

ANEXOS Documento 2 Tomo II

Proceso de diseño y desarrollo de productos mediante técnica DFMA aplicando agentes inteligentes

Eduardo Barea Escobar

I.T. en Diseño Industrial

Tutor: Juan R. Lama Ruiz

Escuela Politécnica Superior, Universidad de Sevilla

Índice general

1. Código Fuente - DFMAsoft.py	1
--------------------------------	---

1 Código Fuente - DFMAsoft.py

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3  #
4  #
5  # Copyright 2013 Eduardo Barea <bareaescobar@gmail.com>
6  #
7  # This program is free software; you can redistribute it and/or
   modify
8  # it under the terms of the GNU General Public License as
   published by
9  # the Free Software Foundation; either version 2 of the License,
   or
10 # (at your option) any later version.
11 #
12 # This program is distributed in the hope that it will be useful
   ,
13 # but WITHOUT ANY WARRANTY; without even the implied warranty of
14 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
15 # GNU General Public License for more details.
16 #
17 # You should have received a copy of the GNU General Public
   License
18 # along with this program; if not, write to the Free Software
19 # Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
20 # MA 02110-1301, USA.
21 #
22 #
23
24 import spade
25 from easygui import *
26 import sys
27 import os
28 import time
29 from xml.sax.handler import ContentHandler
30 import xml.sax
```

```

31 from xml.dom import minidom
32 from xml.etree import ElementTree as ET
33 from xlrd import open_workbook
34 import xlwt
35 from xlutils.copy import copy
36 try:
37     from occmodel import Tools
38     from occmodelviewer import viewer
39 except:
40     print "No se pueden cargar módulos occmodel"
41
42 class inicio:
43     """
44     Este class inicia el programa, muestra la ventana de inicio,
45     muestra la ventana de selección de archivo, comprueba el
46     documento es
47     compatible con el formato XML y si es así inicia los agentes
48     dfma y
49     representación.
50     """
51     def __init__(self):
52         self.ventana_inicio()
53
54     class ventana_inicio:
55         def __init__(self):
56             print "Comienza ventana inicio . . ."
57             self.ventana_show()
58
59         def ventana_show(self):
60             print "Se va a cargar la ventana inicio"
61             msg = """
62             Software para la interpretación de ficheros XML-STEP
63             Bajo paradigma multiagente para el análisis del
64             Diseño para la Fabricación y Ensamblaje.
65             Desarrollado por Eduardo Barea
66             Escuela Politécnica Superior de Sevilla."""
67             image = "logo.gif"
68             choices = ["Aceptar"]
69             title = "DFMAsoft para XML-STEP"
70             reply = buttonbox( msg, image = image, choices =
71                             choices, title=title)

```

```
70         self.ventana_end()
71
72     def ventana_end(self):
73         print "fin ventana inicio."
74         inicio.ventana_seleccion()
75
76     class ventana_seleccion:
77     def __init__(self):
78         print "Comienza ventana selección . . ."
79         self.ventana_show()
80
81     def ventana_show(self):
82         print "Se va a cargar la ventana de selección"
83         msg = "Por favor seleccione el fichero CAD en formato
            XML-STEP"
84         title = "DFMAsoft – Seleccion de fichero XML-STEP"
85         filetypes = [ "*.step", "*.stp", "*.xml", "XML-STEP" ]
            ]
86         global fichero
87         fichero = fileopenbox(msg=msg, title=title, default='
            *', filetypes=filetypes)
88         print fichero
89         self.ventana_end()
90
91     def ventana_end(self):
92         """
93         aquí se parsea el documento para comprobar si es un
            fichero xml
94         compatible
95         """
96         print "fin de la ventana de selección"
97         print "Parseando documento . . ."
98         xml_analizador = xml.sax.make_parser()
99         xml_analizador.setContentHandler(ContentHandler())
100         try:
101             xml_analizador.parse(fichero)
102             print "El fichero XML %s está bien formado." %
                fichero
103         except Exception, err:
104             if fichero == None:
105                 msg = "No se ha seleccionado ningún fichero.
                    ¿Desea salir de la aplicación?"
```

```

106         title = "¿Desea salir?"
107         error = ynbox(msg=msg, title=title)
108         if error == 1:
109             #os.system('clear')
110             print "deteniendo agentes"
111             a = dfma("dfma@127.0.0.1", "secret")
112             b = representacion("representacion@127
113                               .0.0.1", "secret")
114             a.stop()
115             b.stop()
116             sys.exit(0)
117
118         else:
119             self.ventana_show()
120
121     else:
122         msg = "Error El fichero " + str(fichero) + "
123             no tiene" \
124             " formato XML-STEP. Por favor elija otro
125             fichero"
126         title = "Error de selección"
127         error = msgbox(msg=msg, title=title,
128                       ok_button="Aceptar", image=None, root=None
129                       )
130         if error == "Aceptar":
131             self.ventana_show()
132
133     a = dfma("dfma@127.0.0.1", "secret")
134     #a.setDebugToScreen()
135     a.start()
136
137     b = representacion("representacion@127.0.0.1", "
138                       secret")
139     print "inicia representación"
140     #b.setDebugToScreen()
141     b.start()
142
143     alive = True
144     while alive:
145         try:
146             time.sleep(1)
147             # Si detecto que el agente se ha parado,
148             salimos del bucle infinito

```

```

141         if not a.isAlive():
142             alive = False
143         except KeyboardInterrupt:
144             alive=False
145             a.stop()
146             b.stop()
147     sys.exit(0)
148
149 class dfma(spade.Agent.Agent):
150     """
151     Este agente genera la información que obtiene del XML y
152     solicita la
153     información necesaria para obtener el índice DFMA
154     """
155     class indice_dfma(spade.Behaviour.OneShotBehaviour):
156         def onStart(self):
157             print "start agente indice dfma"
158             self.partes()
159             """
160             el agente da el número de piezas que contiene el
161             producto analizado
162             """
163             self.ventana_numero()
164
165         def ventana_numero(self):
166             msg = "\nEl producto analizado consta de: " + str(
167                 numero_filas) + " componentes\n"
168             title = "Número de componentes"
169             partes = ccbox(msg=msg, title=title)
170             if partes == 1:
171                 self.nombres()
172                 self.ventana_nombre()
173             else:
174                 inicio.ventana_seleccion()
175
176         def ventana_nombre(self):
177             """
178             el agente da el nombre de caa uno de las piezas del
179             producto
180             """
181             Texto = ""

```

```

179         for i in range(numero_filas):
180             Texto = Texto + "Componente " + str(i+1) + ":
                " + matriz [i][1] + "\n\n"
181 msg = "\nLos nombres de los componentes son: \n\n" +
        Texto
182 title = "Nombre de cada componente"
183 partes = ccbox(msg=msg, title=title)
184 if partes == 1:
185     global Z
186     Z = 0
187     self.ventana_critica(Z)
188 else:
189     self.ventana_numero()
190
191 def ventana_critica(self, Z):
192     """
193     según la selección se le da valor critico a la pieza
        o no
194     """
195     global nombre_pieza
196     nombre_pieza = matriz[Z][1]
197     print("ventana critica num " + str(Z+1))
198     msg = "\nLa pieza " + str(nombre_pieza) + " se define
        por:\n"
199     title = "Criticidad de la pieza " + str(Z+1)
200     choices = ["1 - Tiene movimiento respecto a otras
        piezas"\
201 , "2 - Tiene que ser de un material distinto al resto
        de piezas"\
202 , "3 - Debe ser separada para ensamblar otras piezas"
        \
203 , "4 - Ninguna de las anteriores"]
204     choice = choicebox(msg, title, choices)
205     if choice == choices[3]:
206         matriz[Z][2] = 0
207     elif choice == None:
208         self.ventana_nombre()
209     else:
210         matriz[Z][2] = 1
211     print("pieza critica - " + str(nombre_pieza) + "
        valor: " + str(matriz[Z][2]))
212     self.imprime_matriz()

```



```

213         Z += 1
214         print "_____"
215         if Z == numero_filas:
216             Z = 0
217             self.ventana_manipulacion(Z)
218         else:
219             self.ventana_critica(Z)
220
221     def ventana_manipulacion(self, Z):
222         """
223         evalúa el índice de manipulación de la pieza según
224         las tablas
225         de Boothroyd
226         """
227         global nombre_pieza
228         nombre_pieza = matriz[Z][1]
229         print("ventana manipulacion num " + str(Z+1))
230         pic = "/home/z/Dropbox/PROYECTO/Programacion/Python/PROGRAMA/forma.gif"
231         choice = ['A', 'B', 'C', 'D', 'E', 'F']
232         msg = "\nCuál de las siguientes formas se asemeja más"
233         a la pieza " + str(nombre_pieza) + ":\n"
234         title = "Manejabilidad de la pieza " + str(Z+1)
235         reply = buttonbox(msg = msg, title = title, image =
236             pic, choices = choice)
237         """
238         esta ventana da los valores de simetría
239         """
240         if reply == None:
241             self.ventana_critica()
242         else:
243             if reply == "A":
244                 matriz[Z][3] = "0"
245                 matriz[Z][4] = "0"
246                 print "pulsado boton A"
247             elif reply == "B":
248                 matriz[Z][3] = "180"
249                 matriz[Z][4] = "0"
250                 print "pulsado boton B"
251             elif reply == "C":
252                 matriz[Z][3] = "180"
253                 matriz[Z][4] = "90"

```

```
251         print "pulsado boton C"
252     elif reply == "D":
253         matriz[Z][3] = "90"
254         matriz[Z][4] = "180"
255         print "pulsado boton D"
256     elif reply == "E":
257         matriz[Z][3] = "360"
258         matriz[Z][4] = "0"
259         print "pulsado boton E"
260     else:
261         matriz[Z][3] = "360"
262         matriz[Z][4] = "360"
263         print "pulsado boton F"
264
265     alfa = int(matriz[Z][3])
266     beta = int(matriz[Z][4])
267     print alfa
268     print beta
269     sumaangulo = alfa + beta
270     print sumaangulo
271
272     print("pieza manipulacion - " + str(nombre_pieza) + "
        Alfa: " + str(alfa) + " Beta: " + str(beta))
273     self.imprime_matriz()
274
275     msg = "\nLa pieza " + str(nombre_pieza) + " se define
        por:\n"
276     title = "Manipulación de la pieza " + str(Z+1)
277     choices = ["1A - Puede facilmente manipularse con 1
        sola mano"\
278     , "1B - Se manipula con 1 mano con dificultades de
        manipulacion"\
279     , "2 - Es necesario 1 mano y un elemento de sujecion"
        "\
280     , "3A - Se necesitan ambas manos para su montaje
        siendo sencillo"\
281     , "3B - Se necesitan ambas manos pero tiene
        dificultades en el manejo"\
282     , "4 - Es necesario de ambas manos y es de grandes
        dimensiones"]
283     choice = choicebox(msg, title, choices)
284     if choice == None:
```

```

285         Z = 0
286         self.ventana_critica(Z)
287     else:
288         matriz[Z][5] = choices.index(choice)
289         print("manipulacion de - " + str(nombre_pieza) +
290               " valor: " + str(matriz[Z][2]))
291         self.imprime_matriz()
292         self.ventana_tamanos(Z)
293     for n in range(len(fieldNames)):
294         if fieldValues[n] == "":
295             msgbox("por vuelva a introducir valores
296                   correctos para la pieza " + str(
297                     nombre_pieza))
298             self.ventana_tamanos(Z)
299             #ATENCIÓN TENGO QUE PROHIBIR QUE SE
300             #INTRODUZCA TEXTO
301             #CONVERTIR COMAS 0,2 EN PUNTOS 0.6
302             """
303             ahora se calcula cual de los tres números es el menor
304             y cuál es
305             el mediano. El pequeño se utiliza para el grosor
306             """
307             a = fieldValues[0]
308             b = fieldValues[1]
309             c = fieldValues[2]
310
311             if a == min(a,b): # a candidato a ser minimo, entre a
312                             y c
313                 if a == min(a,c):
314                     pequeno = a
315                     mediano = min(b,c)
316                 else:
317                     pequeno = c
318                     mediano = a
319             else: # b candidato a ser minimo, entre b y c
320                 if b == min(b,c):
321                     pequeno = b
322                     mediano = min(a,c)
323                 else:
324                     pequeno = c
325                     mediano = b

```

```
321         print fieldValues
322         pequeno = float(pequeno)
323         mediano = float(mediano)
324         print "pequeño: " + str(pequeno)
325         print "mediano: " + str(mediano)
326
327     if matriz[Z][5] == 0:
328
329         if pequeno <= 2:
330             if mediano <= 6:
331                 matriz[Z][6] = "4"
332                 if sumaangulo < 360:
333                     matriz[Z][9] = "0"
334                     matriz[Z][8] = "2.18"
335                 elif 360 <= sumaangulo < 540:
336                     matriz[Z][9] = "1"
337                     matriz[Z][8] = "2.55"
338                 elif 540 <= sumaangulo < 720:
339                     matriz[Z][9] = "2"
340                     matriz[Z][8] = "2.85"
341                 else:
342                     matriz[Z][9] = "3"
343                     matriz[Z][8] = "3"
344             else:
345                 matriz[Z][6] = "3"
346                 if sumaangulo < 360:
347                     matriz[Z][9] = "0"
348                     matriz[Z][8] = "1.69"
349                 elif 360 <= sumaangulo < 540:
350                     matriz[Z][9] = "1"
351                     matriz[Z][8] = "2.06"
352                 elif 540 <= sumaangulo < 720:
353                     matriz[Z][9] = "2"
354                     matriz[Z][8] = "2.36"
355                 else:
356                     matriz[Z][9] = "3"
357                     matriz[Z][8] = "2.51"
358         else:
359             if mediano < 6:
360                 matriz[Z][6] = "2"
361                 if sumaangulo < 360:
362                     matriz[Z][9] = "0"
```

```
363         matriz[Z][8] = "1.88"
364     elif 360 <= sumaangulo < 540:
365         matriz[Z][9] = "1"
366         matriz[Z][8] = "2.25"
367     elif 540 <= sumaangulo < 720:
368         matriz[Z][9] = "2"
369         matriz[Z][8] = "2.55"
370     else:
371         matriz[Z][9] = "3"
372         matriz[Z][8] = "2.7"
373 elif 6 < mediano <= 15:
374     matriz[Z][6] = "1"
375     if sumaangulo < 360:
376         matriz[Z][9] = "0"
377         matriz[Z][8] = "1.43"
378     elif 360 <= sumaangulo < 540:
379         matriz[Z][9] = "1"
380         matriz[Z][8] = "1.8"
381     elif 540 <= sumaangulo < 720:
382         matriz[Z][9] = "2"
383         matriz[Z][8] = "2.1"
384     else:
385         matriz[Z][9] = "3"
386         matriz[Z][8] = "2.25"
387 else:
388     matriz[Z][6] = "0"
389     if sumaangulo < 360:
390         matriz[Z][9] = "0"
391         matriz[Z][8] = "1.13"
392     elif 360 <= sumaangulo < 540:
393         matriz[Z][9] = "1"
394         matriz[Z][8] = "1.5"
395     elif 540 <= sumaangulo < 720:
396         matriz[Z][9] = "2"
397         matriz[Z][8] = "1.8"
398     else:
399         matriz[Z][9] = "3"
400         matriz[Z][8] = "1.95"
401
402 elif matriz[Z][5] == 1:
403     if pequeno <= 2:
404         if mediano <= 6:
```

```
405         matriz[Z][6] = "9"
406     if sumaangulo < 360:
407         matriz[Z][9] = "0"
408         matriz[Z][8] = "2.98"
409     elif 360 <= sumaangulo < 540:
410         matriz[Z][9] = "1"
411         matriz[Z][8] = "3.38"
412     elif 540 <= sumaangulo < 720:
413         matriz[Z][9] = "2"
414         matriz[Z][8] = "3.7"
415     else:
416         matriz[Z][9] = "3"
417         matriz[Z][8] = "4"
418 else:
419     matriz[Z][6] = "8"
420     if sumaangulo < 360:
421         matriz[Z][9] = "0"
422         matriz[Z][8] = "2.45"
423     elif 360 <= sumaangulo < 540:
424         matriz[Z][9] = "1"
425         matriz[Z][8] = "3"
426     elif 540 <= sumaangulo < 720:
427         matriz[Z][9] = "2"
428         matriz[Z][8] = "3.18"
429     else:
430         matriz[Z][9] = "3"
431         matriz[Z][8] = "3.34"
432 else:
433     if mediano < 6:
434         matriz[Z][6] = "7"
435         if sumaangulo < 360:
436             matriz[Z][9] = "0"
437             matriz[Z][8] = "2.65"
438         elif 360 <= sumaangulo < 540:
439             matriz[Z][9] = "1"
440             matriz[Z][8] = "3.06"
441         elif 540 <= sumaangulo < 720:
442             matriz[Z][9] = "2"
443             matriz[Z][8] = "3.38"
444         else:
445             matriz[Z][9] = "3"
446             matriz[Z][8] = "3.55"
```

```
447         elif 6 < mediano <= 15:
448             matriz[Z][6] = "6"
449             if sumaangulo < 360:
450                 matriz[Z][9] = "0"
451                 matriz[Z][8] = "2.17"
452             elif 360 <= sumaangulo < 540:
453                 matriz[Z][9] = "1"
454                 matriz[Z][8] = "2.57"
455             elif 540 <= sumaangulo < 720:
456                 matriz[Z][9] = "2"
457                 matriz[Z][8] = "2.9"
458             else:
459                 matriz[Z][9] = "3"
460                 matriz[Z][8] = "3.06"
461         else:
462             matriz[Z][6] = "5"
463             if sumaangulo < 360:
464                 matriz[Z][9] = "0"
465                 matriz[Z][8] = "1.84"
466             elif 360 <= sumaangulo < 540:
467                 matriz[Z][9] = "1"
468                 matriz[Z][8] = "2.25"
469             elif 540 <= sumaangulo < 720:
470                 matriz[Z][9] = "2"
471                 matriz[Z][8] = "2.57"
472             else:
473                 matriz[Z][9] = "3"
474                 matriz[Z][8] = "2.73"
475
476     elif matriz[Z][5] == 2:
477         msg = "\nEn la pieza " + str(nombre_pieza) + "
            además de ser necesario utilizar una mano y un
            elemento de sujeción:\n"
478         title = "Manipulacion con 1 mano y un elemento de
            sujecion de la pieza " + str(Z+1)
479         choices2 = ["1 - No necesita util y es facil su
            manipulacion" \
480 , "2 - No necesita utiles pero su manipulacion es
            dificil" \
481 , "3 - Necesita elementos de aumento visual y es
            facil su manipulacion" \
```

```
482         , "4 – Necesita elementos de aumento visual y es
          difícil su manipulacion" \
483         , "5 – Es necesario el uso de utensilios de
          precision como pinzas" \
484         , "6 – Es necesario uso de herramientas
          especificas para su manipulacion" ]
485 choice2 = choicebox(msg, title , choices2)
486
487 if choice2 == None:
488     Z = 0
489     self.ventana_critica(Z)
490 else:
491     matriz[Z][7] = choices2.index(choice2)
492
493 if matriz[Z][7] == 0:
494     if pequeno > 0.25:
495         matriz[Z][6] = "0"
496         if alfa <= 180:
497             if 0 <= beta <= 180:
498                 matriz[Z][9] = "4"
499                 matriz[Z][8] = "3.6"
500             #elif beta == 360:
501             else:
502                 matriz[Z][9] = "5"
503                 matriz[Z][8] = "4"
504         else:
505             if 0 <= beta <= 180:
506                 matriz[Z][9] = "6"
507                 matriz[Z][8] = "4.8"
508             #elif beta == 360:
509             else:
510                 matriz[Z][9] = "7"
511                 matriz[Z][8] = "5.1"
512     else:
513         matriz[Z][6] = "1"
514         if alfa <= 180:
515             if 0 <= beta <= 180:
516                 matriz[Z][9] = "4"
517                 matriz[Z][8] = "6.85"
518             #elif beta == 360:
519             else:
520                 matriz[Z][9] = "5"
```



```
521             matriz[Z][8] = "7.25"
522         else :
523             if 0 <= beta <= 180:
524                 matriz[Z][9] = "6"
525                 matriz[Z][8] = "8.05"
526             #elif beta == 360:
527             else :
528                 matriz[Z][9] = "7"
529                 matriz[Z][8] = "8.35"
530     elif matriz[Z][7] == 1:
531         if pequeno > 0.25:
532             matriz[Z][6] = "2"
533             if alfa <= 180:
534                 if 0 <= beta <= 180:
535                     matriz[Z][9] = "4"
536                     matriz[Z][8] = "4.35"
537                 #elif beta == 360:
538                 else :
539                     matriz[Z][9] = "5"
540                     matriz[Z][8] = "4.75"
541             else :
542                 if 0 <= beta <= 180:
543                     matriz[Z][9] = "6"
544                     matriz[Z][8] = "5.55"
545                 #elif beta == 360:
546                 else :
547                     matriz[Z][9] = "7"
548                     matriz[Z][8] = "5.85"
549         else :
550             matriz[Z][6] = "3"
551             if alfa <= 180:
552                 if 0 <= beta <= 180:
553                     matriz[Z][9] = "4"
554                     matriz[Z][8] = "7.6"
555                 #elif beta == 360:
556                 else :
557                     matriz[Z][9] = "5"
558                     matriz[Z][8] = "8"
559             else :
560                 if 0 <= beta <= 180:
561                     matriz[Z][9] = "6"
562                     matriz[Z][8] = "8.8"
```

```
563         #elif beta == 360:
564         else :
565             matriz[Z][9] = "7"
566             matriz[Z][8] = "9.1"
567     elif matriz[Z][7] == 2:
568         if pequeno > 0.25:
569             matriz[Z][6] = "4"
570         if alfa <= 180:
571             if 0 <= beta <= 180:
572                 matriz[Z][9] = "4"
573                 matriz[Z][8] = "5.6"
574             #elif beta == 360:
575             else :
576                 matriz[Z][9] = "5"
577                 matriz[Z][8] = "6"
578         else :
579             if 0 <= beta <= 180:
580                 matriz[Z][9] = "6"
581                 matriz[Z][8] = "6.8"
582             #elif beta == 360:
583             else :
584                 matriz[Z][9] = "7"
585                 matriz[Z][8] = "7.1"
586     else :
587         matriz[Z][6] = "5"
588         if alfa <= 180:
589             if 0 <= beta <= 180:
590                 matriz[Z][9] = "4"
591                 matriz[Z][8] = "8.35"
592             #elif beta == 360:
593             else :
594                 matriz[Z][9] = "5"
595                 matriz[Z][8] = "8.75"
596         else :
597             if 0 <= beta <= 180:
598                 matriz[Z][9] = "6"
599                 matriz[Z][8] = "9.55"
600             #elif beta == 360:
601             else :
602                 matriz[Z][9] = "7"
603                 matriz[Z][8] = "9.55"
604     elif matriz[Z][7] == 3:
```

```
605         if pequeno > 0.25:
606             matriz[Z][6] = "6"
607             if alfa <= 180:
608                 if 0 <= beta <= 180:
609                     matriz[Z][9] = "4"
610                     matriz[Z][8] = "6.35"
611                 elif beta == 360:
612                     else:
613                         matriz[Z][9] = "5"
614                         matriz[Z][8] = "6.75"
615             else:
616                 if 0 <= beta <= 180:
617                     matriz[Z][9] = "6"
618                     matriz[Z][8] = "7.55"
619                 elif beta == 360:
620                     else:
621                         matriz[Z][9] = "7"
622                         matriz[Z][8] = "7.85"
623         else:
624             matriz[Z][6] = "7"
625             if alfa <= 180:
626                 if 0 <= beta <= 180:
627                     matriz[Z][9] = "4"
628                     matriz[Z][8] = "8.6"
629                 elif beta == 360:
630                     matriz[Z][9] = "5"
631                     matriz[Z][8] = "9"
632             else:
633                 if 0 <= beta <= 180:
634                     matriz[Z][9] = "6"
635                     matriz[Z][8] = "9.8"
636                 elif beta == 360:
637                     else:
638                         matriz[Z][9] = "7"
639                         matriz[Z][8] = "10.1"
640         elif matriz[Z][7] == 4:
641             matriz[Z][6] = "8"
642             if alfa <= 180:
643                 if 0 <= beta <= 180:
644                     matriz[Z][9] = "4"
645                     matriz[Z][8] = "7"
646                 elif beta == 360:
```

```

647         else :
648             matriz[Z][9] = "5"
649             matriz[Z][8] = "8"
650     else :
651         if 0 <= beta <= 180:
652             matriz[Z][9] = "6"
653             matriz[Z][8] = "8"
654         #elif beta == 360:
655         else :
656             matriz[Z][9] = "7"
657             matriz[Z][8] = "9"
658     else :
659         matriz[Z][6] = "9"
660         if alfa <= 180:
661             if 0 <= beta <= 180:
662                 matriz[Z][9] = "4"
663                 matriz[Z][8] = "7"
664             #elif beta == 360:
665             else :
666                 matriz[Z][9] = "5"
667                 matriz[Z][8] = "8"
668         else :
669             if 0 <= beta <= 180:
670                 matriz[Z][9] = "6"
671                 matriz[Z][8] = "9"
672             #elif beta == 360:
673             else :
674                 matriz[Z][9] = "7"
675                 matriz[Z][8] = "10"
676
677     elif matriz[Z][5] == 3:
678         if alfa <= 180:
679             if mediano > 15:
680                 matriz[Z][6] = "0"
681                 matriz[Z][9] = "8"
682                 matriz[Z][8] = "4.1"
683             elif 6 < mediano <= 15:
684                 matriz[Z][6] = "1"
685                 matriz[Z][9] = "8"
686                 matriz[Z][8] = "4.5"
687         else :
688             matriz[Z][6] = "2"

```

```
689             matriz[Z][9] = "8"
690             matriz[Z][8] = "5.1"
691         else :
692             if mediano > 6:
693                 matriz[Z][6] = "3"
694                 matriz[Z][9] = "8"
695                 matriz[Z][8] = "5.6"
696             else :
697                 matriz[Z][6] = "4"
698                 matriz[Z][9] = "8"
699                 matriz[Z][8] = "6.75"
700
701     elif matriz[Z][5] == 4:
702         if alfa <= 180:
703             if mediano > 15:
704                 matriz[Z][6] = "5"
705                 matriz[Z][9] = "8"
706                 matriz[Z][8] = "5"
707             elif 6 < mediano <= 15:
708                 matriz[Z][6] = "6"
709                 matriz[Z][9] = "8"
710                 matriz[Z][8] = "5.25"
711             else :
712                 matriz[Z][6] = "7"
713                 matriz[Z][9] = "8"
714                 matriz[Z][8] = "5.85"
715         else :
716             if mediano > 6:
717                 matriz[Z][6] = "8"
718                 matriz[Z][9] = "8"
719                 matriz[Z][8] = "6.35"
720             else :
721                 matriz[Z][6] = "9"
722                 matriz[Z][9] = "8"
723                 matriz[Z][8] = "7"
724
725     elif matriz[Z][5] == 5:
726         msg = "\nEn la pieza " + str(nombre_pieza) + " se
              usan ambas manos y es de gran tamaño y:\n"
727         title = "Manipulacion con 2 manos y gran tamaño
              en pieza " + str(Z+1)
```

```
728         choices2 = ["1 – Pesa menos de 10Lb y siendo de
                        facil manejo"\
729         , "2 – Pesa menos de 10Lb pero es de dificil
                        manejo"\
730         , "3 – Pesa mas de 10Lb pero es de facil manejo"\
731         , "4 – Pesa mas de 10Lb y es de dificil manejo"\
732         , "5 – Las partes se enredan, se entrelian o son
                        flexibles" \
733         , "6 – Se necesitan dos personas o ayuda mecanica
                        " ]
734     choice2 = choicebox(msg, title , choices2)
735
736     if choice2 == None:
737         Z = 0
738         self.ventana_critica(Z)
739     else :
740         matriz[Z][7] = choices2.index(choice2)
741
742     if matriz[Z][7] == 0:
743         if alfa <= 180:
744             matriz[Z][6] = "0"
745             matriz[Z][9] = "9"
746             matriz[Z][8] = "2"
747         else :
748             matriz[Z][6] = "1"
749             matriz[Z][9] = "9"
750             matriz[Z][8] = "3"
751     elif matriz[Z][7] == 1:
752         if alfa <= 180:
753             matriz[Z][6] = "2"
754             matriz[Z][9] = "9"
755             matriz[Z][8] = "2"
756         else :
757             matriz[Z][6] = "3"
758             matriz[Z][9] = "9"
759             matriz[Z][8] = "3"
760     elif matriz[Z][7] == 2:
761         if alfa <= 180:
762             matriz[Z][6] = "4"
763             matriz[Z][9] = "9"
764             matriz[Z][8] = "3"
765     else :
```

```

766             matriz[Z][6] = "5"
767             matriz[Z][9] = "9"
768             matriz[Z][8] = "4"
769         elif matriz[Z][7] == 3:
770             if alfa <= 180:
771                 matriz[Z][6] = "6"
772                 matriz[Z][9] = "9"
773                 matriz[Z][8] = "4"
774             else:
775                 matriz[Z][6] = "7"
776                 matriz[Z][9] = "9"
777                 matriz[Z][8] = "5"
778         elif matriz[Z][7] == "4":
779             matriz[Z][6] = "8"
780             matriz[Z][9] = "9"
781             matriz[Z][8] = "7"
782         else:
783             matriz[Z][6] = "9"
784             matriz[Z][9] = "9"
785             matriz[Z][8] = "9"
786
787     self.imprime_matriz()
788
789
790     print "_____"
791
792     self.ventana_ensamble(Z)
793
794
795     def ventana_ensamble(self, Z):
796         """
797         esta función evalúa el código de insercion según
798         Boothroyd y le
799         asigna un tiempo de inserción
800         """
801         msg = "\nRespecto a otra pieza a ensamblar, la pieza "
802             + matriz[Z][1] + ":\n"
803         title = "Insercion de piezas"
804         choices3 = ["1 - Son procesos completamente distintos
            "\n
            , "2 - Se insertan de forma inmediata"\n
            , "3 - Se ensamblan mediante un proceso"]

```

```
805         choice3 = choicebox(msg, title, choices3)
806         if choice3 == None:
807             Z = 0
808             self.ventana_manipulacion(Z)
809         else:
810             if choice3[0] == "1":
811                 msg = "\n Seleccione la facilidad de inserción
812                 :\n"
813                 title = "Inserción Inmediata – Inserción"
814                 choices4 = ["1 – Fácil localización"\
815                 , "2 – Visión o acceso difícil"\
816                 , "3 – Visión y acceso difícil"\
817                 ]
818                 choice4 = choicebox(msg, title, choices4)
819                 matriz[Z][10] = int(choice4[0]) - 1
820                 if choice4 == None:
821                     Z = 0
822                     self.ventana_manipulacion(Z)
823                 else:
824                     msg = "\n ¿Después del ensamblado necesita
825                     de sujeción para mantener la posición
826                     y la orientación?\n"
827                     title = "Procesos distintos"
828                     choices5 = ["1 – No, y es de fácil
829                     alineación"\
830                     , "2 – No, pero su alineación es compleja
831                     "\
832                     , "3 – Sí, pero es de fácil alineación"\
833                     , "4 – Sí y su alineación es difícil"\
834                     ]
835                     choice5 = choicebox(msg, title, choices5)
836                     if choice5 == None:
837                         Z = 0
838                         self.ventana_manipulacion(Z)
839                     else:
840                         msg = "¿Es necesario el uso de fuerza
841                         porque la pieza ofrece
842                         resistencia?"
843                         title = "Resistencia de la Pieza"
844                         error = ynbox(msg=msg, title=title)
845                         if error == 0:
846                             if choice5[0] == 1:
```



```
840         matriz[Z][11] = 0
841     elif choice5[0] == 2:
842         matriz[Z][11] = 2
843     elif choice5[0] == 3:
844         matriz[Z][11] = 6
845     else:
846         matriz[Z][11] = 8
847     else:
848         if choice5[0] == 1:
849             matriz[Z][11] = 1
850         elif choice5[0] == 2:
851             matriz[Z][11] = 3
852         elif choice5[0] == 3:
853             matriz[Z][11] = 7
854         else:
855             matriz[Z][11] = 9
856
857 elif choice3[0] == "2":
858     msg = "\n Seleccione la facilidad de inserción
859         :\n"
860     title = "Inserción Inmediata – Inserción"
861     choices4 = ["1 – Fácil localización"\
862         , "2 – Visión o acceso dificil"\
863         , "3 – Visión y acceso dificil"\
864         ]
865     choice4 = choicebox(msg, title , choices4)
866     matriz[Z][10] = int(choice4[0]) + 2
867     if choice4 == None:
868         Z = 0
869         self.ventana_manipulacion(Z)
870     else:
871         msg = "\n Seleccione la facilidad de
872             alineación:\n"
873         title = "Inserción Inmediata – Alineación"
874
875         choices5 = ["01 – Apriete con fácil
876             alineación"\
877             , "02 – Apriete con alineación dificil"\
878             , "03 – Inserción por doblado fácil"\
879             , "04 – Inserción por doblado dificil"\
880             , "05 – Inserción por doblado dicil con
881             resistencia"\
```

```

877         , "06 – Remachado fácil"\
878         , "07 – Remachado difícil"\
879         , "08 – Remachado difícil con resistencia
            "\
880         , "09 – Roscado con fácil alineación"\
881         , "10 – Roscado de difícil alineación"\
882     ]
883     choice5 = choicebox(msg, title , choices5)
884     if choice5 == None:
885         Z = 0
886         self.ventana_manipulacion(Z)
887     else:
888         matriz[Z][11] = int(choice5[0:2]) - 1
889
890 elif choice3[0] == "3":
891     msg = "\n Elija uno de estos procesos:\n"
892     title = "Operación distinta"
893     choices4 = ["01 – Doblado"\
894         , "02 – Remachado"\
895         , "03 – Roscado"\
896         , "04 – Grandes deformaciones"\
897         , "05 – Unión por fricción"\
898         , "06 – Soldadura eléctrica"\
899         , "07 – Soldadura por arco"\
900         , "08 – Proceso químico"\
901         , "09 – Orientación/Ajuste de piezas sin
            sujeción"\
902         , "10 – Otros procesos (Ej: Inserción de
            líquidos...) "\
903     ]
904     choice4 = choicebox(msg, title , choices4)
905     if choice4 == None:
906         Z = 0
907         self.ventana_manipulacion(Z)
908     else:
909         matriz[Z][10] = 9
910         print choice4[0:2]
911         matriz[Z][11] = int(choice4[0:2]) - 1
912
913     matriz[Z][12] = str(matriz[Z][10]) + str(matriz[Z]
914         ][11])
915     """

```

```
915         matriz 12 es el codigo de inserción el cual según  
916         las tablas  
917         tiene el valor de tiempo asociado a matriz 13  
918         """  
919         if matriz[Z][12] == "00":  
920             matriz[Z][13] = "1.5 "  
921         elif matriz[Z][12] == "01":  
922             matriz[Z][13] = "2.5 "  
923         elif matriz[Z][12] == "02":  
924             matriz[Z][13] = "2.5 "  
925         elif matriz[Z][12] == "03":  
926             matriz[Z][13] = "3.5 "  
927         elif matriz[Z][12] == "06":  
928             matriz[Z][13] = "5.5 "  
929         elif matriz[Z][12] == "07":  
930             matriz[Z][13] = "6.5 "  
931         elif matriz[Z][12] == "08":  
932             matriz[Z][13] = "6.5 "  
933         elif matriz[Z][12] == "09":  
934             matriz[Z][13] = "7.5 "  
935         elif matriz[Z][12] == "10":  
936             matriz[Z][13] = "4 "  
937         elif matriz[Z][12] == "11":  
938             matriz[Z][13] = "5 "  
939         elif matriz[Z][12] == "12":  
940             matriz[Z][13] = "5 "  
941         elif matriz[Z][12] == "13":  
942             matriz[Z][13] = "6 "  
943         elif matriz[Z][12] == "16":  
944             matriz[Z][13] = "8 "  
945         elif matriz[Z][12] == "17":  
946             matriz[Z][13] = "9 "  
947         elif matriz[Z][12] == "18":  
948             matriz[Z][13] = "9 "  
949         elif matriz[Z][12] == "19":  
950             matriz[Z][13] = "10 "  
951         elif matriz[Z][12] == "20":  
952             matriz[Z][13] = "5.5 "  
953         elif matriz[Z][12] == "21":  
954             matriz[Z][13] = "6.5 "  
955         elif matriz[Z][12] == "22":  
956             matriz[Z][13] = "6.5 "
```

```
956         elif matriz[Z][12] == "23":
957             matriz[Z][13] = "7.5 "
958         elif matriz[Z][12] == "26":
959             matriz[Z][13] = "9.5 "
960         elif matriz[Z][12] == "27":
961             matriz[Z][13] = "10.5 "
962         elif matriz[Z][12] == "28":
963             matriz[Z][13] = "10.5 "
964         elif matriz[Z][12] == "29":
965             matriz[Z][13] = "11.5 "
966         elif matriz[Z][12] == "30":
967             matriz[Z][13] = "2 "
968         elif matriz[Z][12] == "31":
969             matriz[Z][13] = "5 "
970         elif matriz[Z][12] == "32":
971             matriz[Z][13] = "4 "
972         elif matriz[Z][12] == "33":
973             matriz[Z][13] = "5 "
974         elif matriz[Z][12] == "34":
975             matriz[Z][13] = "6 "
976         elif matriz[Z][12] == "35":
977             matriz[Z][13] = "7 "
978         elif matriz[Z][12] == "36":
979             matriz[Z][13] = "8 "
980         elif matriz[Z][12] == "37":
981             matriz[Z][13] = "9 "
982         elif matriz[Z][12] == "38":
983             matriz[Z][13] = "6 "
984         elif matriz[Z][12] == "39":
985             matriz[Z][13] = "8 "
986         elif matriz[Z][12] == "40":
987             matriz[Z][13] = "4.5 "
988         elif matriz[Z][12] == "41":
989             matriz[Z][13] = "7.5 "
990         elif matriz[Z][12] == "42":
991             matriz[Z][13] = "6.5 "
992         elif matriz[Z][12] == "43":
993             matriz[Z][13] = "7.5 "
994         elif matriz[Z][12] == "44":
995             matriz[Z][13] = "8.5 "
996         elif matriz[Z][12] == "45":
997             matriz[Z][13] = "9.5 "
```

```
998         elif matriz[Z][12] == "46":
999             matriz[Z][13] = "10.5"
1000         elif matriz[Z][12] == "47":
1001             matriz[Z][13] = "11.5"
1002         elif matriz[Z][12] == "48":
1003             matriz[Z][13] = "8.5"
1004         elif matriz[Z][12] == "49":
1005             matriz[Z][13] = "10.5"
1006         elif matriz[Z][12] == "50":
1007             matriz[Z][13] = "6"
1008         elif matriz[Z][12] == "51":
1009             matriz[Z][13] = "9"
1010         elif matriz[Z][12] == "52":
1011             matriz[Z][13] = "8"
1012         elif matriz[Z][12] == "53":
1013             matriz[Z][13] = "9"
1014         elif matriz[Z][12] == "54":
1015             matriz[Z][13] = "10"
1016         elif matriz[Z][12] == "55":
1017             matriz[Z][13] = "11"
1018         elif matriz[Z][12] == "56":
1019             matriz[Z][13] = "12"
1020         elif matriz[Z][12] == "57":
1021             matriz[Z][13] = "13"
1022         elif matriz[Z][12] == "58":
1023             matriz[Z][13] = "10"
1024         elif matriz[Z][12] == "59":
1025             matriz[Z][13] = "12"
1026         elif matriz[Z][12] == "90":
1027             matriz[Z][13] = "4"
1028         elif matriz[Z][12] == "91":
1029             matriz[Z][13] = "7"
1030         elif matriz[Z][12] == "92":
1031             matriz[Z][13] = "5"
1032         elif matriz[Z][12] == "93":
1033             matriz[Z][13] = "3.5"
1034         elif matriz[Z][12] == "94":
1035             matriz[Z][13] = "7"
1036         elif matriz[Z][12] == "95":
1037             matriz[Z][13] = "8"
1038         elif matriz[Z][12] == "96":
1039             matriz[Z][13] = "12"
```

```

1040         elif matriz[Z][12] == "97":
1041             matriz[Z][13] = "12"
1042         elif matriz[Z][12] == "98":
1043             matriz[Z][13] = "9"
1044         elif matriz[Z][12] == "99":
1045             matriz[Z][13] = "12"
1046
1047         else:
1048             print("Tiempo inserción = 0")
1049             matriz[Z][13] = "0"
1050     Z += 1
1051     if Z == numero_filas:
1052         self.valor_indice()
1053     else:
1054         self.ventana_manipulacion(Z)
1055
1056     def valor_indice(self):
1057         """
1058         tras los cálculos de los valores se generan los
1059         valores de tiempo
1060         total, que es la suma de los tiempos de cada pieza
1061         """
1062         global tiempo_total
1063         tiempo_total = 0
1064         global minimas_partes
1065         minimas_partes = 0
1066         for i in range(numero_filas):
1067             tiempo_total = float(matriz[i][8]) + float(matriz
1068                 [i][13]) + tiempo_total
1069             minimas_partes = matriz[i][2] + minimas_partes
1070         print("_____")
1071         self.imprime_matriz()
1072         print("tiempo total: ")
1073         print(tiempo_total)
1074         print("minimas partes: ")
1075         print(minimas_partes)
1076         """
1077         el valor del indice es porcentaje 3 veces el numero
1078         de piezas
1079         críticas entre el tiempo total
1080         """

```

```

1078         valor_indice = (3 * minimas_partes / tiempo_total) *
1079             100
1080         global valor_indice_r
1081         valor_indice_r = "%.2f" % valor_indice
1082         print " valor indice dfma: "
1083         print valor_indice_r
1084
1085         choice = [ 'Exportar datos a Excel', 'Cerrar el
1086             programa' ]
1087         msg = """
1088             El índice DFMA de su producto es de :
1089
1090             """ + str(valor_indice_r) + """ %.
1091
1092             Ahora usted puede: """
1093         title = "Índice DFMA"
1094         reply = messagebox(msg = msg, title = title , image =
1095             None, choices = choice )
1096         if reply == None:
1097             Z = 0
1098             self.ventana_manipulacion(Z)
1099         else :
1100             if reply[0] == "E":
1101                 self.excel()
1102             else :
1103                 self.salida()
1104
1105         def salida(self):
1106             """
1107             esta funcion detiene los agentes y cierra la
1108             aplicación
1109             """
1110             print "adios"
1111             print "deteniendo agentes"
1112             a = dfma("dfma@127.0.0.1", "secret")
1113             b = representacion("representacion@127.0.0.1", "
1114                 secret")
1115             a.stop()
1116             b.stop()
1117             sys.exit(0)
1118
1119         def _process(self):

```

```

1115         print "Inicio evaluación documento . . ."
1116
1117     def onEnd(self):
1118         print "kill agente indice dfma"
1119
1120     #-----FUNCIONES UTILIZADAS-----#
1121     def partes(self):
1122         """
1123         esta funcion es la encargada de buscar cada parte que
1124         compone el
1125         producto y enumerarla
1126         """
1127         print ("listado de cada parte 'pieza' del producto")
1128         global vari_nodo
1129         vari_nodo = "Config_control_design:
1130                     Next_assembly_usage_occurrence" #LAST
1131                     MODIFICATION
1132         #vari_nodo = "Config_control_design:Product_category"
1133                     #buscar partes
1134         #vari_nodo = "Config_control_design:Product"
1135                     #buscar productos
1136
1137         self.buscar()
1138         global lista
1139         lista = range(len(valor_id))
1140         global numero_filas
1141         numero_filas = len(lista)
1142         global matriz
1143         """
1144         ahora se crea una matriz con N lineas según el numero
1145         de piezas
1146         y a cada pieza se le da una id igual al numero de
1147         linea (empieza
1148         por 0)
1149         """
1150         matriz = [None] * numero_filas
1151         for i in range(numero_filas):
1152             matriz[i] = [None] * numero_columnas
1153         for i in range(len(lista)):
1154             matriz[i][0] = i
1155
1156     def nombres(self):
1157         """

```



```

1150         esta funcion lista los nombres de cada pieza buscando
1151         en el
1152         documento xml la variable vari_nodo
1153         """
1154         print("Listado de nombres correspondientes a cada
1155               parte")
1156         global vari_nodo
1157         vari_nodo = "Config_control_design:
1158                   Next_assembly_usage_occurrence"
1159         self.buscar()
1160         global vari_atributo
1161         vari_atributo = "Name"
1162         self.obtener()
1163         for i in range(len(lista)):
1164             matriz[i][1] = valor_atributo[i]
1165
1166     def buscar(self):
1167         """
1168         esta función es la que busca en el documento los
1169         valores
1170         de vari_nodo. Se utilizan dos parseadores por la
1171         facilidad
1172         de busqueda de uno entre lineas y otro entre nodos
1173         """
1174         global archivo_leer
1175         archivo_leer = minidom.parse(fichero)
1176         global etree
1177         etree = ET.parse(fichero)
1178         global root
1179         root = etree.getroot()
1180         print("Busca los nodos y atributos previamente
1181               definidos")
1182         print("Búsqueda de etiqueta: " + vari_nodo)
1183         print(".....")
1184         global valor_id
1185         valor_id = []
1186         for x in range(len(archivo_leer.documentElement.
1187                           childNodes)):
1188             try:
1189                 if "id" in archivo_leer.getElementsByTagName(
1190                     vari_nodo)[x].attributes.keys():

```

```

1183         valor_id.append(archivo_leer.
                        getElementsByTagName(vari_nodo)[x].
                        attributes.get("id").value)

1184     except:
1185         break
1186     print "lista de valores encontrados: "
1187     print(valor_id)
1188     print "_____"
1189
1190 def obtener(self):
1191     """
1192     realiza iteraciones por cada numero de pieza para
1193     conseguir
1194     N numero de valores en la busqueda
1195     """
1196     for y in range(len(valor_id)):
1197         print "XXXXXXXXX ITERACION XXXXXXXXXXXXX"
1198         for child in root.findall("./*[@id='" + valor_id
1199             [y] + "'""):
1200             print "BUSQUEDA de ID=" + valor_id[y]
1201             print "BUSQUEDA DE ATRIBUTO/NODO=" +
1202                 vari_atributo
1203             #print "PADRE: " + str(root.findall(".*/*"))
1204             print "NODO: " + str(child)
1205             #print "HIJO: " + str(root.findall(".*/*"))
1206             #print "TAG: " + str(child.tag)
1207             #print "ATRIBUTOS: " + str(child.attrib)
1208             try:
1209                 valor_atributo.append(str(child.find(
1210                     vari_atributo).text))
1211                 print "VALOR ATRIBUTO: " + valor_atributo
1212                 [y]
1213             except:
1214                 print "No tiene característica " +
1215                     vari_atributo
1216         print "

```

```

1211
1212 def imprime_matriz(self):
1213     """

```

```

1214         Esta función imprime la matriz por pantalla para ir
1215         comprobando
1216         que los valores introducidos son correctos
1217         """
1218         for i in range(numero_filas):
1219             print matriz[i]
1220
1221     def ventana_tamanos(self , Z):
1222         """
1223         Función para ventana que solicita valores de tamaño
1224         de la pieza
1225         """
1226         msg = "Por favor introduzca los valores de tamaño de"
1227         la_pieza " + str(nombre_pieza) + " en mm,"
1228         separando los decimales con '.' : "
1229         title = "Cálculo tamaño pieza"
1230         global fieldNames
1231         fieldNames = [ 'Alto', 'Ancho', 'Largo' ]
1232         global fieldValues
1233         fieldValues = []
1234         fieldValues = multenterbox(msg, title , fieldNames ,
1235                                   fieldValues)
1236
1237     def excel(self):
1238         """
1239         Función que se encarga de trbajar con la hoja de
1240         calculo
1241         abre la plantilla introduce los datos y te solicita
1242         guardarlo en
1243         otra ubicación
1244         """
1245         rb = open_workbook('plantilla.xls',formatting_info=
1246                             True)
1247         wb = copy(rb)
1248         ws = wb.get_sheet(0)
1249         ws.write(2,3,fichero)#nombre fichero
1250         style0 = xlwt.easyxf('font: colour blue, bold on')
1251         style1 = xlwt.easyxf('alignment: horizontal center')
1252         style2 = xlwt.easyxf('font: colour black, bold on')
1253         for i in range(numero_filas):
1254             ws.write(i + 6,0,int(matriz[i][0]) + 1,style1)#id

```

```

1247         ws.write(i + 6,1,str(matriz[i][1]),style1) #
           nombre
1248         ws.write(i + 6,2,str(matriz[i][2]),style1)#
           critica
1249         ws.write(i + 6,3,str(matriz[i][9]) + str(matriz[i]
           [[6]),style1)#cod manip
1250         ws.write(i + 6,4,float(matriz[i][8]),style1)#
           tiempo man
1251         ws.write(i + 6,5,str(matriz[i][12]),style1)#cod
           inser
1252         ws.write(i + 6,6,matriz[i][13],style1)#tiempo
           inserc
1253         ws.write(i + 6,7,float(matriz[i][8]) + float(
           matriz[i][13]),style1)#tiempo unit pieza
1254         ws.write(i + 6,8,0.4 *(float(matriz[i][8]) +
           float(matriz[i][13])),style1)
1255         ws.write(i + 8,2,"NM = " + str(minimas_partes),
           style0)#suma criticas
1256         ws.write(i + 8,7,"TM = " + str(tiempo_total), style0)
           #suma tiempos
1257         ws.write(i + 8,8,"CM=" + str(0.4 * tiempo_total),
           style0)#suma criticas
1258         ws.write(i + 10,7,"Indice DFMA = " + valor_indice_r +
           " %", style2)
1259
1260         msg = "Guardar Tabla de estudio DFMA del producto"
1261         title = "Exportar datos excel"
1262         default = "DFMAsoft"
1263         filetypes = [ "*.xls", "*.xlsx", "*.odf", "*.ots", "
           Hojas de datos" ]
1264         excel = filesavebox(msg, title, default, filetypes)
1265         if filesavebox == None:
1266             self.valor_indice()
1267         elif filesavebox == "plantilla.xls":
1268             msgbox("Se ha elegido otro nombre para el fichero
           . No puede sustituir la plantilla")
1269             archivo_salida = "DFMAnoPLANTILLA.xls"
1270         else:
1271             archivo_salida = excel
1272         wb.save(excel)
1273         msgbox("El fichero excel ha sido generado. El
           programa se finalizará")

```

```

1274         self.salida()
1275
1276     def _setup(self):
1277         """
1278         Primer paso del agente. se dan valores a las variables
1279         globales
1280         """
1281         print "Inicio agente parseo"
1282         global numero_columnas
1283         numero_columnas = 14      #MODIFICAR CUANDO SE TENGAN MAS
1284                                   DATOS PARA CREAR MATRIZ MAS GRANDE
1285         global valor_atributo
1286         valor_atributo = []
1287         self.addBehaviour(self.indice_dfma())
1288
1289     class representacion(spade.Agent.Agent):
1290         """
1291         Se inicia el agente representación. Es el encargado de
1292         mostrar en 3D el
1293         fichero. Actualmente lee un fichero .wrl
1294         """
1295
1296     class representador(spade.Behaviour.OneShotBehaviour):
1297         '''
1298         Agente de representacion
1299         '''
1300         def onStart(self):
1301             print "start agente representación"
1302
1303         def _process(self):
1304             """
1305             Abre el fichero .stp.wrl en el programa view3dscene
1306             """
1307             print "Representando documento . . ."
1308             #NO CONSIGO INSTALAR OCCMODEL, GEOTOOLS, GLTOOLS
1309             #SI SE CONSIGUIESE EL CODIGO ES:
1310             ##
1311             try:
1312                 ##
1313                 objects = Tools.readSTEP(
1314                     fichero)
1315                 ##
1316                 viewer(objects)
1317             except:

```

```
1311         ##                                     print "agente de  
1312             representación no funciona "  
1313         nombrearchivo = fichero.split(".")  
1314         del nombrearchivo[-1]  
1315         nombrearchivo.append("wrl")  
1316         ficherowrl = ".".join(nombrearchivo)  
1317         print ficherowrl  
1318         orden = "view3dscene " + ficherowrl  
1319         try:  
1320             os.system(orden)  
1321         except:  
1322             print "no se encuentra fichero 3D wrl o su  
1323                 navegador WRL no es compatible"  
1324  
1325         def onEnd(self):  
1326             print "kill agente representacion"  
1327  
1328         def _setup(self):  
1329             self.addBehaviour(self.representador())  
1330  
1331         if __name__ == "__main__":  
1332             """  
1333             Se llama al módulo de inicio y comienza el programa  
1334             NO OLVIDAR EJECUTAR RUNSPADE.PY ANTES!!  
1335             """  
  
1336         a = inicio()
```