ACTA DIDACTICA NAPOCENSIA

Volume 6, Number 3, 2013

NASSI-SCHNEIDERMAN DIAGRAM IN HTML BASED ON AML

László Menyhárt

Abstract: In an earlier work I defined an extension of XML called Algorithm Markup Language (AML) for easy and understandable coding in an IDE which supports XML editing (e.g. NetBeans). The AML extension contains annotations and native language (English or Hungarian) tag names used when coding our algorithm. This paper presents a drawing tool with which teachers and students can define their algorithms in visualized Nassi-Schneiderman diagram (NSD). The algorithm is stored in AML format. Users are not supposed to know the XML.

First I demonstrate the visualization from AML to Nassi-Schneiderman diagram using XSL transformations. The diagrams are in HTML format because it is easy to generate from an XML file and most browser support its presentation with special JavaScripts. Second I provide a larger toolset for obtaining AML code from the visualized NSD. HTML and JavaScript together are also a good choice for this because they support the scripting with which the algorithm is modified.

I found that HTML is fine for AML visualization. Algorithms can be modified from the graphical appearance with JavaScript functions. I developed an application which is available on my website.

This visualization tool is useful to users who are not familiar with XML language but would like to present their algorithms in a graphical mode. The paper is especially more practical to those teachers who would like to present the algorithmic way of thinking to their students and those students who would like to practice it.

Keywords: Algorithm, Nassi-Schneiderman Diagram, HTML, XML, XSLT, JavaScript

1. Introduction

Nassi-Schneiderman diagrams (NSD) are excellent for visualizing a procedure without learning a language [8], [6], [7]. There are some tools for visualizing algorithms in NSD, for example Structorizer [1] is one of them. There are some new tools in a Bsc thesis and a student's research work (a report of scientific think tank), see [2], [3]. In this paper I present another visualization tool with which algorithms can be drawn platform independently on every computer screen which has a browser. I am interested in platform independent web solutions. Since my research and teaching activities are related to XML, I have chosen the server-client platform for developing a tool. The algorithm is stored on server side in Algorithm Markup Language (AML) format. AML was defined in one of my previous papers [10], and it is based on XML. Tag names are defined in the XML Schema Definition. I have given two schemas to define AML. One of them is in English; the other one is Hungarian. I use the Hungarian version at teaching; therefore, students can follow the courses and learn more easily. Possible node names and their annotations are published at coding time in an IDE which supports XML editing (e.g. NetBeans).

In this paper an algorithm will be visualized in NSD. The visualized NSD is described in HTML with `div`-s and `table`-s. My web application uses XSL transformation on the server side, but it has many clever solutions on the client side too.

Although the AML is a modified type of XML, from this file users can generate source code; however, this paper is not about code generation. I am only concerned with the visualization and modification of algorithms.

I also present a new toolkit for editing algorithms without knowing their storing format, the AML. Users can manage their editing commands from the visualized layer by clicking their mouse.

Received 4 September 2013.

A visualized algorithm can be understood easily, and if it is editable, it helps the learning process as well.

My next paper will be about a more complete web application which assists in problem solving, with which users will be able to edit their algorithms via browsers and with which they can generate pre-processed documentation.

2. Visualization

In this paper I present the Nassi-Schneiderman diagram, which visualizes algorithms using special boxes and free texts in the boxes. The reason I have slightly modified the previously defined AML is that every node should contain a simple text instead of more precisely defined expressions. For example an operation can simply be a free text instead of a variable (on the left side) and an expression (on the right side).

Figure 1 gives an example for the AML. It is an XML file, which defines an algorithm. This algorithm counts the number of items with a T property in an array.

```
<?xml version="1.0" encoding="UTF-8"?>
<procedure        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"        name="Counting"
xsi:noNamespaceSchemaLocation="aml_en.xsd">
       <sequence>
              <operation id="1">
                     <text>count:=0;</text>
              </operation>
              <loopCheckBefore id="2">
                     <condition>
                            <text>i:=1..N</text>
                     </condition>
                     <sequence>
                            <if id="4">
                                   <condition>
                                          <text>T(X[i])</text>
                                   </condition>
                                   <then>
                                          <operation id="5">
                                                 <text>count:=count+1;</text>
                                          </operation>
                                   </then>
                                   <else>
                                          <operation id="6">
                                                 <text>-</text>
                                          </operation>
                                   </else>
                            </if>
                     </sequence>
              </loopCheckBefore>
       </sequence>
</procedure>
```

Figure 1. *Counting algorithm in AML*

This XML file is not as easy to understand as a source code for any programing language. Users prefer simple, logically arranged pictures, for example the Nassi-Schneiderman diagrams. Figure 2 presents the NSD of the previous algorithm.

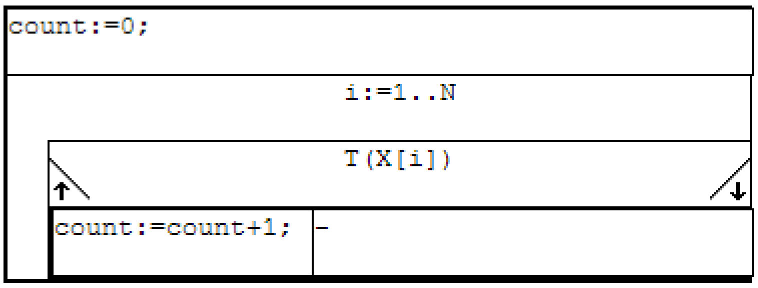
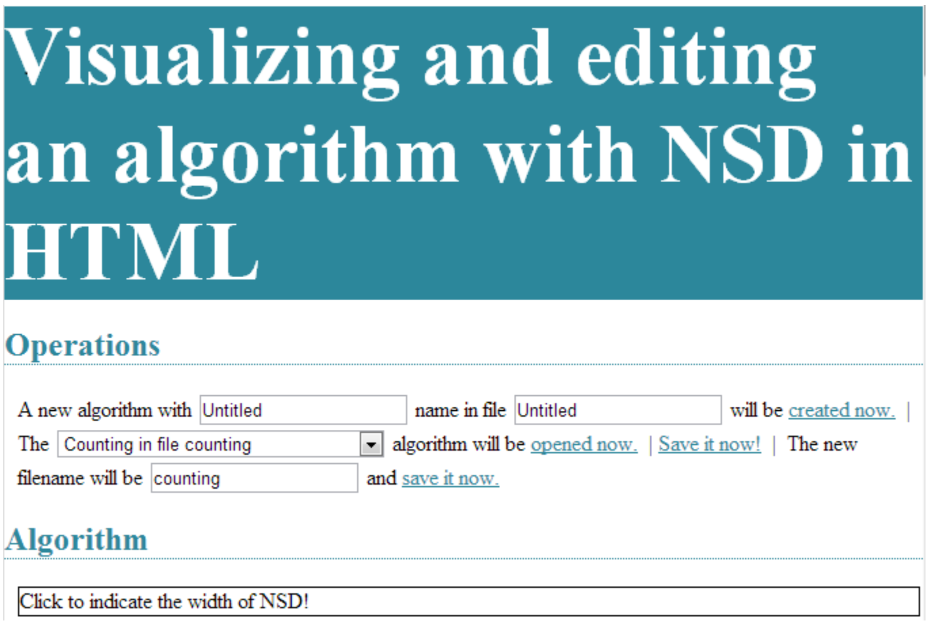


Figure 2. *Counting algorithm in NSD format*

This diagram is drawn as a picture as in the Structorizer [1]. I chose the HTML presentation as in [2]. I developed a server-client web application as a tool. The web application is published on a Glassfish Application Server because I developed the tool in Java. The AML files are stored on the server side. The visualization of AML starts on the server with an XSL Transformation. When a user chooses an algorithm, its source XML is loaded from the file system to the memory and it will be transformed to HTML format with a self-implemented XSLT. The diagram appears from this HTML in the browser on the client side.

The complete web application has a header as shown in Figure 3.


Visualizing and editing an algorithm with NSD in HTML

Operations

A new algorithm with Untitled name in file Untitled will be created now. | The Counting in file counting algorithm will be opened now. | Save it now! | The new filename will be counting and save it now.

Algorithm

Click to indicate the width of NSD!

Figure 3. *Header of the tool*

The web application works with AJAX, so the header is constant. Only a `div` will be replaced when the user opens another algorithm.

Once displayed, the bordered box can be used to set the width of the diagram. It affects only the client side, the browser.

3. Editing an algorithm

Once an algorithm is ready users can draw it. However a picture is difficult to edit, therefore, rather than editing, they should start the drawing again. However this is not how people think – they usually notice a mistake during design. The design can be helped by a visualized algorithm, which is why the ability to edit is a requirement.

This tool is good for editing an algorithm from the browser. Every action starts on client side, but the modification reaches the server and finally the result will be downloaded to the client. First I present how users can create a new diagram, "save" or "save as". Second I demonstrate the process behind each of the steps. Third I show the possibilities on the client side. Fourth I describe the server side activities.

3.1. Operations

On the top of this web site the following operations are available. Users are able to run four commands by clicking on the links.

A new algorithm can be created in a new file with its own name. The initial algorithm contains only an operation with "-" (minus) character. For example the command to create a new algorithm with name "Test" in a `test.aml` file looks as in Figure 4.

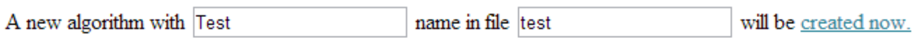

A new algorithm with Test name in file test will be created now.

Figure 4. *Creating Test algorithm in test.aml*

After clicking on link "created now" a new algorithm is drawn on the screen. Meantime a new `test.aml` file is created on the server. Figure 5 depicts the content of the new file. Figure 6 shows the diagram of the new algorithm.

```
<?xml version="1.0" encoding="UTF-8"?>
<procedure          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"          name="Test"
xsi:noNamespaceSchemaLocation="aml_en.xsd">
       <sequence>
              <operation id="1">
                     <text>-</text>
              </operation>
       </sequence>
</procedure>
```

Figure 5. *Source of initial algorithm*

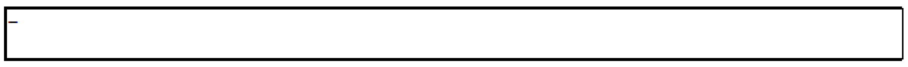


Figure 6. *Diagram of initial algorithm*

A previously created algorithm can be opened by clicking on link "opened now". Users can choose an existing algorithm from the drop-down list. The list contains names of algorithms with their container file's name.

The loaded algorithm can be edited in the server's memory. The modified algorithm is saved on the file system when link "Save it now!" is used.

If the user would like to save the algorithm into a new file, a new name should be given in the latest textbox and link "save it now." must be clicked.

3.2. Opportunities on client side

The users see a diagram on the screen which is a state of an algorithm. There are some editing features with which the algorithm can be changed. When the user right-clicks on one of the NSD boxes, a floating menu appears near the cursor. This menu contains the possible events. Figure 7 shows the menu items.

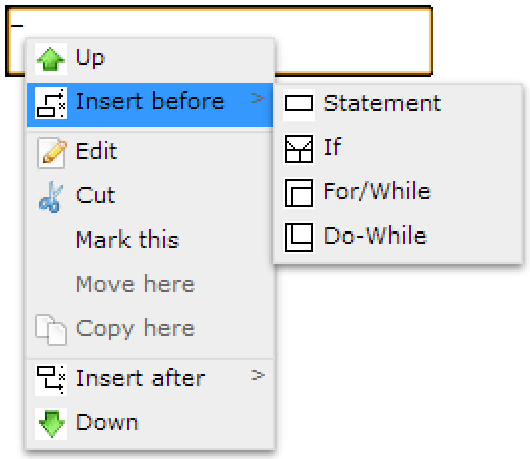


Figure 7. *Floating menu*

Table 1 lists the items and its impact.

Table 1. *Menu items*

Item	Impact
Up	Selected item and the previous one change their position
Insert before	Insert a new item before the selected one
Edit	Selected box value will be editable in a textarea. After editing user must click on button OK or Cancel.
Cut	Selected item will be deleted
Mark this	Mark the selected item
Move here	Marked item will be moved before the selected one
Copy here	Marked item will be copied before the selected one
Insert after	Insert a new item after the selected one
Down	Selected item and the next one change their position

The user must choose the type of element before inserting a new item. The following types are available:

- **statement** with free text (assignment, function call, etc.),
- **if** as two-way decision,
- **for loop** or **while loop** because these two loops have the same pattern in the diagram,
- **do-while loop**.

3.3. Process of editing an algorithm

See Figure 8. to understand the process of editing an algorithm. At every action the client sends a request to the server with parameters. When the user would like to edit a text then first a `textarea` appears with the original value. After editing button `OK` saves the new value and loads the NSD while button Cancel loads back the original NSD only. Finally the client draws the diagram in the browser which shows the algorithm's new state.

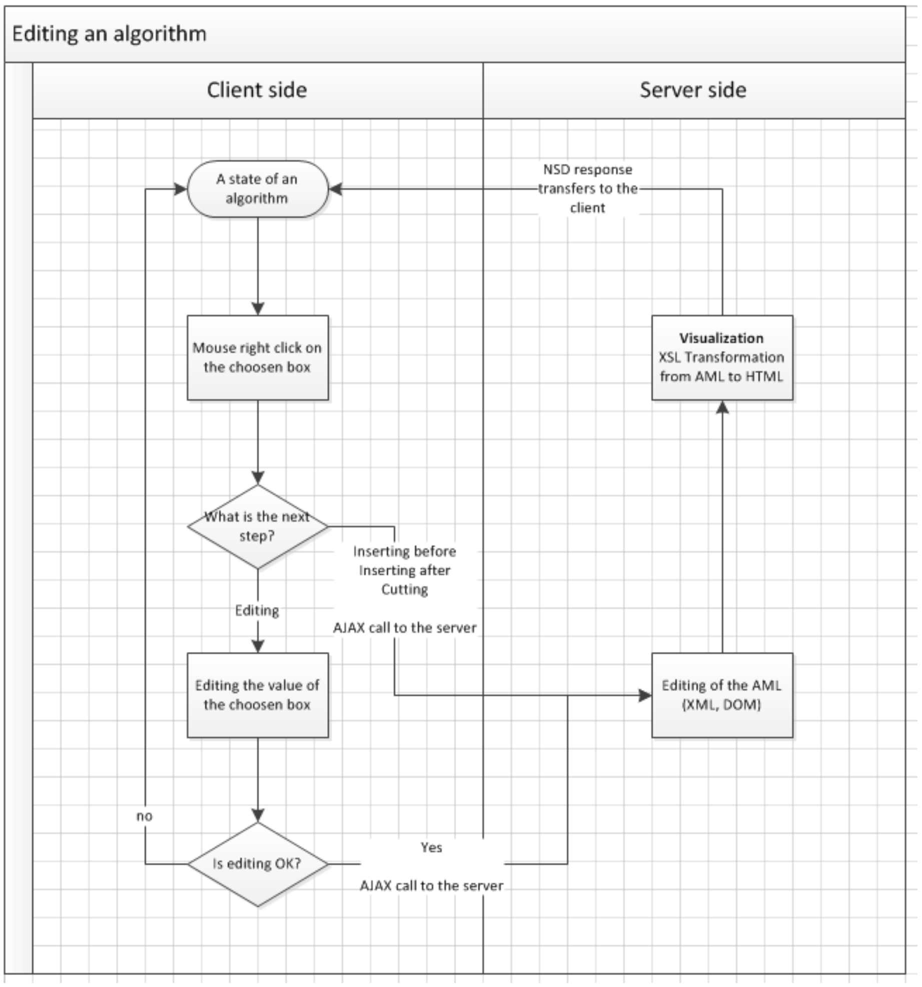


Figure 8. *Process of editing an algorithm*

3.4. Events on the server side

Meanwhile, the page is not reloaded, but the server is reached with AJAX calls. Every AML file is stored on the server side. The actual algorithm is loaded into the memory and it is saved temporarily in the session. This special XML file is handled with DOM function library so its modification is very easy and fast. The modified AML is transformed to HTML again and this part of the website is downloaded to the client. Every important node has an attribute which stores a unique identifier. This id is present in the HTML source as well. The context menu handler has access to this attribute of the selected item, on which the mouse right-click happened. The AJAX call from the Javascript contains this information so the server knows exactly the identifiers of the selected item and the action.

4. Related works

There are some tools drawing an algorithm which are similar to Nassi-Schneiderman diagram. Some of these allow users to draw a complete algorithm. Others are good for editing them as well. Paint or any other graphics software may be used as well as the tables in Microsoft Word or any other office application. Microsoft Visio does the same with a simple template [4]. SmartDraw [5] or Structorizer

[1] applications are good for editing an algorithm. So the middle of the algorithm can be modified at any time and the diagram will be repainted as many times as it is needed. To use these applications users must at least download and install a new application. This year, in parallel with but unknown, a Bsc thesis and a student research work were elaborated on this.

5. Conclusion

My tool is a platform independent solution. Users can visualize their algorithms and they can edit them in most browsers with this implemented client-server tool. They will be able to take a screenshot and use the NSD in their documents. So it is a good choice for students who would like to generate an NSD in an easy way. It is also a useful tool for teachers who would like to use a simple tool to visualize algorithms.

6. Future work

After this paper I continue my research by improving the website [9] with new features. I am implementing usermanagement, handling rights and helping the design method. I am working on a Wizard with which users will be helped in the over-thinking method of a programing task. Teachers are getting an excellent tool to teach algorithmic thinking, coding and design.

References

[1] Structorizer, http://structorizer.fisch.lu/

[2] A Bsc's tesis in ELTE from motiabt.elte, http://stukimania.hu/

[3] Németh Balázs's student research work, DAVIK, (Dinamikus Algoritmus Vizualizációs Keretrendszer) Dinamic Algorithm VIsualisation Framework

[4] A Visio Template, http://users.tm.net/tonietienne/

[5] SmartDraw, http://www.smartdraw.com/resources/tutorials/nassi-shneiderman-diagrams/

[6] Weiss, E. H., "Visualizing a Procedure with Nassi-Schneiderman Charts", Journal of Technical Writing and Communication, Vol.20, No.3, 1990, pp. 237-254

[7] T. Dean Hendrix, James H. Cross , "Language Indepdendent Generation of Graphical Rep-resentations of Source Code", Proceeding CSC '95 Proceedings of the 1995 ACM 23rd an-nual conference on Computer science Pages 66-72, 1995.

[8] B. Schneiderman, R. Mayer, D. McKay, P. Heller, "Experimental investigations of the utility of detailed flowcharts in programming" Communications of the ACM, no. 20, pp. 373-381, 1977.

[9] Programming Fundamentals Wizard, http://xml.inf.elte.hu:8080/PFW

[10] L. Menyhárt, "Can a language be before the "First programming language? "", Teaching Mathematics and Computer Science, Vol.,No., pp. 209-224, 2011

Authors

László Menyhárt, Eötvös Loránt University, Budapest (Hungary), e-mail: menyhart@inf.elte.hu