Computer Science Journal of Moldova, vol.**14**, no.2(41), 2006

The Development Methodology of the UML Electronic Guide

N.A. Magariu, L.L. Nigreţcaia-Croitor, M.V. Croitor

Abstract

A technological model for realization of the electronic guide to UML language is considered. This model includes description of peculiarities of using the special graphic editor for constructing the UML diagrams, XML vocabularies (XMI, DocBook, SVG, XSLT) for representing the text and diagrams and JavaScript code for constructing the tests.

1 About the problem

The high capacities of modern computers and networks of computers on the one hand and the needs of the big number of users on the other hand push developers of information technologies to create new approaches and means for the Complex Programming Systems (CPS) development. One of such means is the UML (Unified Modeling Language). The UML represents a graphic language for specifying various aspects of Object Oriented (OO) Programming Systems (PS) design [1]. It is already the input language for some CASE (Computer Aided Software Engineering) systems [2]. The methodology of applying the UML and methods of its realization are the subjects of modern scientific research. Our research of the tendencies in CPS elaboration methodology using OO CASE systems demonstrated that on the one hand one needs a lot of time to master these systems, and on the other hand, these systems themselves should be more effective [3]. In Software Design Research Laboratory, Faculty of Mathematics and Computer Science, State University of Moldova, an improved architecture of a CASE system named “MD Case” is being elaborated. General functional requirements to “MD Case” system are: - support for the forward and reverse engineering; - support for round-trip engineering which facilitates the obtaining complete program's code in a concrete OO language; - exporting and importing the UML models, represented in XMI language; - support for wide possibilities of using design patterns; - providing the possibility of automated project evaluation; - providing the possibility to master this system very quickly [3]. One of the “MD Case” system components is the “Electronic Guide” subsystem, which must help the user to become familiar with UML language constructions and methodology of their applying. The problem of learning and mastering the UML language is very important, that's why it had been decided to develop the independent electronic guide to UML language in Romanian, Russian and English.

2 Guide's concept

Modern technologies for the development of the electronic guides are usually being reduced to writing the static web-applications, dynamic web-applications or the complex interactive system, allowing carrying out user's mastering and examination concerning all aspects of the discipline [4]. On analyzing these technologies it has been found that the method of electronic guide construction based on representing the informational and programming components in HTML, XML, Java, PHP, etc. is the most commonly used. Such guides could be realized as web pages, which are executed by browser. Static web pages usually are generated automatically from .doc files with the help of MS Word or another editor. For the most part they support only sequential viewing and studying information. Dynamic web pages are interactive. They allow user to enter some information and to receive information depending on entered data. Such applications usually contain a database, an interface for maintaining and a user interface. The following tools can be listed among technological means, necessary for building such type of application: a HTML or XML editor, a data manager like MySQL, a programming system like JDK, servers Apache, PHP, etc. Sometimes special tools for automatic construction of e-Learning systems are used. One of these is the “Combinator” – an instrumental system for guide development [5]. Often the necessary tools for electronic guides’ development and maintenance require additional licensing costs. Such technologies often use texts and multimedia files (pictures, audio, video), which are represented in different formats. This fact implies heterogeneity of data representation forms: text is usually represented in HTML format, graphics – in JPEG or GIF formats, audio – in MIDI or MP3 formats, and video – in AVI or MPEG. The heterogeneity of the data representation forms limits the information search, because it takes place within text format only.

The proposed technological model solves this problem by using the XML language to represent both text and other types of data. Since XML format represents a structured text, which describes the content of the document, but not its representation, the searching doesn’t depend on the type of information (text, graphics, audio information). In addition such method of information representation is very useful for native databases creation.

Having analyzed the contents of the UML guide, it has been found, that this guide should contain following types of information: texts, UML diagrams and tests. Texts can be in Romanian, Russian or English languages. UML diagrams should be imported from CASE systems with UML as input language. After reading and learning any part of the guide, the user can test his knowledge immediately. Tests can be complex enough and can contain UML diagrams.

Taking into account the above-enumerated requirements it had been decided to realize the guide as a dynamic web-application, which uses XML to represent all the information. This had required analyses of above mentioned types of information representation in XML, and an effective method of their combination in a web-application. As a result of the analysis of web-applications creation methods, the method of the electronic guide realization, based on using of various XML vocabularies, had been chosen [6]. These vocabularies are: XMI (XML-based Metadata Interchange), SVG (Scalable Vector Graphics), DocBook, and XSLT (XSL for Transformations) [7,8,9,10]. The JavaScript language had been chosen for tests implementation.

3 The proposed technological model

To represent the formatted text in the guide, it had been decided to use DocBook language, because DocBook documents can be processed in the same way as XML documents, and because of the possibility of their simple transformation into PDF or CHM documents [9, 10]. Representation of the text in DocBook format can be obtained using the AbiWord processor, which reads Word Document (*.doc) files and generates the DocBook (*.dbk) files [11]. Thus, the guide's texts can be prepared by means of the MS Word-like system, and later transformed into DocBook format.

To represent UML diagrams in the guide, it had been decided to use XMI, a language for describing and interchanging metadata and, particularly, UML models. For the UML diagrams' construction it had been decided to use the "MD Case" system's graphic editor, which allowed to build UML models and to receive their descriptions in XMI language. There is an important property of the "MD Case" system that distinguishes it from the most of the similar systems. This property consists in the possibility to store the coordinates and dimensions of UML diagram elements in the generated XMI document [6,12].

The modern web-browsers are not adapted to display UML diagrams represented in XMI language. However, two-dimensional graphics can be rather easily displayed by browsers if they are represented in the SVG language. The problem of transforming the XMI document into SVG-document was solved by using the XSLT technology. For the purpose of viewing SVG graphics the SVG Viewer must be integrated in the Web browser as a plug-in.

For tests realization the JavaScript language is being used. The scripts can be included in XSLT templates or in DocBook document.

The guide content is being developed as an XML document, where the texts are represented in DocBook format, the UML diagrams – in XMI format, and the tests are realized in JavaScript. The browser visualizes the XML document, using the corresponding XSLT transformation and SVG Viewer.

The general logical scheme of our technological model is presented in the Figure 1.

On analyzing this model one can observe that the XMI descriptions of UML diagrams (*.xmi) must be included in the DocBook document (*.dbk). The *.dbk file is processed in accordance to XSLT transformation templates (*.xslt) and using special XSLT processor, integrated in Web browser. These XSLT templates are used to transform DocBook document into HTML document with SVG images descriptions included.

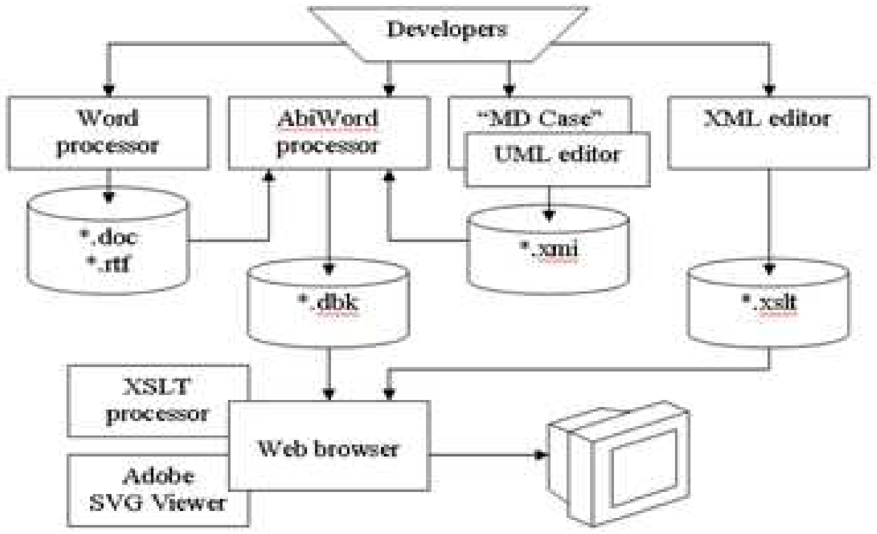

Figure 1. The general scheme of the technological model

4 Instruments application particularities

The common structure of the UML diagram description in XMI format looks in much the same way as the example, which contains XMI description of the UML class diagram:

```
<XMI version = "1.1" xmlns:UML = "http://omg.org/UML">
  <XMI.header>
    <XMI.documentation>
      <XMI.exporter xmi.extender = "MD Case"/>
    </XMI.documentation>
    <XMI.model xmi.name = ""/>
    <XMI.metamodel xmi.name = "UML"/>
  </XMI.header>
  <XMI.content>
    <UML:Namespace.ownedElement>
      <UML:Class name = "NewClass1" xmi.id = "NewClass1"
      isAbstract = "false"/>
    </UML:Namespace.ownedElement>
  </XMI.content>
  <XMI.extensions xmi.extender = "MD Case">
    <UML:Diagrams>
      <UML:Diagram kind = "Class Diagram">
        <UML:Diagram.elements>
          <UML:DiagramElement type = "clasa" name =
          "NewClass1" originx = "74" originy = "73"
          width = "110" height = "110"/>
       </UML:Diagram.elements>
      </UML:Diagram>
    </UML:Diagrams>
  </XMI.extensions>
</XMI>
```

Because the standard XMI description of the UML model doesn't contain the information about representation of the model as a set of diagrams, it had been decided to extend this description by adding special elements: **UML:Diagrams**, **UML:Diagram**, **UML:Diagram.elements**, **UML:DiagramElement**. New elements had been added inside the **XMI.extensions** element that was specially destined for describing the extensions of the XMI format. The attributes of above-mentioned new elements contain all the information needed for visual representation of the UML diagrams. Particularly, they contain coordinates and dimensions of the diagram elements. For example, in above-stated XMI-document the element **UML:DiagramElement** contains attributes: **originx="74", originy="73", width="110", height="110"**. These attributes are being used while transforming the XMI specifications into SVG specifications. The values of these attributes specify the rectangle in the following SVG-document, which represents specification for the graphical notation of the UML class described in the above XMI document:

```
...
<svg:svg width="400" height="400">
   <svg:g>
      <svg:rect x="74" y="73" height="110" width="110"
      style="stroke:rgb(0,0,0);fill:rgb(255,255,255);"/>
      <svg:line style="stroke:rgb(0,0,0); stroke-width:1;
      " x1="74" y1="93" x2="184" y2="93"/>
      <svg:line style="stroke:rgb(0,0,0); stroke-width:1;
      " x1="74" y1="113" x2="184" y2="113"/>
      <svg:text x="94" y="93" style="font-size:12;
      font-weight:12; font-family: Courier;
      font-style:italic">NewClass1</svg:text>
   </svg:g>
</svg:svg>
...
```

The **svg** is the root element in this description. The size of the image is specified with the help of its attributes. The **g** element serves as a container of graphic elements. The **rect** element describes a rectangular. Its attributes set the coordinates of the top left corner of a rectangular, the size of a rectangular, its color and the color of the borders. The **line** element describes a segment. Its attributes specify coordinates of the segment endings, color and thickness of the line. It is necessary to accentuate, that the SVG-document was obtained as the result of the XSLT transformation, applied to the XMI document.

XSLT transformations represent instructions of kind “if node X is found then the Y action is to be executed”. The root element of the XSLT transformation may contain such child elements as **xsl:template** which sets a rule of transformation, **xsl:apply-templates** which is used to apply the rules, **xsl:element** – to create element nodes, etc. To demonstrate the common structure of XSLT template let us consider a template, transforming some parts of the mentioned above XMI document.

The XSLT template that transforms the above-stated XMI class description to SVG format (using **svg** namespace) is represented in APPENDIX 1.

Thus, for all XMI-descriptions of the elements of UML-diagrams we can define the corresponding templates. On applying these templates, the XSLT processor, which is the part of web browser, generates documents with both HTML and SVG elements. Now it is required to display simultaneously both the text and figures. Usually, browsers ignore all tags that are not the standard HTML tags. Adobe SVG Viewer plug-in – a special COM-component used to process SVG elements, could be integrated in web-browser [13]. Adobe SVG Viewer plug-in processes elements of the SVG namespace, therefore the SVG namespace should be declared in the root template. The plug-in can be activated in the following way:

```
...
<html xmlns:svg = "http://www.w3.org/2000/svg">
   <object id = "AdobeSVG" CLASSID = "clsid:78156a80-c6a1-
   4bbf-8e6a-3cd390eeb4e2"/>
   <xsl:processing-instruction name = "import"> namespace =
   "svg" implementation = "#AdobeSVG" </xsl:processing-
   instruction>
...
</html>
...
```

This fragment of code means that all tags, belonging to **svg** namespace will be processed by Adobe SVG Viewer plug-in.

For processing and transforming XML documents the special programs – XML processors are used. Modern browsers come with XML processors, implemented as COM components. The XML processor allows using JavaScript code simultaneously with XSLT. That can essentially increase the capabilities of the guide. For example, it can allow the inclusion in guide some knowledge evaluation system. Besides this, taking into consideration that all the data is stored in a single file, per page representation of the information is required. JavaScript allows achieving this goal.

The proposed model of guide should have a simple knowledge evaluation system, meant for self-control. In such a system the security for user unauthorized data access is not a primary goal because he is interested in an adequate evaluation of himself. Therefore the information about tests could be stored in an XML-oriented database. One of the methods for tests representation in XML documents is the method, corresponding to QTI (Question and Test Interoperability) specification from IMS (Instruction Management System) standard [14]. At the same time the testing process itself can be implemented in JavaScript language.

The guide which is elaborated in conformance to this technological model represents an XML-oriented database. The access and representation of the data from this database, as well as interaction with user is realized by browser according to XSLT templates and JavaScript programs. The content of database could be easily modified without making any modifications in XSLT templates.

Replacing ,,UML editor" by ,,SVG editor" makes this technological model suitable for construction of any other electronic technical literature which contains a lot of graphics.

Implementation of this model doesn't require large financial costs for buying expensive tools and allows the creation of the platform independent applications, which can run both online and offline.

References

[1] OMG's UML Resource Page http://www.uml.org/

[2] OMG's List of UML Tools http://www.uml.org/#Links-Tools

[3] Magariu N. *Complex programs development by means of CASE systems.* Second Conference of the Mathematical Society of the Republic of Moldova: Communications, Chisinau, August, 17-19, 2004. Ch: Tipogr. Acad. Şt. RM – 2004, pp. 207–210.

[4] Carmocanu Gh., Magariu N. *E-Learning applications development.* The scientifical annals of the State University of Moldova. "Physical and Mathematical Sciences" series. Chisinau, CEP USM, 2005, pp.161–170. (Romanian)

[5] The technology of dictionary, guide and manual construction. http://www.elcode.ru/other_tech.html

[6] Magariu N. A. *The technological model for the realization of the electronic guide to UML language.* The proceedings of the 1-st scientific-practical conference "The Modern Problems of the Applied Computer Science", May 23-25, 2005, St.-Petersburg Engineering and Economics University. INJECON, 2005, pp. 73–76. (Russian)

[7] OMG's XMI Specification http://www.omg.org/technology/documents/modeling_spec_catalog.htm#XMI

[8] W3C's SVG Resource Page http://www.w3.org/Graphics/SVG/

[9] DocBook Specifications
http://www.docbook.org/specs/index.html

[10] W3C's XSLT Resource Page http://www.w3.org/TR/xslt20/

[11] AbiWord Web Page http://www.abisource.com/

[12] Buga T., Magariu N. *The development of the ,,MD Case" system's graphic editor.* The scientifical annals of the USM. "Students works" series. Natural sciences. Chişinău, CEP USM, 2005, pp. 61–65. (Romanian)

[13] Adobe SVG Viewer Web Page http://www.adobe.com/svg/viewer/install/

[14] IMS Question & Test Interoperability Specification http://www.imsglobal.org/question/index.html

APPENDIX 1. The XSLT template for XMI class transformation into SVG format

```
... <xsl:template match="UML:Diagram">
   <xsl:if test="@kind='Class Diagram' and ./UML:Diagram.
   elements/UML:DiagramElement/@type='clasa'">
      <xsl:variable name="x" select="./UML:Diagram.
      elements/UML:DiagramElement/@originx"/>
      <xsl:variable name="y" select="./UML:Diagram.
      elements/UML:DiagramElement/@originy"/>
      <xsl:variable name="h" select="./UML:Diagram.
      elements/UML:DiagramElement/@height"/>
      <xsl:variable name="w" select="./UML:Diagram.
      elements/UML:DiagramElement/@width"/>
      <xsl:variable name="name" select="./UML:Diagram.
      elements/UML:DiagramElement/@name"/>
      <svg:rect x="{$x}" y="{$y}" height="{$h}"
      width="{$w}"
      style="stroke:rgb(0,0,0);fill:rgb(255,255,255);"/>
      <svg:line style="stroke:rgb(0,0,0); stroke-width:1;">
         <xsl:attribute name="x1">
            <xsl:value-of select="$x"/>
         </xsl:attribute>
         <xsl:attribute name="y1">
            <xsl:value-of select="$y + 20"/>
         </xsl:attribute>
         <xsl:attribute name="x2">
            <xsl:value-of select="$x + $w"/>
         </xsl:attribute>
         <xsl:attribute name="y2">
            <xsl:value-of select="$y + 20"/>
         </xsl:attribute>
      </svg:line>
      <svg:line style="stroke:rgb(0,0,0); stroke-width:1;">
         <xsl:attribute name="x1">
            <xsl:value-of select="$x"/>
         </xsl:attribute>
         <xsl:attribute name="y1">
            <xsl:value-of select="$y + 40"/>
         </xsl:attribute>
         <xsl:attribute name="x2">
            <xsl:value-of select="$x + $w"/>
         </xsl:attribute>
         <xsl:attribute name="y2">
            <xsl:value-of select="$y + 40"/>
         </xsl:attribute>
      </svg:line>
      <svg:text x="{$x + 20}" y="{$y + 16}" style="font-
      size:12; font-weight:12; font-family: Courier;
      font-style:italic">
         <xsl:value-of select="$name"/>
      </svg:text>
   </xsl:if>
   <xsl:apply-templates/>
</xsl:template>
...
```

N.A.Magariu, L.L.Nigreţcaia-Croitor, M.V.Croitor

Received August 23, 2006

State University of Moldova
60, A. Mateevici str.
Chişinău, MD−2009, Moldova
Phone: (373+2) 57-77-33
E–mail: *magariu@usm.md*, *liudmile@softhome.net*, *krey@usm.md*