course what was the operation system. As we use Linux as the server base, we will face with porting problems when mathematical application was written under Windows. I’ve already told about the possibility of using systems like Wine for solving this problem. As we can see this part also contains description of all libraries used by mathematical application.

The second section is the workflow section. It contains all description of variables. Variables are joined into varsets. Parameter “id” is a unique name that will be used for linking the program to the web interface.

The User Interface Section contains preliminary description of web interface. As we can see these nodes are not the exact HTML tags. They are only a simple description. For example “button”, “textbox”, etc. This part is used to generate web application automatically, and link it with variables. The only thing that is important is the id parameter. Linker will use this parameter value for linking all the parts together. A variable with id=”domain” will be linked to any user interface part that also has id=”domain”.

Prospects of the further research work in this area.

Once completed our system will follow the general trend in computer science which turns away from “stand alone” software (that is installed on local computers and can only be executed on these computers via graphical interface) and proceeds towards service-oriented software (that is installed on remote server computers and transforms each method into a service that can be invoked over Internet via standardized Web interfaces). The mathematical methods will become remote services that can be invoked by clients without requiring of a local software installation.

BIBLIOGRAPHIC REFERENCES

1. Job Description language – comprehensive description language for specifying program (job)requirements is JDL which is used in some grid middleware [https://edms.cern.ch/file/555796/1/EGEE-JRA1-TEC-555796-JDL-Attributes-v0-8.pdf] 03/05/2006 JRA1-Middleware
2. Glue Information model specification 2.0 [http://www.ogf.org/documents/GFD.147.pdf] March 3,2009 GFD-R-P.147