15. ActiveX Automation

Termenul de ActiveX Automation grupează o mulţime de tehnologii software care asigură interacţiunea între componente software, în prezenţa unei reţele, indiferent de limbajul de programare în care au fost create componentele. Termenul de Automation descrie modul de interacţiune între componente. Aceasta se poate manifesta printr-un simplu schimb de date sau prin comenzi date de către o componentă alteia. ActiveX se bazează pe modelul de obiecte de tip componentă (COM de la Component Object Model) ce asigură o arhitectură cvasi-independentă de platformă (maşina şi sistem de operare) pentru aplicaţiile client/server. Clientul ActiveX este o aplicaţie ce apelează un obiect ActiveX. Denumirea de obiect ActiveX se foloseşte pentru un obiect expus ce respectă arhitectura COM. Serverul ActiveX este o aplicaţie care expune un model de obiecte prin care se lasă programat. Un model de obiecte (object model) sau clasă de obiecte este o structură statică formată din toate obiectele şi relaţiile dintre acestea ce se pot folosi pentru programarea unei aplicaţii. În VB, pentru a putea lucra eficient cu obiectele unei alte aplicaţii trebuie setată o referinţă la modelul de obiecte a aplicaţiei de programat. În acest fel se va beneficia de o facilitate numită legare în avans (**early binding** în engleză) care permite declararea unor variabile de tipul unor obiecte specifice aplicaţiei de programat.

15.1 Clase de obiecte

Pe măsură ce se instalează aplicaţii şi controale ActiveX la nivelul registrului Windows se creează noi intrări în care se marchează componentele din clasa celor ce pot fi controlate. Deoarece o aplicaţie poate expune mai multe clase de obiecte în vederea programării prin clienţii de Automation este necesară cunoaşterea numelui aplicaţiei şi a tipului de obiect pentru a controla aplicaţia server din cea client. Această informaţie este încapsulată la nivelul identificatorului clasei sub forma *Server.Clasa*.

15.2 Etapele lucrului cu obiecte ActiveX

Obiectele standard din VB sunt manipulate direct de VB, însă pentru ca cele externe să poată fi folosite în VB trebuie parcurse următoarele etape:

! se creează o variabilă de tipul obiect pe baza obiectelor expuse în modelul de date al aplicaţie server;

! metodele şi proprietăţile obiectului vor fi folosite în codul VB în funcţie de acţiunea pe care dorim să o face la nivelul aplicaţiei server;

! se eliberează memoria alocată pentru variabila obiect creată în VB.

Diferenţa între obiectele VB standard şi cele externe este:

! în cazul obiectelor externe se poate să nu se creeze, în vederea manipulării, toate obiectele afişate în modelul de obiecte;

! aplicaţia externă, pentru a putea fi terminată, va trebui închisă cu o secvenţă de metode specifică ei (terminarea aplicaţiei VB nu duce neapărat la terminarea automată şi a aplicaţiei server).

15.3 Crearea unei variabile obiect ActiveX Automation

Toate aplicaţiile VB ce programează aplicaţii server trebuie la un moment dat să creeze o instanţă a obiectului server. Crearea instanţei are ca efect stabilirea unui canal de comunicaţie cu aplicaţia server pentru a-i comunica obiectele ce urmează a fi controlate din exterior. Rezultatul creării cu succes a instanţei este o referinţă la obiectul server stocată într-o variabilă. Cu ajutorul acestei variabile obiect se va putea controla aplicaţia server folosind metodele şi proprietăţile expuse de ea.

15.4 Legarea în avans şi legarea târzie

Există două modalităţi pentru crearea unei instanţe de obiecte ActiveX Automation: legarea în avans şi legarea târzie, fiecare având avantajele şi dezavantajele ei.

Legarea în avans

La legarea în avans (**early binding**) se adaugă o referinţă la biblioteca de tipuri a componentei în timpul proiectării aplicaţiei pentru a informa VB cu privire la serverul de Automation folosit şi la obiectele lui. Denumirea de legare în avans vine de la faptul că VB poate citi definiţiile de interfaţă şi va cunoaşte care sunt obiectele, împreună cu metodele şi proprietăţile lor, pe care le suportă aplicaţia server înainte de execuţia codului (când se creează canalul între client şi server). În acest caz VB sprijină scrierea codului prin **IntelliSense**, verificarea sintactică în momentul compilării şi folosirea navigatorului de obiecte pentru inspectarea bibliotecii de obiecte. Cunoaşterea modelului de obiecte mai permite ca VB să scurteze comunicaţia între client şi server efectul fiind cel de creştere a vitezei aplicaţiei. Dacă este posibil, această metode este de preferat în locul celei care urmează. Secvenţa de cod tipică legării în avans este de forma:

```
Dim ob As Clasa.Server
Set ob = New Server.Clasa
```

New se foloseşte împreună cu **Set** pentru crearea unei noi instanţe de clasă.

Legarea târzie

Legarea târzie (**late binding**) nu necesită setarea referinţei la biblioteca de tipuri ale aplicaţiei server. Din acest motiv aici nu se cunoaşte modelul de obiecte a aplicaţiei server şi deseori apar erori în timpul execuţie aplicaţiei. Până la instanţiere, care se face aici în momentul execuţiei aplicaţiei, VB nu are de unde să ştie tipul obiectelor de creat şi nici dacă metodele şi proprietăţile codului există sau sunt folosite corect. Problema tratării erorilor în acest caz necesită o atenţie deosebită. Avantajul acestei legări constă în independenţa codului de versiunea componentei folosite. Dacă serverul are multe versiuni, de exemplu Internet Explorer are multe versiunile 3, 4, 5, şi 6 pentru fiecare versiune există câte o bibliotecă de obiecte. În cazul legării în avans, dacă s-a scris un cod pentru o anumită versiune şi aceasta nu există pe calculator codul nu va funcţiona decât în condiţiile în care se va seta referinţă la o nouă bibliotecă. Secvenţa de cod tipică legării târzii este de forma:

```
Dim ob As Object
Set ob = CreateObject("Server.Clasa")
```

`CreateObject()` și **`GetObject()`** sunt funcții pentru a crea obiecte pe baza unui server de aplicații către care nu s-a creat o referință

15.5 Modele de obiecte

Pentru a putea programa aplicația server trebuie cunoscut destul de bine modelul de obiecte al acesteia. Pentru aplicațiile Microsoft acesta poate fi găsit în **Help** sau dedus prin folosirea navigatorului de obiecte (***Object Browser***).

15.5.1 Modelul de obiecte Excel

În *Figura 79* se prezintă un extras din modelul de obiecte a lui Microsoft Excel. Observați ca modelul pleacă de la un părinte numit Application care reprezintă aplicația Excel.

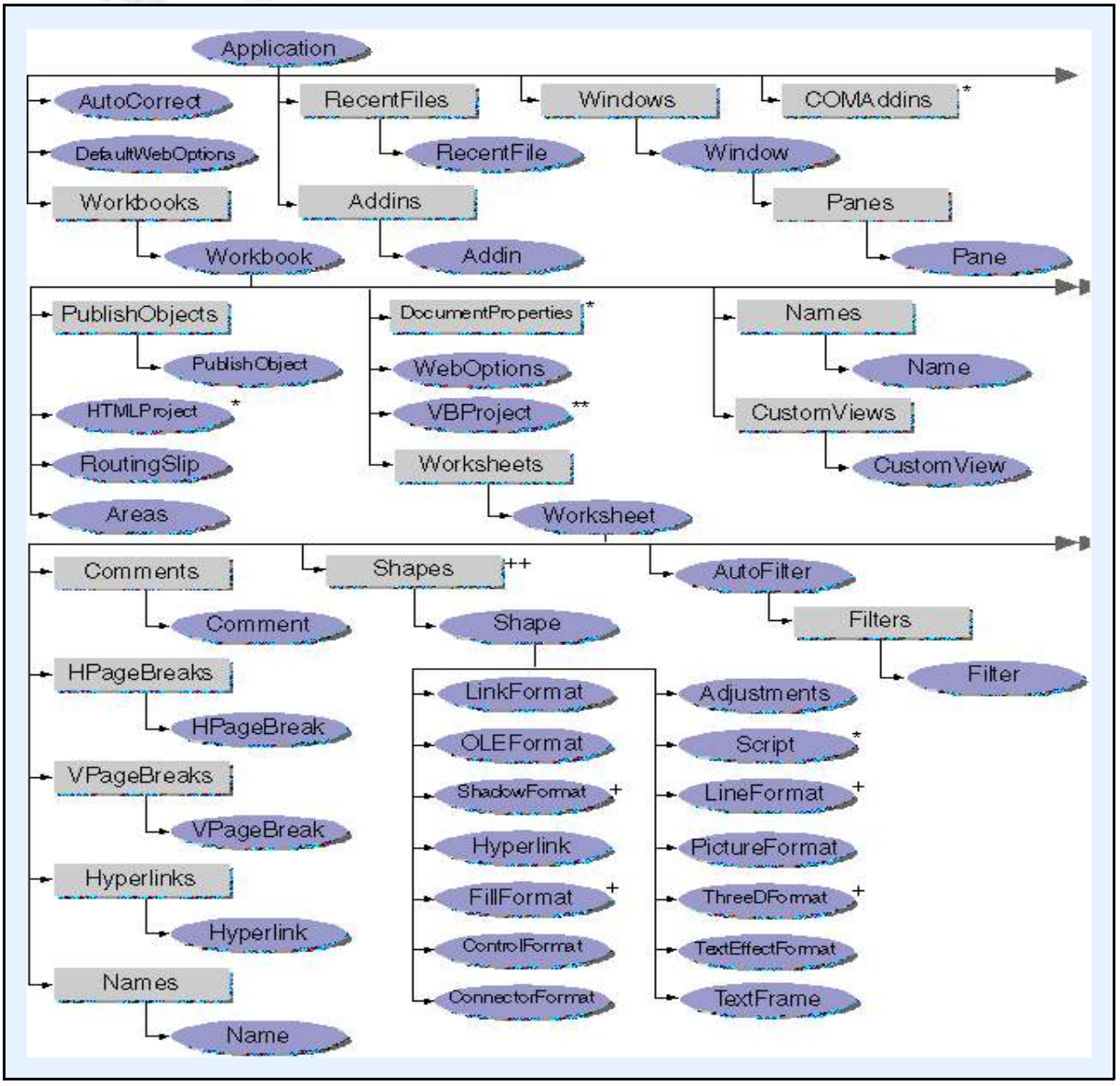

Figura 79 - Extras din modelul de obiecte a lui Excel.

Acesta este nivelul cel mai de sus al ierahiei de obiecte ale modelului. Numai acesta poate fi creat direct din VB. Una dintre proprietățile obiectului Application este colecția Workbooks care se

foloseşte pentru stocarea mai multor obiecte Workbook. Acestea nu au sens în afara lui Excel din acest motiv ele nu pot fi create independent de obiectul Application.

Fiecare Workbook are propriile lui metode şi proprietăţi, colecţia Worksheets se foloseşte pentru a referi obiectele Worksheet dintr-un Workbook. Mai departe fiecare Worksheet are propriile lui metode şi proprietăţi.

Iată un exemplu ActiveX Automation cu Excel în continuare. Pentru rularea aplicaţiei trebuie parcurse următoarele etape:

- se creează un **Proiect Standard** nou VB cu numele *Proiect1ActiveX*;
- se creează un **CommandButton** nou cu numele *Command1*;
- se face clic dublu pe *Command1*;
- din meniul *Project*, selectaţi *References*, apoi localizaţi şi bifaţi checkbox-ul corespunzător lui "**Microsoft Excel 10.0 Object Library**" sau, eventual, o versiune mai mică a bibliotecii de obiecte pentru programarea lui Excel;
- introduceţi şi apoi rulaţi codul următor:

```
Private Sub Command1_Click()
    'Declaratia variabilelor obiecte Excel
    Dim obExcel As Excel.Application
    Dim wbExcel As Excel.Workbook
    Dim wsExcel As Excel.Worksheet

    'Crearea unei noi sesiuni de lucru cu Excel
    Set obExcel = New Excel.Application
    'Noului document se adauga un Workbook
    Set wbExcel = obExcel.Workbooks.Add

    'Noului Workbook se adauga un Worksheet
    Set wsExcel = wbExcel.Worksheets.Add

    'Aplicatia Excel devine vizibila pentru utilizator
    obExcel.Visible = True
    obExcel.ActiveWindow.WindowState = xlMaximized

    On Error Resume Next
    With wsExcel
        'Numele noului Worksheet
        .Name = "Test"
        'Inserarea de valori numerice in celule
        .Cells(1, 1).Value = 1
        .Cells(2, 1).Value = 6
        .Cells(3, 1).Value = 7

        'Totalul valorilor inserate
        .Cells(5, 1).Value = "=SUM(A1:A3)"
        'Modifica culoarea interioara a
        ' unui grup de celule
```

```
        .Range("A1:A4").Interior.Color = vbCyan

        With .Cells(5, 1)
            .Font.Bold = True
            .Font.Color = vbRed
            .Borders(xlEdgeTop).LineStyle = xlContinuous
            .Borders(xlEdgeTop).Weight = xlThick
            .Select = True
        End With

    End With

    'Se activeaza sistemul de alerta Excel
    ' la parasirea aplicatiei in lipsa salvarii
    ' documentului curent
    obExcel.DisplayAlerts = True
    'Se paraseste Excel
    ' numai daca in fereastra de dialog afisata se apasa No
    obExcel.Quit
    'Se elibereaza spatiul alocat obiectului Excel
    Set obExcel = Nothing
End Sub
```


La începutul codului se declară variabilele **obExcel**, **wbExcel**, **wsExcel** în care se vor stoca referințe la obiectele Excel **Application**, **Workbook** și **Worksheet**. Apoi, se deschide o nouă sesiune de lucru cu Excel. Deoarece nu există nici un **Workbook** deschis, interfața între Excel și utilizator nu va fi vizibilă. Folosind metoda **Add** se adaugă un **Workbook** nou în care se adaugă și un **Worksheet** nou. Apoi, se completează trei celule și se face suma valorilor din ele, se modifică aspectul celulelor și a sumei, iar în final se părăsește aplicația Excel. Ca urmare a setării **obExcel.DisplayAlerts = True** atunci când se face părăsirea Excel-ului fără a fi salvat în prealabil modificările aduse documentelor se va afișa o fereastră de dialog ce permite această salvare, fie asigură evitarea închiderii Excel-ului.

15.5.2 Modelul de obiecte Word

În *Figura 80* se prezintă un extras din modelul de obiecte corespunzător lui Word. În momentul de față acest model prezintă multe asemănări cu cel a lui Excel, însă în versiunile ceva mai antice (2 sau 3, în acest moment ultima versiune este 10) deosebirile erau cele preponderente. La baza ierahiei de obiecte este și aici obiectul Application care conține colecția Documents. Obiectul Document asigură facilitățile de manipulare a documentului Word plecând de la salvare și mergând până la închiderea documentelor. Fiecare obiect Document are mai multe proprietăți folosite pentru manipularea textului cum ar fi Sections, Paragraphs, Sentences și Words. Fiecare dintre aceste proprietăți întoarce o referință la un obiect Range. Obiecte Range ale lui Word sunt conceptual similare cu cele ale lui Excel asigurând accesul la conținutul documentelor și formatarea blocurilor de text. Pentru manipularea textului unui document Word folosirea lui Range este esențială deoarece Word nu are un obiect separat pentru elementele de text fundamentale cum sunt caracterul și cuvântul. Asemenea lui Excel, Word consideră aceste elemente ca parte a unei clase generice numite obiectul Range. Un obiect Range se definește ca o porțiune de text continuă

dintr-un document şi poate fi orice plecând de la un simplu caracter până la întreg documentul. Există două modalităţi de bază pentru întoarcerea unui obiect Range: metoda **Range** a obiectului Document şi proprietatea Range.

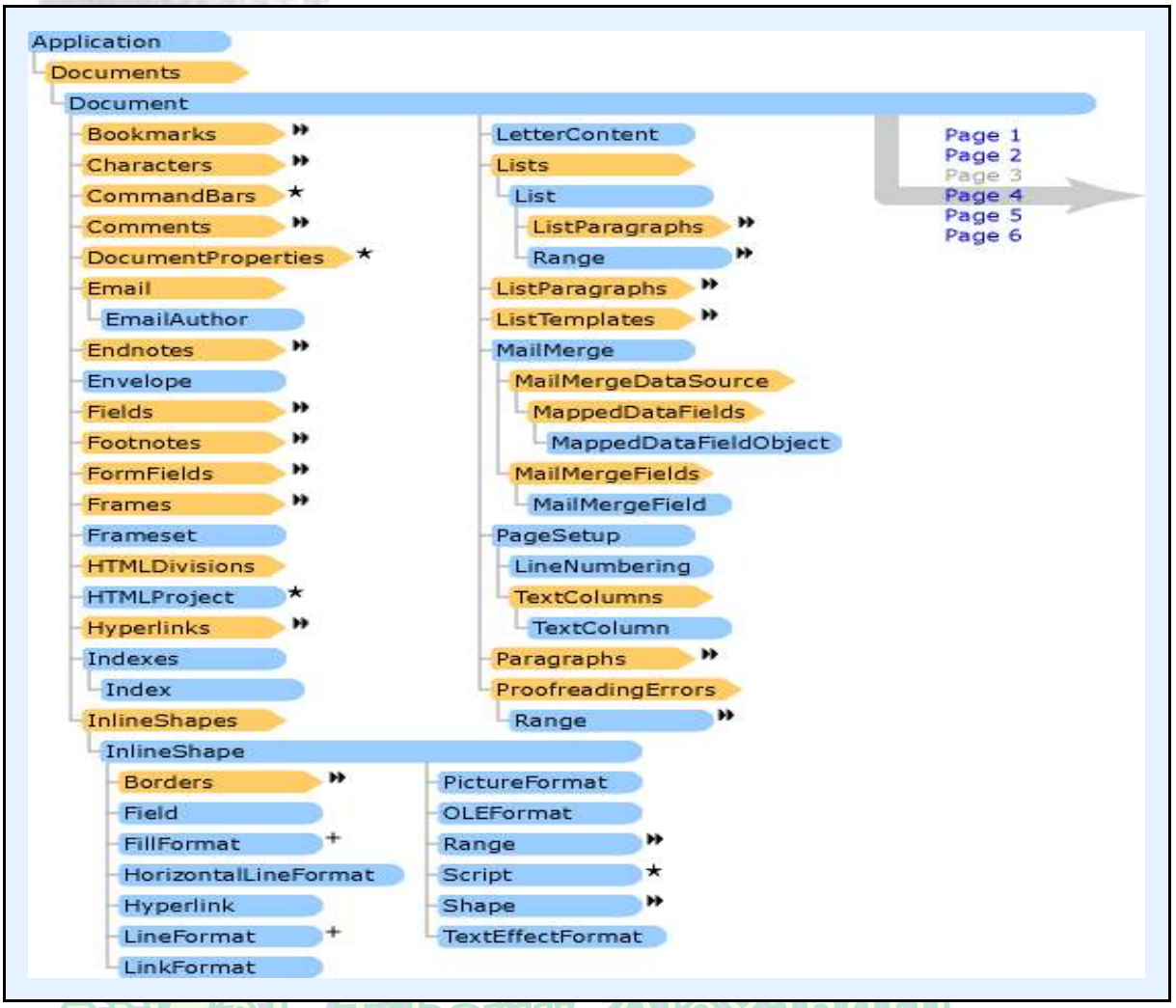

Figura 80 - Extras din modelul de obiecte a lui Word.

Iată un exemplu ActiveX Automation cu Word în continuare. Pentru rularea aplicaţiei trebuie parcurse următoarele etape:

- deschideţi **Proiectul Standard** anterior *Proiect1ActiveX*;
- se creează un **CommandButton** nou cu numele *Command2*;
- se face clic dublu pe *Command2*;
- din meniul *Project*, selectaţi *References*, apoi localizaţi şi bifaţi checkbox-ul corespunzător lui "**Microsoft Word 10.0 Object Library**" sau, eventual, o versiune mai mică a bibliotecii de obiecte pentru programarea lui Word;
- introduceţi şi apoi rulaţi codul următor:

```
Private Sub Command2_Click()
    'Declaratia variabilelor obiecte Word
    Dim obWord As Word.Application
```

```
Dim obDoc As Word.Document
Dim obRange As Word.Range
Dim obTab As Word.Table

'Crearea unei noi sesiuni de lucru Word
Set obWord = New Word.Application
'Aplicatia Word devine vizibila
obWord.Visible = True
'Se adauga un document nou
Set obDoc = obWord.Documents.Add

'Incepe inserarea datelor in document
Set obRange = obDoc.Range(0)
obRange.InsertAfter ("Catre: " & vbTab & vbTab)
obRange.InsertAfter ("Vasile ANTON" & vbCrLf)
obRange.InsertAfter ("Adresa: " & vbTab)
obRange.InsertAfter ("str. Magnoliei, nr. 13, ap. 3" & _
 vbCrLf)

'Ingrosarea textului Vasile ANTON
With obRange.Find
    .Forward = True
    .Wrap = wdFindContinue
    .Execute FindText:="Vasile ANTON"
End With
obRange.Font.Bold = True

'Ingrosarea textului str. Magnoliei, nr. 13, ap. 3
With obRange.Find
    .Forward = True
    .Wrap = wdFindContinue
    .Execute FindText:="str. Magnoliei, nr. 13, ap. 3"
End With
obRange.Font.Bold = True

Set obRange = obDoc.Range(obDoc.Range.End - 1)
obRange.ParagraphFormat.Alignment = wdAlignParagraphCenter
obRange.Font.Size = 17
obRange.InsertAfter (vbCrLf & vbCrLf & "CONSUMATIE" & _
 vbCrLf & vbCrLf)

Set obRange = obDoc.Range(obDoc.Range.End - 1)
obRange.Font.Size = 12
'Crearea unui tabel Word
Set obTab = obDoc.Tables.Add(Range:=obRange, _
 NumRows:=4, NumColumns:=4)
With obTab
    'Capul de tabel
    .Cell(1, 1).Range.Text = "PRODUS"
```

```
        .Cell(1, 2).Range.Text = "PRET"
        .Cell(1, 3).Range.Text = "CANTITATE"
        .Cell(1, 4).Range.Text = "TOTAL"
        .Rows(1).HeadingFormat = True
        'Primul rand
        .Cell(2, 1).Range.Text = "Vin rosu"
        .Cell(2, 2).Range.Text = "100000"
        .Cell(2, 3).Range.Text = "12"
        .Cell(2, 4).Formula Formula:="=b2*c2"
        .Cell(2, 4).Range.ParagraphFormat.Alignment = _
          wdAlignParagraphRight
        'Al 2-lea rand
        .Cell(3, 1).Range.Text = "Vin alb"
        .Cell(3, 2).Range.Text = "70000"
        .Cell(3, 3).Range.Text = "7"
        .Cell(3, 4).Formula Formula:="=b3*c3"
        .Cell(3, 4).Range.ParagraphFormat.Alignment = _
         wdAlignParagraphRight
        'Al 3-lea rand
        .Cell(4, 2).Merge MergeTo:=.Cell(4, 1)
        .Cell(4, 3).Merge MergeTo:=.Cell(4, 1)
        .Cell(4, 1).Formula Formula:="=(d2 + d3)"
        .Cell(4, 1).Shading.Texture = wdTexture12Pt5Percent
        .Cell(4, 1).Range.ParagraphFormat.Alignment = _
         wdAlignParagraphRight
    End With

    obWord.DisplayAlerts = True
    obWord.Quit
    Set obWord = Nothing
End Sub
```


După intrarea în Word se creează un document nou prin linia **`Set obDoc = obWord.Documents.Add`**. În continuare se creează un obiect Range folosind metoda **`Range`** a noului document creat prin linia **`Set obRange = obDoc.Range()`**. Forma generală a metodei **`Range`** este **`Range(Început, Sfârsit)`**, unde valoarea cea mai mică a lui **`Început`** este 0 şi corespunde caracterului de început al documentului. Valorile **`Început`** şi **`Sfârsit`** sunt însă opţionale, iar atunci când documentul este nou, deci este fără caractere, şi ne pregătim să-i inserăm un conţinut este corect ca ambele valori ce descriu domeniul să lipsească. Conţinutul este inserat folosind metoda **`InsertAfter(text)`** care realizează inserarea unui text la sfârşitul domeniului specificat prin obiectul Range. După inserare domeniul obiectului Range se extinde până la noul text inserat. Aplicaţia mai utilizează variabila obiect **`obTab`** ce va stoca o referinţă la un tabel de patru linii şi patru coloane ce este creat prin linia **`obDoc.Tables.Add(Range:=obRange,NumRows:=4, NumColumns:=4)`**. Prin folosirea metodei **`Cell(Rând, Coloana)`** se obţinea o referinţă la un obiect Cell. Acesta prin proprietăţile lui permite testarea sau modificarea conţinutului tabelului. Pentru alinierea la dreapta a valorilor numerice calculate în coloana 4 se foloseşte proprietatea Range astfel **`.Cell(2, 4).Range.ParagraphFormat.Alignment = wdAlignParagraphRight`**

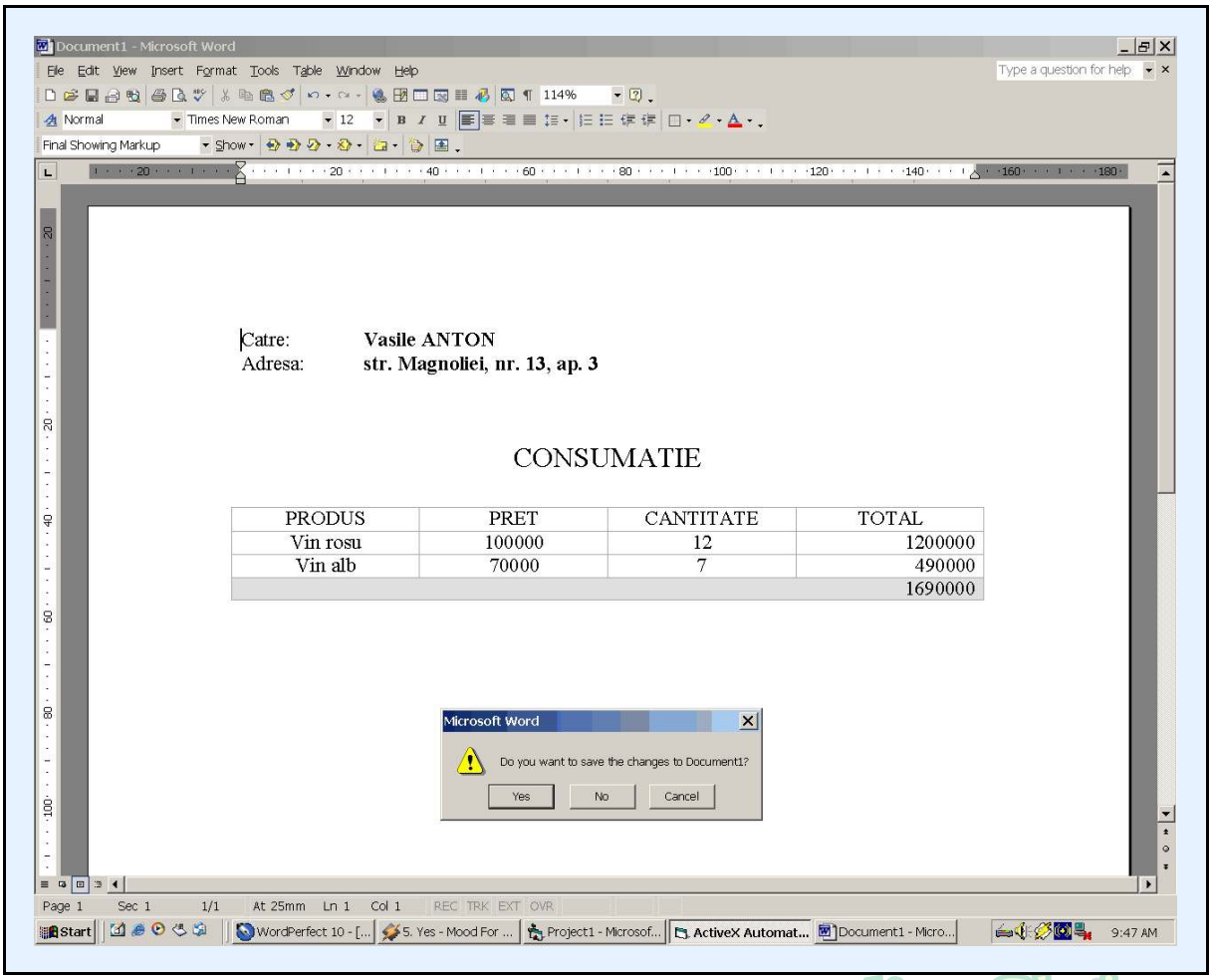

Figura 81 - Document Word creat prin ActiveX Automation din Visual BASIC.

15.5.3 Modelul de obiecte AutoCAD

În *Figura 82* se prezintă un extras din modelul de obiecte al programului de proiectare asistată pe calculator AutoCAD 2000. Observaţi că şi aici, la baza ierarhiei de obiecte stă Application, acesta are o colecţie Documents care este formată din obiecte Document. În cazul aplicaţiei AutoCAD 2000 desenarea se face în spaţiul de modelare, acestuia îi corespunde în ierarhie obiectul ModelSpace. Toate entităţile AutoCAD vor fi desenate în acest spaţiu. Pentru desene în plan, acesta este implicit planul xOy. Dacă desenarea se face în spaţiu (coordonata z a entităţilor este nenulă) este necesară stabilirea unei direcţii de vizualizare diferită de cea perpendiculară pe planul xOy în care să se poate vedea corpul sau suprafaţa desenată.

Iată un exemplu ActiveX Automation cu AutoCAD 2000 în continuare (vezi *Figura 83*). Pentru rularea aplicaţiei trebuie parcurse următoarele etape:

- deschideţi **Proiectul Standard** anterior *Proiect1ActiveX*;
- se creează un **CommandButton** nou cu numele *Command3*;
- se face clic dublu pe *Command3*;
- din meniul *Project*, selectaţi *References*, apoi localizaţi şi bifaţi checkbox-ul corespunzător lui **"AutoCAD 2000 Type Library"** bibliotecii de obiecte corespunzătoare lui AutoCAD 2000;
- introduceţi şi apoi rulaţi codul următor:

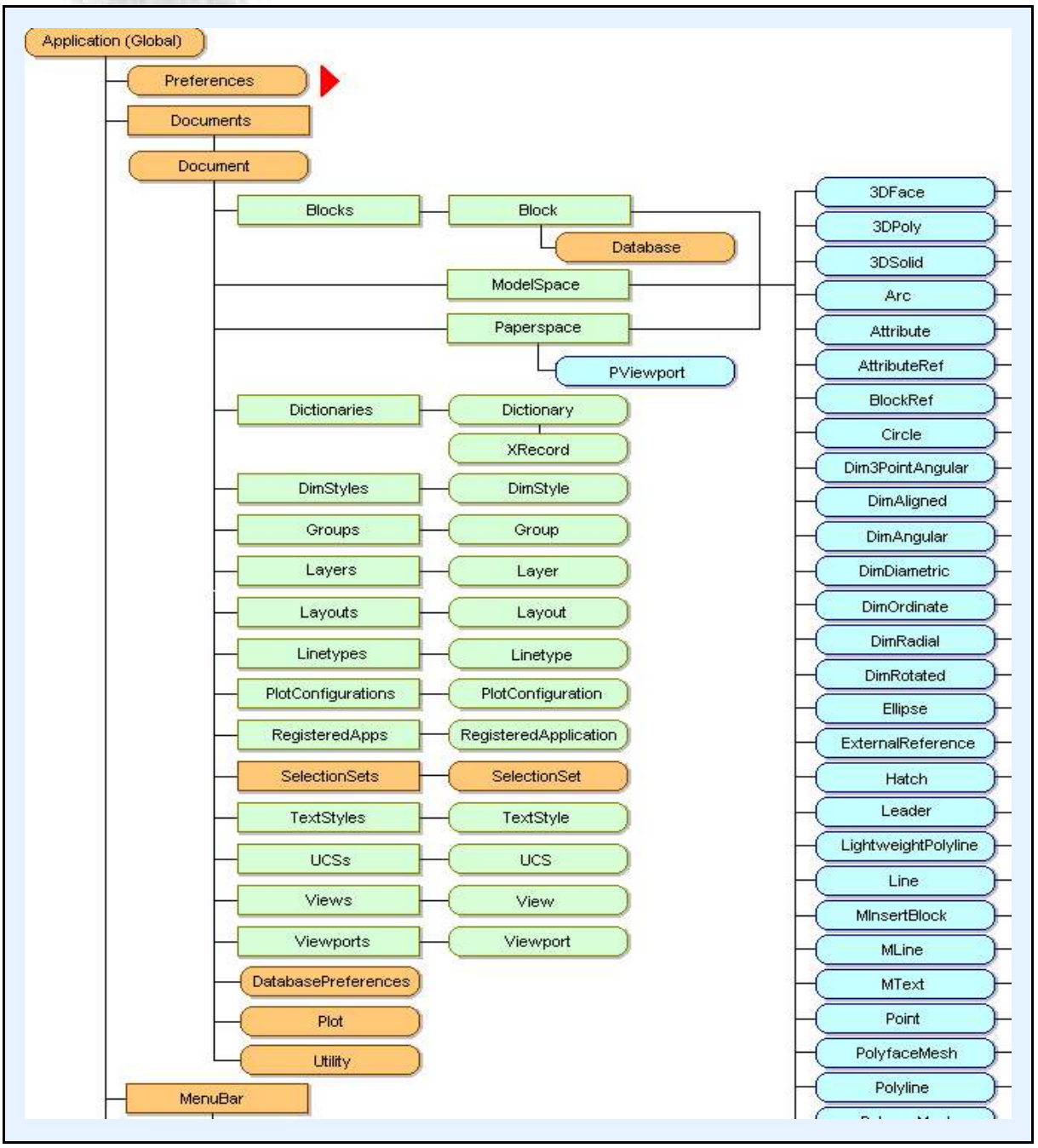

Figura 82 - Extras din modelul de obiecte al AutoCAD-ului 2000.

```
Private Sub Command3_Click()
    Dim obAcad As AcadApplication
    Dim obDoc As AcadDocument
    Dim obMSpace As AcadModelSpace

    'Linie
    Dim obLinie As AcadLine
```

```
Dim punctInceput(0 To 2) As Double
Dim punctSfarsit(0 To 2) As Double

'Cerc
Dim obCerc As AcadCircle

'Crearea unei noi sesiuni de lucru AutoCAD
On Error Resume Next
Set obAcad = GetObject(, "AutoCAD.Application")
If Err Then
    Err.Clear
    Set obAcad = CreateObject("AutoCAD.Application")
    If Err Then
     MsgBox "Nu se poate realiza conexiunea cu AutoCAD" & _
     "(se poate ca sa nu fi instalat pe calculator)!"
     Exit Sub
    End If
End If

On Error GoTo 0
'Aplicatia AutoCAD devine vizibila
obAcad.Visible = True
'Se maximizeaza fereastra aplicatiei
obAcad.WindowState = acMax
'Documentul curent
Set obDoc = obAcad.ActiveDocument
'Spatiul de modelare al documentului curent
Set obMSpace = obDoc.ModelSpace

'Desenarea unei linii din (1,2,0) in (5,7,0)
' Punctul de inceput
punctInceput(0) = 1
punctInceput(1) = 2
punctInceput(2) = 0

'Punctul de sfarsit
punctSfarsit(0) = 5
punctSfarsit(1) = 7
punctSfarsit(2) = 0

'Desenarea liniei
Set obLinie = obMSpace.AddLine(punctInceput, punctSfarsit)
obLinie.Update

'Desenarea cercului
Set obCerc = obMSpace.AddCircle(punctSfarsit, 3)
obCerc.Update

obAcad.ZoomExtents
```

```
    Call suprAcad(obMSpace, obDoc)

    'obAcad.Quit
    'Set obAcad = Nothing
End Sub

Private Sub suprAcad(obMSpace As AcadModelSpace, _
                                        obDoc As AcadDocument)
    Dim meshObj1 As AcadPolygonMesh
    Dim m As Integer, n As Integer
    Dim x01 As Double, x02 As Double, pasx As Double
    Dim y01 As Double, y02 As Double, pasyz As Double
    Dim puncte() As Double
    Dim c As Long
    Dim directieNoua(0 To 2) As Double

    m = 40
    n = 40
    x01 = -4
    x02 = 4
    pasx = (x02 - x01) / m

    y01 = -4
    y02 = 4
    pasy = (y02 - y01) / n

    ReDim puncte(3 * (m + 1) * (n + 1) - 1) As Double

    c = 0
    For j = 0 To m
        For i = 0 To n
            x = x01 + j * pasx
            y = y01 + i * pasy

            puncte(c) = x
            puncte(c + 1) = y
            puncte(c + 2) = 5 * Sin(1 + x ^ 2 + y ^ 2) / _
                            (1 + x ^ 2 + y ^ 2)
            c = c + 3
        Next i
    Next j

    'Crearea unui 3Dmesh in spatiul de modelare
    Set meshObj1 = obMSpace.Add3DMesh(m + 1, n + 1, puncte)
    'Modifica culoarea suprafetei la rosu
    meshObj1.Color = acRed

    'Modifica directia de vizualizare
```

```
    ' pentru ca suprafata sa poata fi vazuta mai bine
    directieNoua(0) = 1
    directieNoua(1) = 1
    directieNoua(2) = 1
    obDoc.ActiveViewport.Direction = directieNoua
    obDoc.ActiveViewport = obDoc.ActiveViewport

    ZoomExtents
End Sub
```

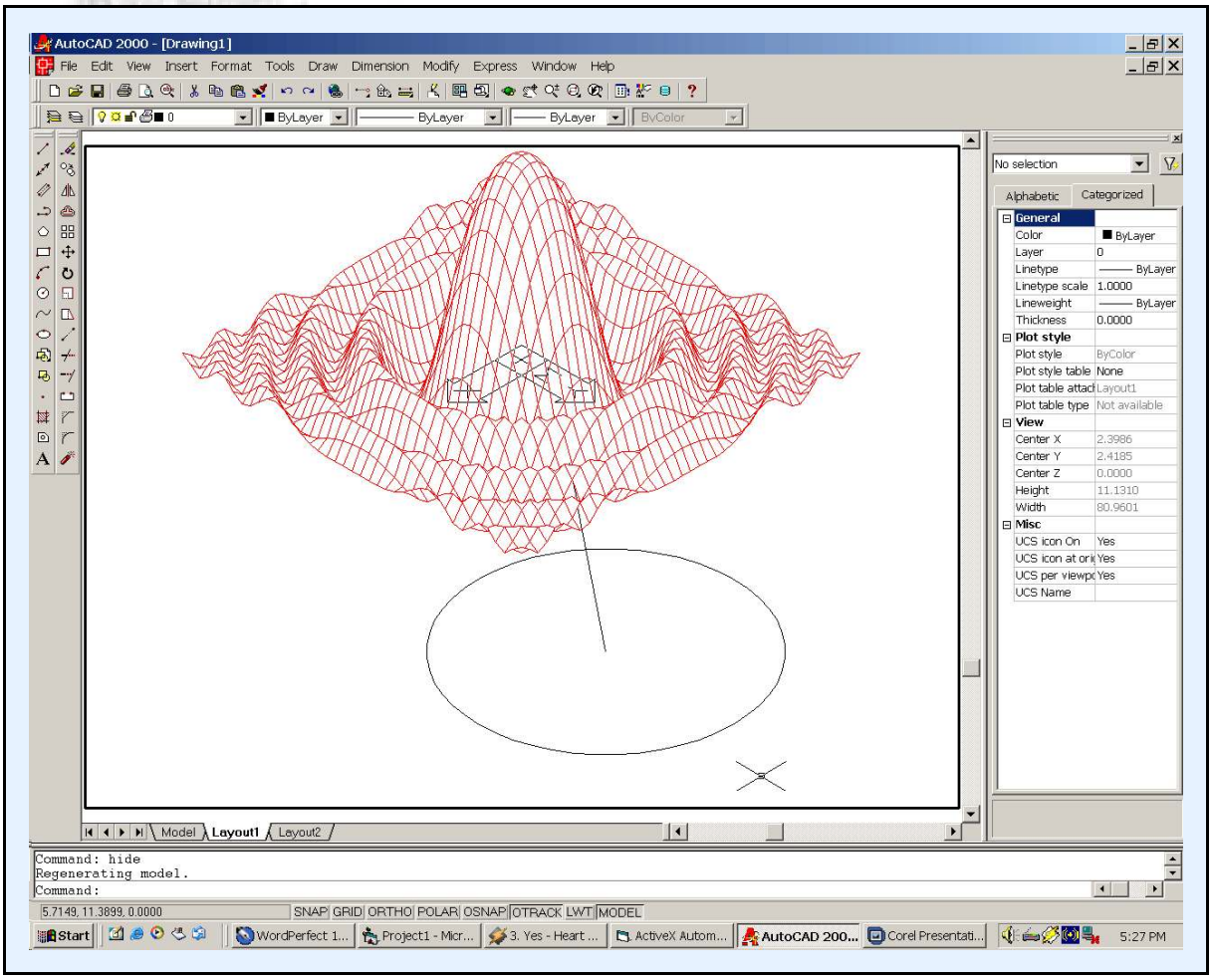

Figura 83 - Desenarea unor obiecte AutoCAD simple folosind ActiveX Automation.

Aplicaţia va desena o linie, apoi un cerc şi în final o suprafaţă. Pentru reprezentarea grafică a suprafeţei se foloseşte entitatea AutoCAD 3DMesh care se defineşte printr-o reţea de puncte pe care AutoCAD le va uni pentru a genera suprafaţa. Spaţiul pentru stocarea acestor puncte se alocă dinamic în procedura **`Sub suprAcad()`**.

În continuare se prezintă o aplicaţie care generează un corp solid AutoCAD. Corpul solid este o roată dinţată cu dinţi drepţi cu deplasare de profil x = 0 ce are dimensiunile geometrice definite prin grupul de formule prezentate în continuare (vezi şi *Figura 85*):

z - numărul de dinți;
α_0 - unghiul de profil al cremalierei de referință;
h_a - coeficientul de înălțime al capului dintelui.
r - raza cercului de divizare;
r_a - raza cercului de cap;
r_b - raza cercului de bază;
α_a - unghiul de angrenare pe capul dintelui.

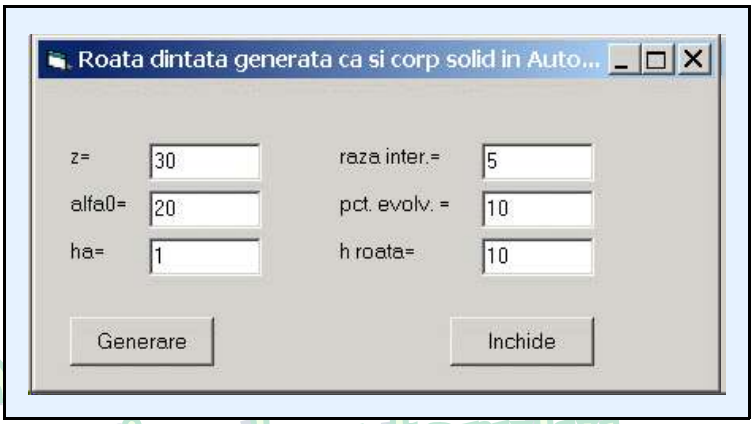

Figura 84 - Datele de intrare pentru aplicația de generare a roții dințate în AutoCAD 2000.

Datele de intare a aplicației se introduc prin formularul din *Figura 84*.

$$r = \frac{z}{2},\ r_a = r + h_a,\ r_b = r\cos(\alpha_a),\ \alpha_a = \operatorname{acos}\left(\frac{r_b}{r_a}\right),\ \operatorname{inv}(\alpha) = \tan(\alpha) - \alpha$$

Arcul AB

Punctul A

$$\theta + \gamma = \frac{\pi}{z}\left(2 - \frac{1}{2\cos(\alpha_0)}\right) - \operatorname{inv}(\alpha_0)$$

$$x_A = -r_b \sin(\theta + \gamma)$$

$$y_A = r_b \cos(\theta + \gamma)$$

Punctul B

$$\theta = \frac{\pi}{2z} + \operatorname{inv}(\alpha_0)$$

$$x_A = -r_b \sin(\theta)$$

$$y_A = r_b \cos(\theta)$$

Arcul BC - flancul evolventic

$$x(\alpha) = -\frac{r_b}{\cos(\alpha)}\sin(\theta - \operatorname{inv}(\alpha))$$

$$y(\alpha) = -\frac{r_b}{\sin(\alpha)}\cos(\theta - \operatorname{inv}(\alpha))$$

$$0^0 < \alpha < \alpha_a$$

Arcul CD

Punctul C

$$\lambda = \frac{\pi}{2z} - inv(\alpha_a) + inv(a_0)$$

$$x_C = -r_a \sin(\lambda)$$

$$y_C = r_a \cos(\lambda)$$

Punctul D

$$\lambda = 0$$

$$x_D = 0$$

$$y_D = r_a$$

```
Private Sub Command1_Click()
    Dim z As Double
    Dim a0 As Double
    Dim ha As Double
    Dim r As Double
    Dim rb As Double
    Dim ra As Double
    Dim aa As Double
    Dim pi As Double
    Dim gamateta As Double
    Dim xa As Double
    Dim ya As Double
    Dim teta As Double
    Dim xb As Double
    Dim yb As Double
    Dim arcab As AcadArc
    Dim arccd As AcadArc
    Dim p3d(0 To 2) As Double
    Dim p3d1(0 To 2) As Double
    Dim p3d2(0 To 2) As Double
    Dim i As Integer
    Dim c As Integer
    Dim alfa As Double
    Dim linevol As AcadLine
    Dim lambda As Double
    Dim pline As AcadLWPolyline
    Dim pline1 As AcadLWPolyline
    Dim p2d() As Double
    Dim div As Integer
    Dim ssetObj As AcadSelectionSet
    Dim mode As Integer
    Dim ob(5) As AcadEntity
    Dim linie As AcadLine
    Dim linie1 As AcadLine
    Dim hroata As Double
    Dim regionObj As Variant
    Dim cil As Acad3DSolid
```

```
Dim rinter As Double

Dim obAcad As AcadApplication
Dim obDoc As AcadDocument
Dim obMSpace As AcadModelSpace

'crearea unei noi sesiuni de lucru AutoCAD
On Error Resume Next
Set obAcad = GetObject(, "AutoCAD.Application")
If Err Then
    Err.Clear
    Set obAcad = CreateObject("AutoCAD.Application")
    If Err Then
     MsgBox "Nu se poate realiza conexiunea cu AutoCAD" & _
     "(se poate ca sa nu fi instalat pe calculator)!"
     Exit Sub
    End If
End If

On Error GoTo 0
'Aplicatia AutoCAD devine vizibila
obAcad.Visible = True
'Se maximizeaza fereastra aplicatiei
obAcad.WindowState = acMax
'Documentul curent
Set obDoc = obAcad.ActiveDocument
'Spatiul de modelare al documentului curent
Set obMSpace = obDoc.ModelSpace

On Error GoTo iesire
rinter = CDbl(txtrinter)
pi = 4# * Atn(1#)
z = CDbl(txtz)
'a0 transfomat din grade in radiani
a0 = CDbl(txta0) / 180# * pi
ha = CDbl(txtha)
div = CInt(txtdiv)
hroata = CDbl(txthroata)

r = z / 2#
rb = r * Cos(a0)
ra = r + ha
aa = Arccos(rb / ra)

'Arcul AB
' punctul A
gamateta = pi / z * (2# - 1# / 2# / Cos(a0)) - Tan(a0) _
 + a0 + 0.1 / div
```

```
xa = -rb * Sin(gamateta)
ya = rb * Cos(gamateta)

' punctul B
teta = pi / 2# / z + Tan(a0) - a0
xb = rb * Sin(teta + pi / 2#)
yb = rb * Cos(teta + pi / 2#)

'Centrul arcului AB
p3d(0) = 0: p3d(1) = 0: p3d(2) = 0
'Trasarea arcului
Set arcab = obMSpace.AddArc(p3d, rb, teta + pi / 2, _
            gamateta + pi / 2)

'Punctele profilului evolventic BC
c = 1
ReDim p2d(c)
For alfa = 0 To aa Step aa / div
    ReDim Preserve p2d(c)
    p2d(c - 1) = xevol(teta, alfa, rb)
    p2d(c) = yevol(teta, alfa, rb)
    c = c + 2
Next alfa

'Generaea unei pline evolventice
Set pline = obMSpace.AddLightWeightPolyline(p2d)

'Arcul 2 x CD
lambda = pi / 2# / z - Tan(aa) + Tan(a0) + aa - a0
p3d(0) = 0: p3d(1) = 0: p3d(2) = 0
Set arccd = obMSpace.AddArc(p3d, ra, -lambda + pi / 2, _
            lambda + pi / 2)

'Linia in raport cu care se face oglindirea
' profilului evolventic
p3d1(0) = 0: p3d1(1) = 0: p3d1(2) = 0
p3d2(0) = 0: p3d2(1) = 1: p3d2(2) = 0

'Oglindirea profilului evolventic
Set pline1 = pline.Mirror(p3d1, p3d2)

Set ob(1) = arcab
Set ob(2) = pline
Set ob(3) = arccd
Set ob(4) = pline1

p3d1(0) = 0: p3d1(1) = 0: p3d1(2) = 0
p3d2(0) = xa: p3d2(1) = ya: p3d2(2) = 0
```

```
'Linie de la centru - (0, 0, 0) la A
Set linie = obMSpace.AddLine(p3d1, p3d2)

xb = rb * Sin(teta)
yb = rb * Cos(teta)
p3d1(0) = xb: p3d1(1) = yb: p3d1(2) = 0
p3d2(0) = 0: p3d2(1) = 0: p3d2(2) = 0
'Linie de la ultimul punct al profilului evolventic
' la centru - (0, 0, 0)
Set linie1 = obMSpace.AddLine(p3d1, p3d2)

Set ob(0) = linie
Set ob(5) = linie1

'Generarea unei regiuni AutoCAD
' deoarece avem un contur inchis
regionObj = obMSpace.AddRegion(ob)

'Multiplicarea regiunii corespunzatoare unui dinte
' de z ori
retval = regionObj(0).ArrayPolar(z, 2 * pi, p3d2)

'Reuniunea tuturor dintilor
Dim rg1 As AcadRegion, rg11 As AcadRegion
Dim rg2 As AcadRegion
Set rg1 = retval(LBound(retval))
For i = LBound(retval) + 1 To UBound(retval)
    Set rg2 = retval(i)
    rg1.Boolean acUnion, rg2
Next i
rg1.Boolean acUnion, regionObj(0)

'Extrudarea regiunii
Dim solidObj As Acad3DSolid
Set solidObj = obMSpace.AddExtrudedSolid(rg1, hroata, 0)

'Definirea unui cilindru
p3d2(0) = 0
p3d2(1) = 0
p3d2(2) = hroata / 2
Set cil = obMSpace.AddCylinder(p3d2, rinter, hroata)

'Scaderea cilindrului din roata
' rezulta alezajul rotii dintate
solidObj.Boolean acSubtraction, cil
rg1.Delete

ZoomAll
```

```
    Exit Sub
iesire:
    MsgBox Err.Description
    Exit Sub
End Sub

Private Sub Command2_Click()
    Unload Me
End Sub

Public Function Arccos(ByVal x As Double) As Double
    Arccos = Atn(-x / Sqr(-x * x + 1)) + 2 * Atn(1)
End Function

Public Function xevol(ByVal teta As Double, ByVal alfa As Double,
ByVal rb As Double) As Double
    xevol = -rb / Cos(alfa) * Sin(teta - Tan(alfa) + alfa)
End Function

Public Function yevol(ByVal teta As Double, ByVal alfa As Double,
ByVal rb As Double) As Double
    yevol = rb / Cos(alfa) * Cos(teta - Tan(alfa) + alfa)
End Function
```


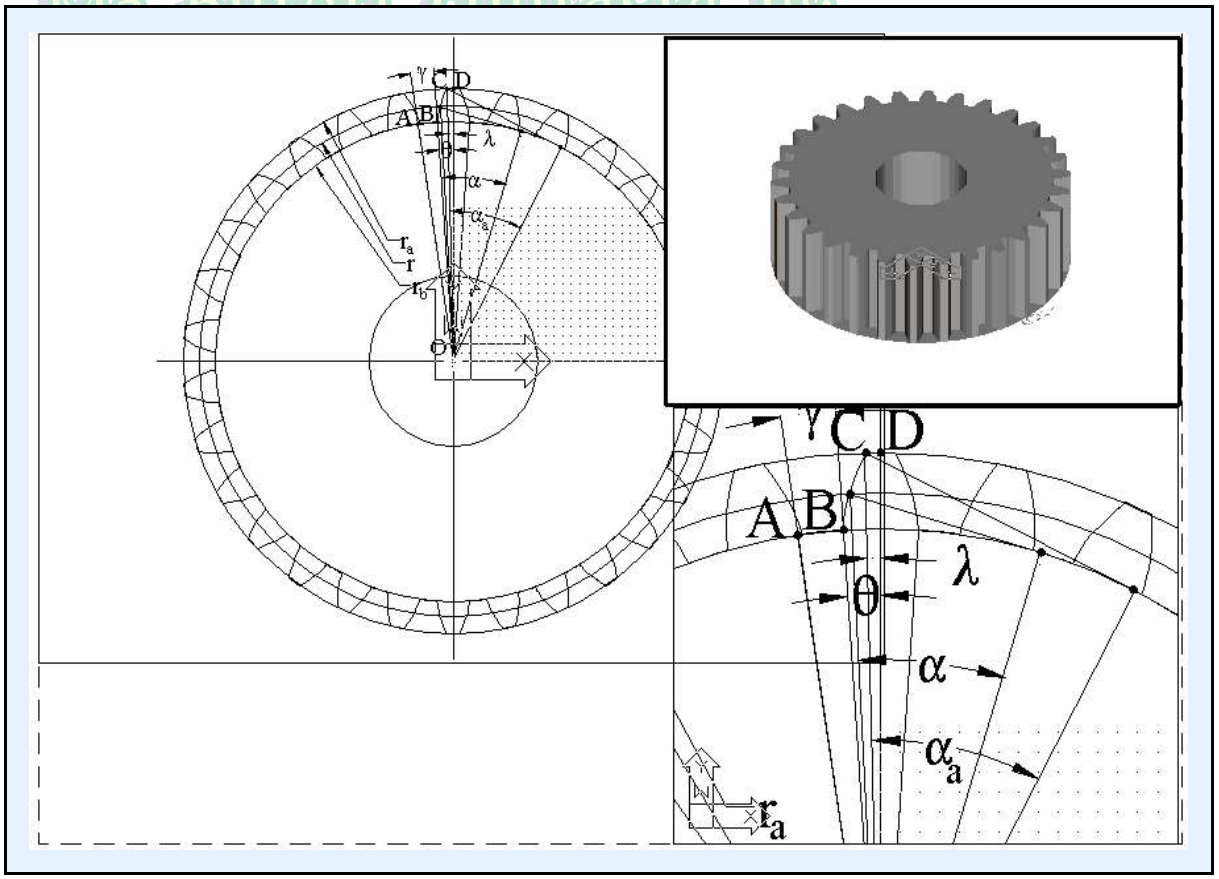

Figura 85 - Roată dințată generată sub formă de solid în AutoCAD 2000.

15.5.4 Modelul de obiecte CATIA

Ultima aplicație care se prezintă este legată de generarea unei suprafețe în CATIA V5R11. Un extras din modelul de date prezentat de acest soft apare în *Figura 86*. La fel ca și în celelalte cazuri, rădăcina modelului de obiecte CATIA este Application. Acesta are o colecție Documents care conține toate documentele deschise în cadrul sesiunii de lucru curente. O proprietate importantă a lui Application este ActiveDocument care reprezintă documentul în curs de editare. Aplicația care urmează să fie prezentată generează o suprafață dată prin ecuații parametrice folosind facilitatea de **Loft** a lui CATIA. Suprafața generată se observă în *Figura 87*.

Pentru rularea aplicației trebuie parcurse următoarele etape:

- deschideți **Proiectul Standard** anterior *Proiect1ActiveX*;
- se creează un **CommandButton** nou cu numele *Command5*;
- se face clic dublu pe *Command5*;
- din meniul *Project*, selectați *References*, apoi localizați și bifați checkbox-urile corespunzătoare lui **"CATIA V5 MecModeInterfaces Object Library"**, **"CATIA V5 InfInterfaces Object Library"** și lui **"CATIA V5 GSMInterfaces Object Library"**;
- creați următorul modul de cod, apoi :

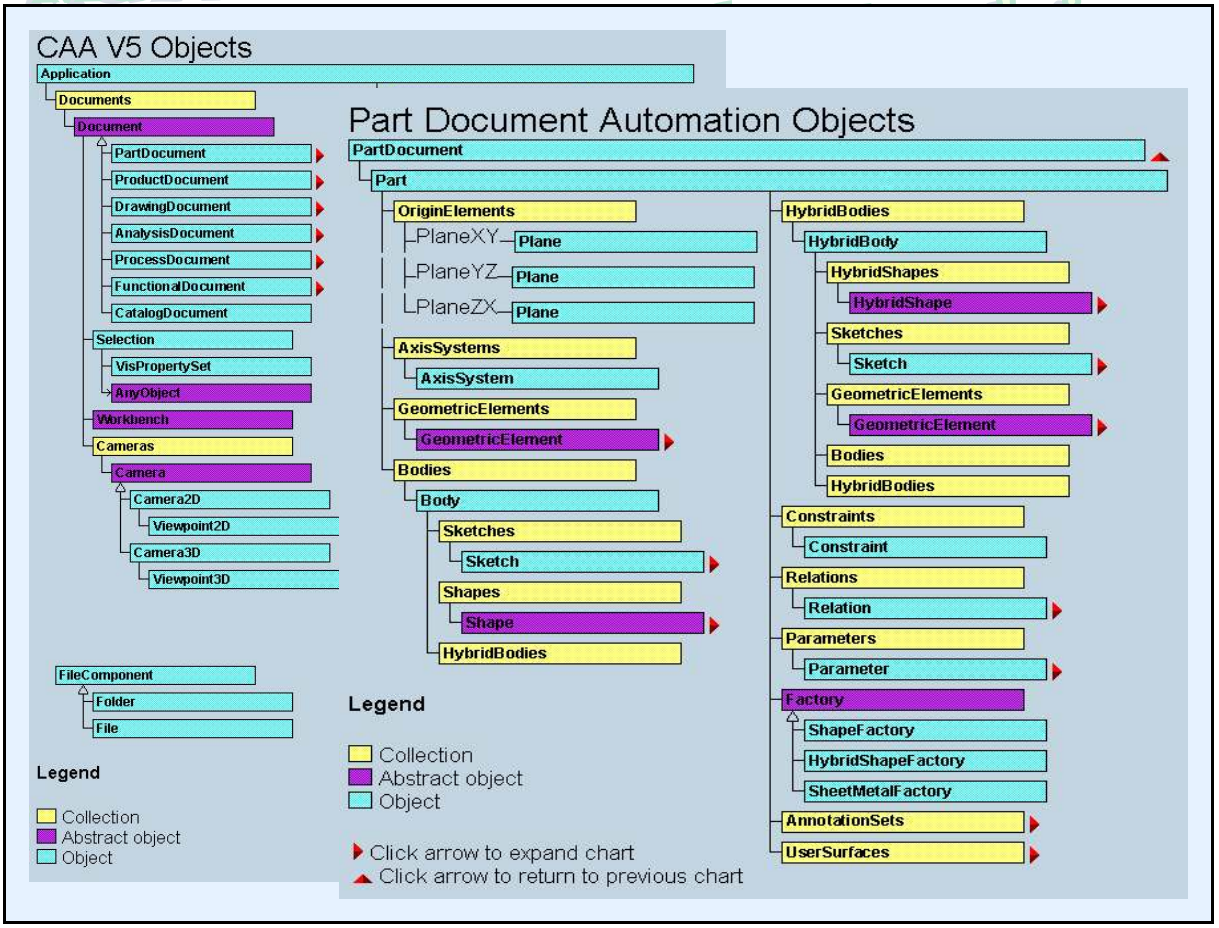

Figura 86 - Extras din modelul de obiecte a lui CATIA V5R11.

```
Public pi As Double
Public alfa0 As Double
Public h As Double
Public m As Double
Public z As Double
Public rb As Double

Public Function asin(x As Double) As Double
    asin = Atn(x / Sqr(1 - x * x))
End Function

Public Function acos(x As Double) As Double
    acos = Atn(-x / Sqr(1 - x * x)) + 2 * Atn(1)
End Function

'Urmeaza definitia suprafetei de reprezentat
Public Function xi(rb As Double, psi As Double, _
                   teta As Double) As Double
    xi = rb * (Cos(teta) + psi * Sin(teta))
End Function

Public Function yi(rb As Double, psi As Double, teta As Double)
As Double
    yi = rb * (Sin(teta) - psi * Cos(teta))
End Function

Public Function zi(h As Double, psi As Double, teta As Double)
As Double
    zi = h * (psi - teta)
End Function

Sub generare()
    Dim m As Integer, n As Integer
    Dim f01 As Double, f02 As Double, pasf As Double, f As Double
    Dim p01 As Double, p02 As Double, pasp As Double, p As Double
    Dim c As Long
    Dim i As Integer, j As Integer
    Dim pt As String

    'Declararea unui obiect CATIA
    Dim obCATIA As Object
    'Declararea unei parti de document CATIA
    Dim partDocument1 As PartDocument
    'Declaratia elementelor de geometrie necesare
    ' generarii suprafetei
    Dim hybridShapePointCoord1 As HybridShapePointCoord
    Dim hybridShapeSpline1 As HybridShapeSpline
    Dim hybridShapes1 As HybridShapes
    Dim reference1 As Reference
```

```
'Crearea unei noi sesiuni de lucru CATIA
On Error Resume Next

Set obCATIA = GetObject(, "CATIA.Application")
If Err Then
    Err.Clear
    Set obCATIA = CreateObject("CATIA.Application")
    If Err Then
       MsgBox "Nu se poate realiza conexiunea cu CATIA" & _
       "(se poate ca sa nu fi instalat pe calculator)!"
       Exit Sub
    End If
End If

'Fereastra CATIA se face vizibila
obCATIA.Visible = True
'Se adauga o parte noua
obCATIA.Documents.Add ("Part")
'Referinta la documentul activ
Set partDocument1 = obCATIA.ActiveDocument
Set part1 = partDocument1.Part
Set hybridShapeFactory1 = part1.HybridShapeFactory
Set hybridBodies1 = part1.HybridBodies
Set hybridBody1 = hybridBodies1.Add()
part1.Update

Call init
m = 10
n = 10
f01 = 0
f02 = gtor(360)
pasf = (f02 - f01) / m

p01 = 0
p02 = gtor(360)
pasp = (p02 - p01) / n

c = 1
'Punctele primei curbe spline
For i = 0 To m
    f = f01 + i * pasf
    p = p02
    Call AdaugaPunct(xi(rb, p, f), yi(rb, p, f), _
                     zi(h, p, f), hybridShapeFactory1, part1)
Next i
'Crearea primei curbe spline
Call AdaugaSpline(part1, hybridShapeFactory1, c, m)

'Punctele celei de a doua curbe spline
```

```
    For i = 0 To n
        f = f02
        p = p01 + i * pasp
        Call AdaugaPunct(xi(rb, p, f), yi(rb, p, f), _
                         zi(h, p, f), hybridShapeFactory1, part1)
    Next i
    'Crearea celei de a doua curbe spline
    Call AdaugaSpline(part1, hybridShapeFactory1, c, n)

    'Crearea unui loft cu cele
    ' 2 curbe spline
    Dim hybridShapeLoft1 As HybridShapeLoft
    Set hybridShapeLoft1 = hybridShapeFactory1.AddNewLoft()
    hybridShapeLoft1.SectionCoupling = 1
    hybridShapeLoft1.Relimitation = 1
    hybridShapeLoft1.PlaneDetection = 1
    Set hybridBodies1 = part1.HybridBodies
    Set hybridBody1 = hybridBodies1.Item("Open_body.1")
    Set hybridShapes1 = hybridBody1.HybridShapes
    Set hybridShapeSpline1 = hybridShapes1.Item("Spline.1")
    Set reference1 = _
        part1.CreateReferenceFromObject(hybridShapeSpline1)
    hybridShapeLoft1.AddSectionToLoft reference1, -1, Nothing
    Dim hybridShapeSpline2 As HybridShapeSpline
    Set hybridShapeSpline2 = hybridShapes1.Item("Spline.2")
    Dim reference2 As Reference
    Set reference2 = _
        part1.CreateReferenceFromObject(hybridShapeSpline2)
    hybridShapeLoft1.AddSectionToLoft reference2, -1, Nothing
    hybridBody1.AppendHybridShape hybridShapeLoft1
    part1.InWorkObject = hybridShapeLoft1
    part1.Update

    'Fit all in pentru ca sa se vada toata suprafata
    Dim specsAndGeomWindow1 As SpecsAndGeomWindow
    Set specsAndGeomWindow1 = obCATIA.ActiveWindow
    Dim viewer3D1 As Viewer3D
    Set viewer3D1 = specsAndGeomWindow1.ActiveViewer
    viewer3D1.Reframe
    Dim viewpoint3D1 As Viewpoint3D
    Set viewpoint3D1 = viewer3D1.Viewpoint3D

End Sub

Public Function gtor(x As Double) As Double
    gtor = x / 180# * 4 * Atn(1#)
End Function

Public Sub init()
```

```
    Dim m As Double
    pi = 4# * Atn(1#)
    alfa0 = gtor(20)
    m = 4
    z = 31
    rb = m * z * Cos(alfa0)
    h = m * z * Tan(alfa0) / 2
End Sub

Public Sub AdaugaPunct(x As Double, y As Double, z As Double, _
 ByVal hybridShapeFactory1 As Object, ByVal part1 As Object)
    Set hybridShapePointCoord1 = _
                  hybridShapeFactory1.AddNewPointCoord(x, y, z)
    Set hybridBodies1 = part1.HybridBodies
    Set hybridBody1 = hybridBodies1.Item("Open_body.1")
    hybridBody1.AppendHybridShape hybridShapePointCoord1
    part1.InWorkObject = hybridShapePointCoord1
    part1.Update
End Sub

Public Sub AdaugaSpline(ByVal part1 As Object, _
            ByVal hybridShapeFactory1 As Object, _
            ByRef c As Long, ByVal n As Integer)
    Dim i As Integer
    Set hybridBodies1 = part1.HybridBodies
    Set hybridBody1 = hybridBodies1.Item("Open_body.1")
    Set hybridShapes1 = hybridBody1.HybridShapes

    Set hybridShapeSpline1 = hybridShapeFactory1.AddNewSpline()
    hybridShapeSpline1.SetSplineType 0
    hybridShapeSpline1.SetClosing 0
    Set hybridShapes1 = hybridBody1.HybridShapes
    For i = 0 To n
       pt = "Point." & Trim(Str(c))
       Set hybridShapePointCoord1 = hybridShapes1.Item(pt)
       c = c + 1
       Set reference1 = _
        part1.CreateReferenceFromObject(hybridShapePointCoord1)
       ybridShapeSpline1.AddPointWithConstraintExplicit _
        reference1, Nothing, -1#, 1, Nothing, 0#
    Next i
    hybridBody1.AppendHybridShape hybridShapeSpline1
    part1.InWorkObject = hybridShapeSpline1
    part1.Update
End Sub
```

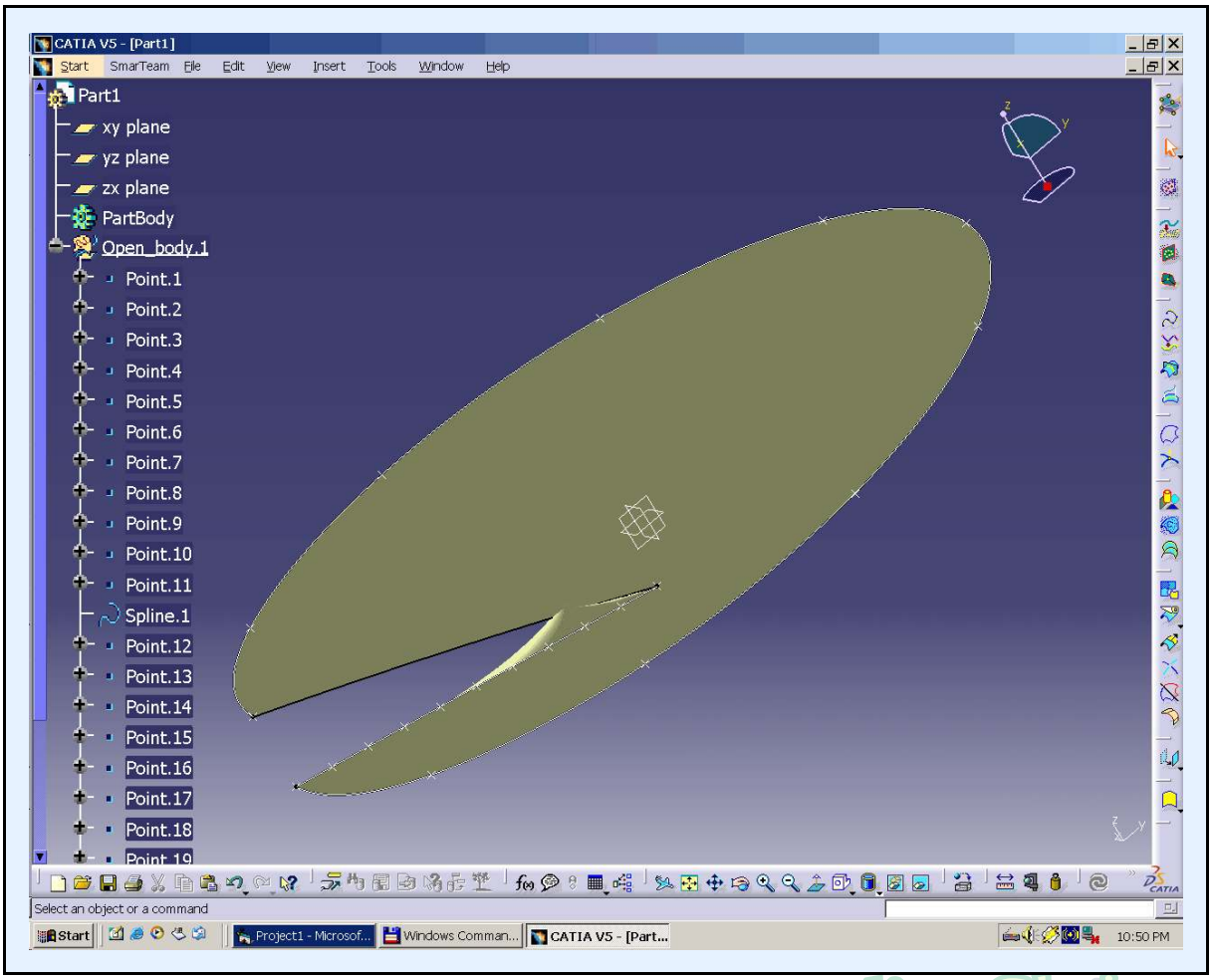

Figura 87 - Suprafață în CATIA V5R11 generată cu Loft.

în final faceți click dublu pe *Command5* și introduceți codul:

```
Private Sub Command5_Click()
    generare
End Sub
```

Pentru a rula aplicația apăsați **<F5>**, apoi faceți clic pe butonul de comandă *Command5*.