TECHNICAL UNIVERSITY OF CLUJ-NAPOCA

ACTA TECHNICA NAPOCENSIS

Series: Applied Mathematics, Mechanics, and Engineering
Vol. 58, Issue II, June, 2015

PROGRAMMING CANVAS X Pro 16 USING SCRIPTING TECHNOLOGIES

Tiberiu Alexandru ANTAL

Abstract: *The paper gives a short overview on the Canvas X Pro 16 integrated environment for vector illustration, imaging, presentations, and Web publishing software together with the scripting technologies that can be used to program the product. Some programming examples in VBScript and Visual Basic about how to draw an extruded spur gear are covering most of the features that are useful for the mechanical engineers work.*
Key words: *Canvas X Pro 16, programming, VBScript, Visual Basic, vector graphics.*

1. INTRODUCTION

Canvas X Pro 16 belongs to the category of vector oriented processing software that creates technical illustrators for many industries as it offers a very flexible, scalable and integrated design environment. Some of the names belonging to the same category of software are: Corel Draw, Inkscape, Adobe Illustrator, however Canvas X has the advantage of being very simple and effective. The software covers at a state-of-the-art level the 2D technical illustration, imaging, presentations, and Web publishing domains and integrates simple and known scripting technologies for programming. It also integrates some simple 3D enhancing procedures (parallel, circular and sweep extrude) based on 2D drawings. The advantages of programming CAD products and some of the available technologies to do that are given in [1] – [4]. Canvas X is not really a CAD software but it does import DWG and DXF files. So most of the CAD work can be done in AutoCAD and then imported to Canvas X. At 2D modeling level Canvas X will do most of the things AutoCAD does.

2. SCRIPTING WITH CANVAS X

The manual [5] gives the technologies that can be used to program Canvas X, while the [6] scripting reference gives a description based on short examples of how to create scripts to automate Canvas X using AppleScript and Visual Basic. Under the Windows operating system scripts can be written in JavaScript and VBScript and executed if Windows Scripting Host is installed. As this is installed by default on most versions of Windows running such a script is done under the command windows by the following syntax:

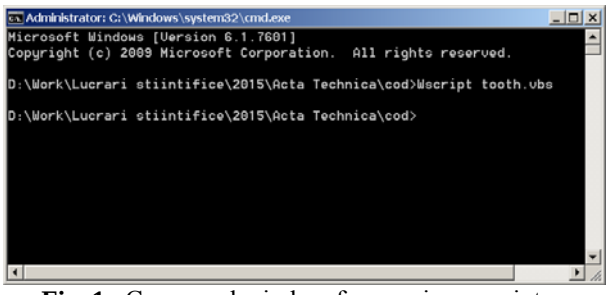

Fig. 1. Command window for running a script.

In Fig. 1 Wscript is the name of the Windows Host executable file (Wscript.exe), followed by a space, then the name of the script file to be run (tooth.vbs). Another way to run a script is directly inside Canvas X by selecting the script from the *File* > *Automation*. The script is loaded using *File* > *Automation* > *Add scrip* and after it's loaded the name of the script will appear in the Automation menu. If we click on the name it will open a new Canvas X

instance and run the script automatically, leaving also the current instance opened.

```
Const cvsIllustrationDocument = 1
' Declare variables.
Dim cv
Dim doc
Dim aly

Set cv =
CreateObject("Canvas.Application")
cv.Visible = true
Set doc =
cv.Documents.Add(cvsIllustrationDocume
nt)
Set aly =
cv.ActiveDocument.ActivePage.ActiveLay
er
'line 1
aly.CreateLine 100, 170, 90, 120
'line 2
aly.CreateLine 100, 170, 110, 120

Set arc1 = aly.CreateArc(90, 102, 4,
18)
arc1.StartAngle = 270
arc1.SpanAngle = 90

Set arc2 = aly.CreateArc(106, 102, 4,
18)
arc2.StartAngle = 90
arc2.SpanAngle = -90

'line 3
aly.CreateLine 94, 102, 106, 102
stop
```

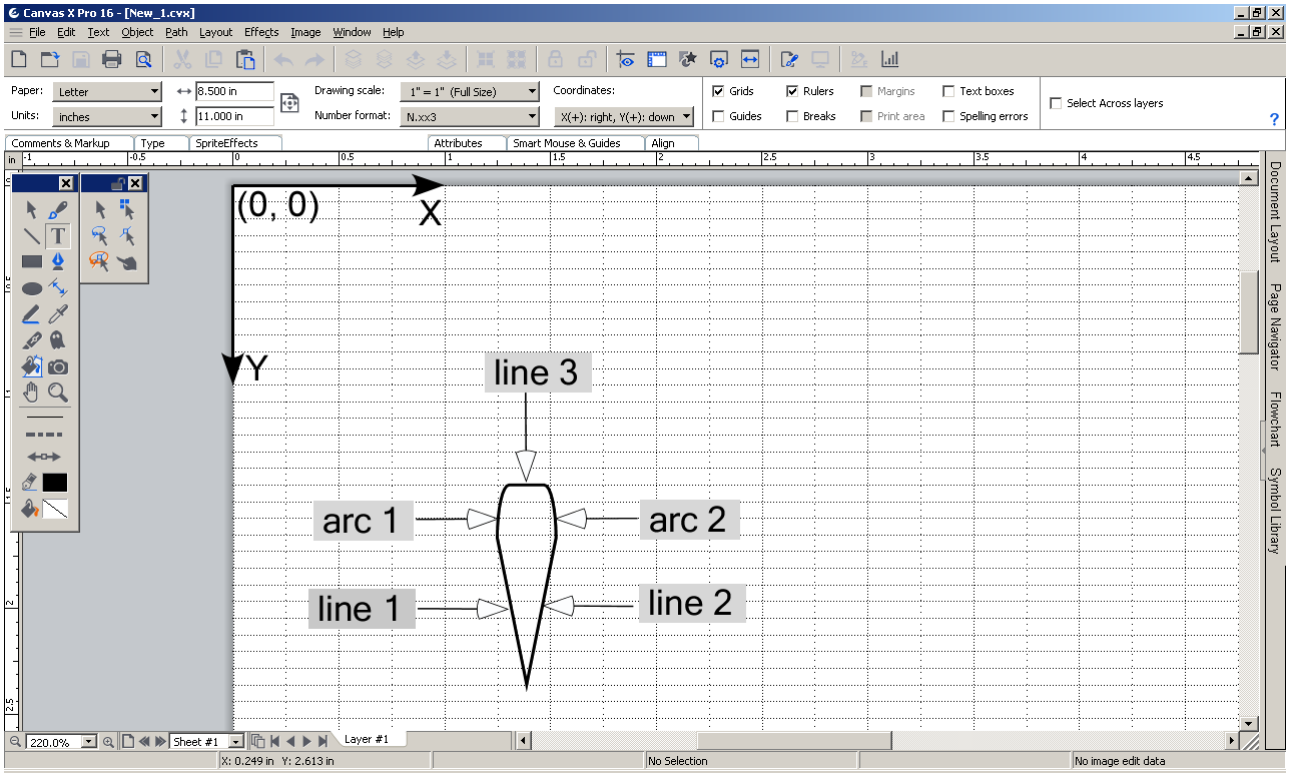

Fig. 2. The tooth.vbs script effect on Canvas X Pro 16.

When running this simple VBScript code Canvas X has a strange habit. Instead of keeping the current environment configuration (which is set from *File* > *Configuration Center*) it opens the original one, where the *paper size* is *Letter* and the *units* are set to *inches*. Unfortunately, the object model has no properties that can be used to set these values when creating a new document. Another bug seems to be related to the *CreateArc*() method of the *CVArc* object. When creating an arc the width, height, starting point in degrees, and the angular length of the arc must be given according to [6]. The code to create the first arc (arc 1) from Figure 1 should normally be the following:

```
aly.CreateArc 90, 102, 4, 18, 270, 90
```

or

```
Set arc1 = aly.CreateArc(90, 102, 4,
18, 270, 90)
```

The code however will not create the expected arc. In Figure 3 we have a smaller arc (in orange) and the expected one (in black). The expected arc is created only is we used the `StartAngle` and `SpanAngle` properties, where the angle values are given in degrees as in the following code:

```
Set arc1 = aly.CreateArc(90, 102, 4,
18)

arc1.StartAngle = 270
arc1.SpanAngle = 90
```

In the first case the angles must be given in radians and not in degrees as stated in [5] and [6] this is why we obtain a bad arc with 5156.620^0 *Delta Angle* and 339.860^0 *Start Angle*.

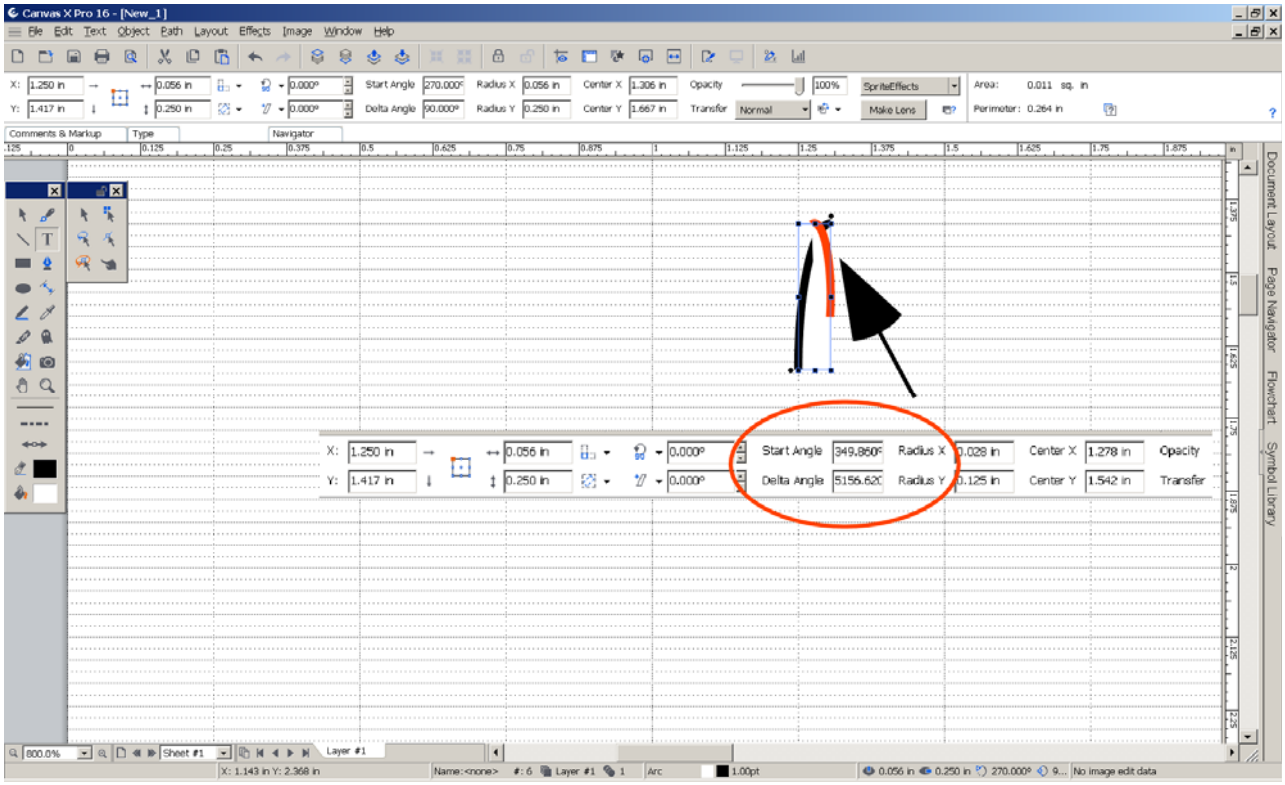

Fig. 3. The *Start* and *Delta Angle* bug with *CreateArc*() method of the *CVArc* object in VBScript.

3. PROGRAMMING Canvas X FROM Visual Basic

To create a Visual Basic script we need to install the language that is contained in the Microsoft Visual Studio package. In the following code I've used Visual Basic 6.0. It is also possible to write the same code from any application (Word, Excel) that contains the VBA editor and language integrated. Access to the Canvas X object model is handled using type libraries. Adding the Canvas X type library is done from *Project* > *Refereces* in the Visual Basic IDE (see Fig. 4.). The *Canvas.tlb* file is stored in the *C:\Program Files\ACD Systems\Canvas X Pro 16* directory.

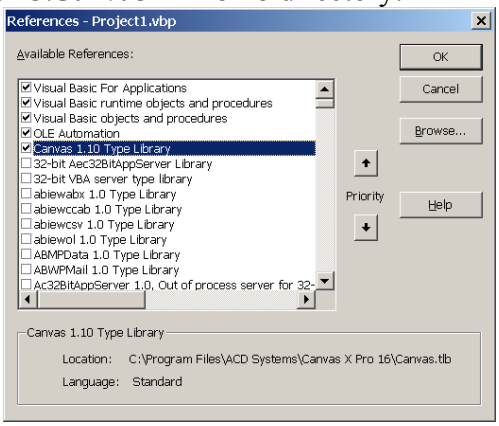

Fig. 4. Adding the Canvas X type library to Visual Basic.

The following code creates an image similar to the one from Fig. 5. The tooth profile is an arc, if necessary it can be replaced by an involute profile if the proper equations are programmed for this.

```
  Dim cv As Canvas.Application
  Dim doc As Canvas.Document
  Dim aly As Canvas.Layer
  Dim rect As Canvas.CVRectangle
  Dim arc1 As Canvas.CVArc
  Dim arc2 As Canvas.CVArc
  Dim line As Canvas.CVLine
  Dim grp As Canvas.CVGroup
  Dim obj As Canvas.DrawObject
  Dim ext As Canvas.CVExtrude
  Dim sel As Canvas.Selection

  'throw an error if the Canvas application can't be started
  On Error Resume Next
  Set cv = GetObject(, "Canvas.Application")
   If Err Then
    Err.Clear
    Set cv = CreateObject("Canvas.Application")
    If Err Then
     MsgBox "Can not conect to Canvas"
     Exit Sub
    End If
   End If

  'clear errors and make the application visible
  On Error GoTo 0
  cv.Visible = True

  'opend a new illustration Canvas document
  Set doc = cv.Documents.Add(cvsIllustrationDocument)

  'get the active layer to start the model
  Set aly = cv.ActiveDocument.ActivePage.ActiveLayer

  'create line 1
  aly.CreateLine 100, 170, 90, 120

  'create line 2
  aly.CreateLine 100, 170, 110, 120
  Set arc1 = aly.CreateArc(90, 102, 4, 18)

  'create arc 1 and arc 2 as in the VBscript example
  arc1.StartAngle = 270
  arc1.SpanAngle = 90
  aly.CreateArc 90, 102, 4, 18, 270, 90
  Set arc2 = aly.CreateArc(106, 102, 4, 18)
  arc2.StartAngle = 90
  arc2.SpanAngle = -90

  'create line 3
  aly.CreateLine 94, 102, 106, 102

  'select the 3 lines and 2 arcs in order to create a single object based on their geometry
  doc.Selection.SelectAll

  'group the selected objects in a single object
  Set grp = doc.Selection.MakeGroup

  'the resulting object is single tooth of the gear, this has to be multiplied to form the full gear
  Set obj = grp.DrawObject

  'replicate the tooth to obtain all the teeth
  obj.Replicate 8, 45, cvsBottomCenterCorner, 0, 0, 0, 0

  'draw a circle over the teeth
  aly.CreateOval 47.5, 117.5, 105, 105

  'select the teeth and the circle to combine the to a gear, at this stage we have a solid gear with no hole
  doc.Selection.SelectAll
  doc.Selection.Combine cvsOutlineCombine

  'create another circle to make a hole in the gear
  aly.CreateOval 80, 150, 40, 40

  'create a new gear by subtracting the circle from the gear
  doc.Selection.SelectAll
  doc.Selection.Combine cvsSubstituteFrontCombine

  'select the gear with hole from the document
  Set sel = doc.Selection
  sel.SelectAll
  Set obj = sel.Item(1)

  'extrude the selection to obtain a solid gear
  Set ext = obj.ConvertToExtrude(cvsParallelStyle)
```

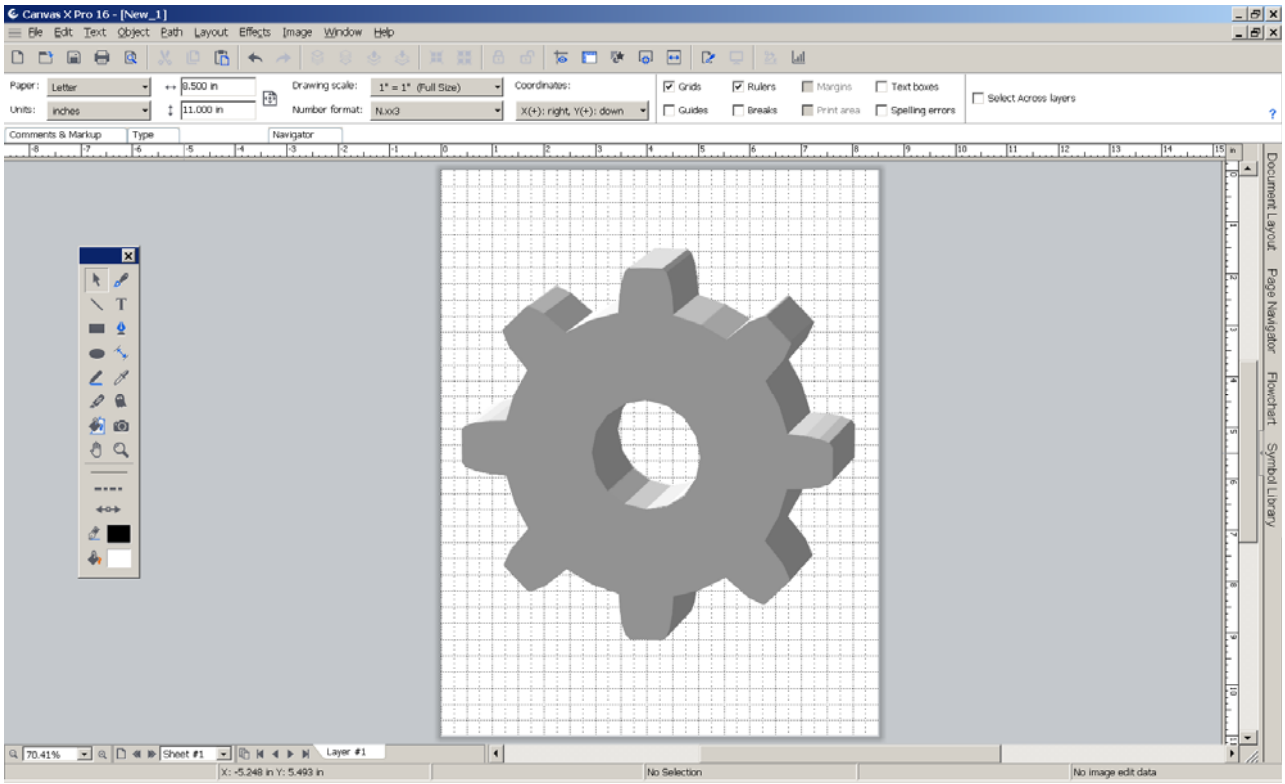

Fig. 5. Spur gear created in Canvas X from Visual Basic.

The following example shows how to create a group of lines as a single object based on a set of equations. The *arcCRIS()* subroutine creates an arc with the following parameters: *r* radius, *a0* start angle, *a1* end angle on the *aly* layer in the *doc* document.

```
  Sub arcCRIS(ByVal r As Double, ByVal
a0 As Double, ByVal a1 As Double,
ByRef aly As Canvas.Layer, ByRef doc
As Canvas.Document)
    Dim n As Double
    Dim pas As Double
    Dim grp As Canvas.CVGroup
    Dim l As Canvas.CVLine

  'steps when creating the arc
  ' from linies
    st = (a1 - a0) / 10#

  'coordinate of the start point
    xa = r * Sin(a0)
    ya = r * Cos(a0)

  'unselect any previous selections
    doc.Selection.DeselectAll

  'draw a set of lines with a
  ' line defined by the (xa,ya)
  ' (xa1,ya1) points, with the st step
  ' from a0 to a1 angles
    For i = a0 + st To a1 Step pas
      xa1 = r * Sin(i)
      ya1 = r * Cos(i)

  'create the line and store a
  ' reference in the l variable
      Set  l  =  aly.CreateLine(xa,  ya,
xa1, ya1)

  ' add the line to the new selection
      doc.Selection.Add l.DrawObject

      xa = xa1
      ya = ya1

    Next i

  'make a group based on the
  ' last selection
    Set grp = doc.Selection.MakeGroup
  End Sub
```

4. TOOLS FOR CREATING VECTOR PATHS BY EQUATIONS

From the *Path* menu, based on the *Math expression 2-D plot* menu item, the user can create vector paths based on equations (Cartesian or polar) as seen in Figure 6. The

obtained vector path can be used in the document as a normal one, created by hand.

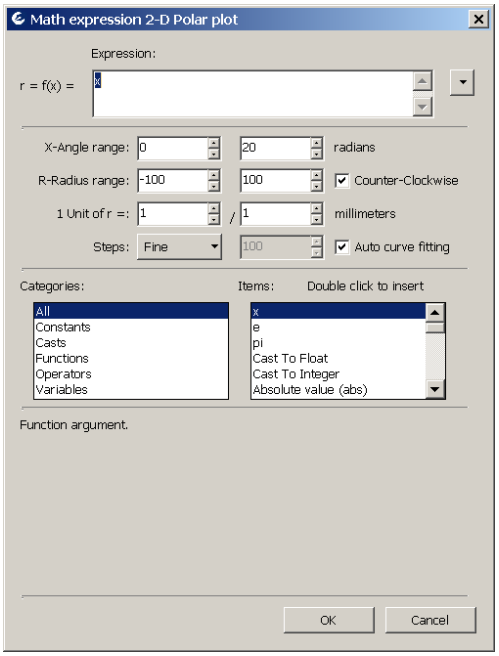

Fig. 6. Creating vector path in Canvas X based on equations.

In Figure 7 such a vector object created with the current pen is filled with ink. Then, the *Bind Text* from the *Effects* menu is used to adjust the vertical orientation of each character to match the path.

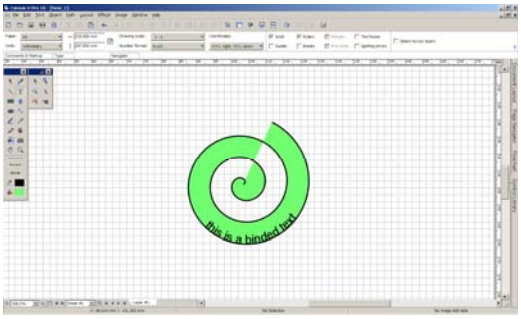

Fig. 7. Using vector path in Canvas X.

5. REFERENCES

[1] ANTAL, T. A., *Mechanism displacement analysis with AutoLisp in AutoCAD*, Acta Technica Napocensis, Series: Applied Mathematics and Mechanics, Nr. 45, p.19-24, U. T. PRESS, ISSN 1221-5872, 2002.

[2] ANTAL, T. A., *Surface generation in Catia v5r11 defined by parametrical equations in visual basic using Activex automation*, Acta Technica Napocensis, Series: Applied Mathemathics and Mechanics, Nr. 46, Vol. 2, p.7-14, ISSN 1221-5872, 2003.

[3] ANTAL, T. A., *Solid gear generation in autocad using Activex Automation and Visual Basic*, Acta Technica Napocensis, Series: Applied Mathemathics and Mechanics, Nr. 47, Vol. III, p.57-62, ISSN 1221-5872, 2004.

[4] ANTAL, T. A., *Visual BASIC pentru ingineri*, RISOPRINT, Cluj-Napoca, p. 244, ISBN: 973-656-514-9, 2003.

[5] ***, *Canvas X Users Guide*, p. 588, Acdsee, 2015.

[6] ***, Canvas *Scripting Reference Guide – Version 6*, p.458, Acdsee, 2015.

Programarea lui Canvas X Pro 16 folosind tehnologii de scripting

Rezumat: Lucrarea oferă o scurtă privire de ansamblu asupra produsului software Canvas X Pro 16 pentru ilustrare vectorială, imagistică, prezentări şi editare Web, împreună cu tehnologiile de scripting care pot fi folosite pentru a programa mediul. Câteva exemple de programare în limbajele VBScript şi Visual Basic cu privire la modul de generare a unei roți dințate cilindrice cu dinți drepți sunt date pentru a ilustra posibilitățile de utilizare în domeniul ingineriei mecanice.

ANTAL Tiberiu Alexandru, Professor, dr. eng., Technical University of Cluj-Napoca, Department of Mechanical System Engineering, antaljr@bavaria.utcluj.ro, 0264-401667, B-dul Muncii, Nr. 103-105, Cluj-Napoca, ROMANIA.