PORTABILIDAD DE LOS MODELOS DE TOMA DE DECISIONES CON HOJA DE CÁLCULO.

Autores:

Bernal García, Juan Jesús; juanjesus.bernal@upct.es

Martínez María Dolores, Soledad María; soledad.martinez@upct.es

Sánchez García, Juan Francisco; jf.sanchez@upct.es

Dpto. de Métodos Cuantitativos e Informáticos. Universidad Politécnica de Cartagena

Palabras clave: Modelos de toma de decisiones, portabilidad, hoja de cálculo.

Resumen:

En múltiples ocasiones sucede que al implantar un modelo de toma de decisiones realizado con hoja de cálculo, éste no funciona adecuadamente debido a la utilización de una aplicación diferente a aquella para la que fue creado. Si bien prácticamente cualquier aplicación de hoja de cálculo permite exportar sus ficheros al formato de otra aplicación diferente, no ocurre lo mismo con la programación realizada mediante "macros" o mediante lenguajes de programación orientada a objetos como VBA, llegando a surgir problemas incluso con distintas versiones de una misma aplicación.

En este trabajo se analizan los problemas de portabilidad que surgen con un modelo de simulación de demanda entre cuatro aplicaciones diferentes de hoja de cálculo (Excel, Quattro Pro, Lotus 1-2-3 y StarOffice), y se propone una solución basada en la utilización del formato estándar HTML y el lenguaje JavaScript.

1. Introducción

Actualmente, la hoja de cálculo más utilizada probablemente sea Excel en sus versiones 2000 ó 97, aunque eso no implica que no existan otras aplicaciones de este tipo y que pueden ser más o menos empleadas por las empresas. Entre estas aplicaciones "relegadas" a segundo término encontramos aplicaciones ya clásicas como Lotus 1-2-3 y Quattro Pro y una aplicación de reciente creación como StarOffice que está teniendo una mayor implantación día a día debido a su gratuidad o coste nulo, así

como a su compatibilidad con Excel 2000 y al hecho de que es la única aplicación de esta magnitud existente para Linux.

Todas las aplicaciones reseñadas pueden grabar sus ficheros en el formato de Excel, e incluso leer directamente los archivos de esta hoja de cálculo sin tener que realizar ningún tipo de conversión previa. El problema, sin embargo surge en el momento en que la hoja creada con Excel incorpora algún "Macro" o programación VBA (Visual Basic para Aplicaciones), en cuyo caso ninguna de las otras aplicaciones respeta esa programación. Únicamente StarOffice consigue abrir la hoja con la programación, pero surge el siguiente problema: la jerarquía de objetos cambia de una aplicación a otra, por lo que el hecho de copiar la programación directamente es inviable. La única posibilidad consiste en rescribir el código para cada aplicación, con la consiguiente pérdida de tiempo y aumento de posibilidades de error.

Hasta hace unos años la programación efectuada con Macros era más o menos portable entre aplicaciones. De hecho cualquier aplicación de hoja de cálculo entendía la programación mediante macros de acuerdo con el estándar creado por Lotus 1-2-3 (Lotus Script), en particular era interpretable desde Excel y desde Quattro Pro.

A continuación vamos a tratar de crear una metodología de portabilidad de modelos desde Excel hasta las distintas aplicaciones de hoja de cálculo y hasta el formato HTML utilizando para su explicación un supuesto práctico.

2. Supuesto práctico

Para ilustrar los problemas citados, vamos a crear un modelo de simulación de producción-demanda en el cual debemos utilizar necesariamente la programación VBA para generar las tiradas aleatorias.

El modelo, partiendo de una serie de datos históricos referidos a la demanda de un determinado producto durante los 50 últimos días, además de calcular los estadísticos típicos, determina su distribución tabular de frecuencias (*Figura 1*).

TOMA DE DATOS DE LA DEMANDA

DATOS HISTÓRICOS

DÍA	NUM. PEDIDOS
1	230
2	245
3	253
4	220
5	230
6	220
7	240
8	189
9	120
10	253
11	155
12	256
13	260
14	158
15	240
16	187
17	198
18	216
. . .	. . .
50	254

Estadísticos	
TAMAÑO MUESTRA:	50
VALOR MÁXIMO:	448
VALOR MÍNIMO:	120
RANGO (MAX-MIN):	328
MEDIA:	231,14
VARIANZA:	4298,12
DESVIACIÓN TÍPICA:	65,56
COEF. PEARSON:	0,28

ESCALONES	FRECUENCIA
175	12
230	14
285	17
340	4
395	2
448	1
	50

Figura 1

A partir de la distribución de frecuencias acumuladas, y mediante la búsqueda en la mimas de un número aleatorio generando uniformemente entre 0 y 1, se consigue simular el número de unidades que son demandadas en un determinado día (*Figura 2*).

SIMULACIÓN DE LA DEMANDA

ei	es	f	fr	fa	mc
				0,00	147,50
120	175	12	0,24	0,24	202,50
175	230	14	0,28	0,52	257,50
230	285	17	0,34	0,86	312,50
285	340	4	0,08	0,94	367,50
340	395	2	0,04	0,98	421,50
395	448	1	0,02	1,00	
		50,00			

Número aleatorio	Demanda simulada
0,97528	367,5

Figura 2

Para probar la bondad del método se realizan 100 tiradas aleatorias (aquí es donde hemos tenido que recurrir a la programación VBA, mediante un módulo asignado al botón *Tiradas*) y se tabulan de forma que podemos ver las desviaciones existentes entre los valores simulados y los valores reales, observándose una desviación mínima tanto de forma analítica como gráfica (*Figura 3*).

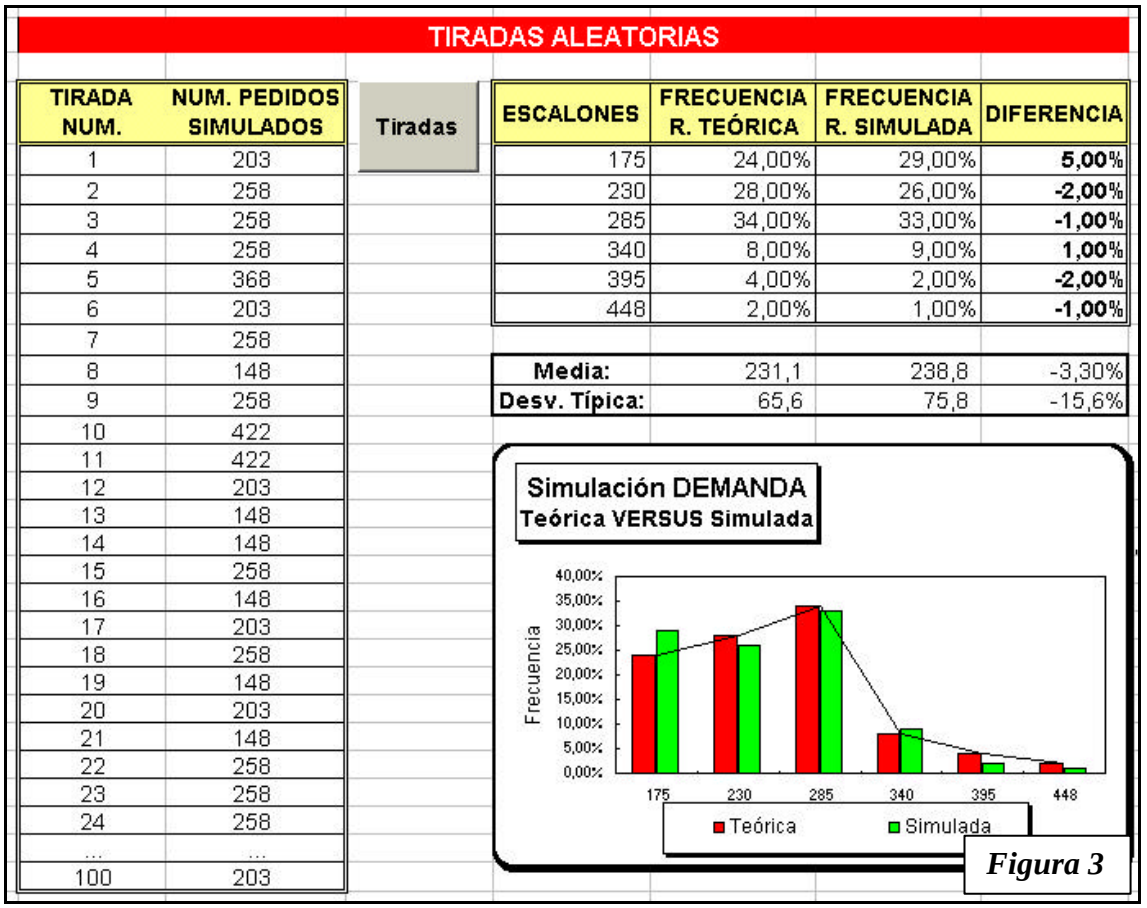
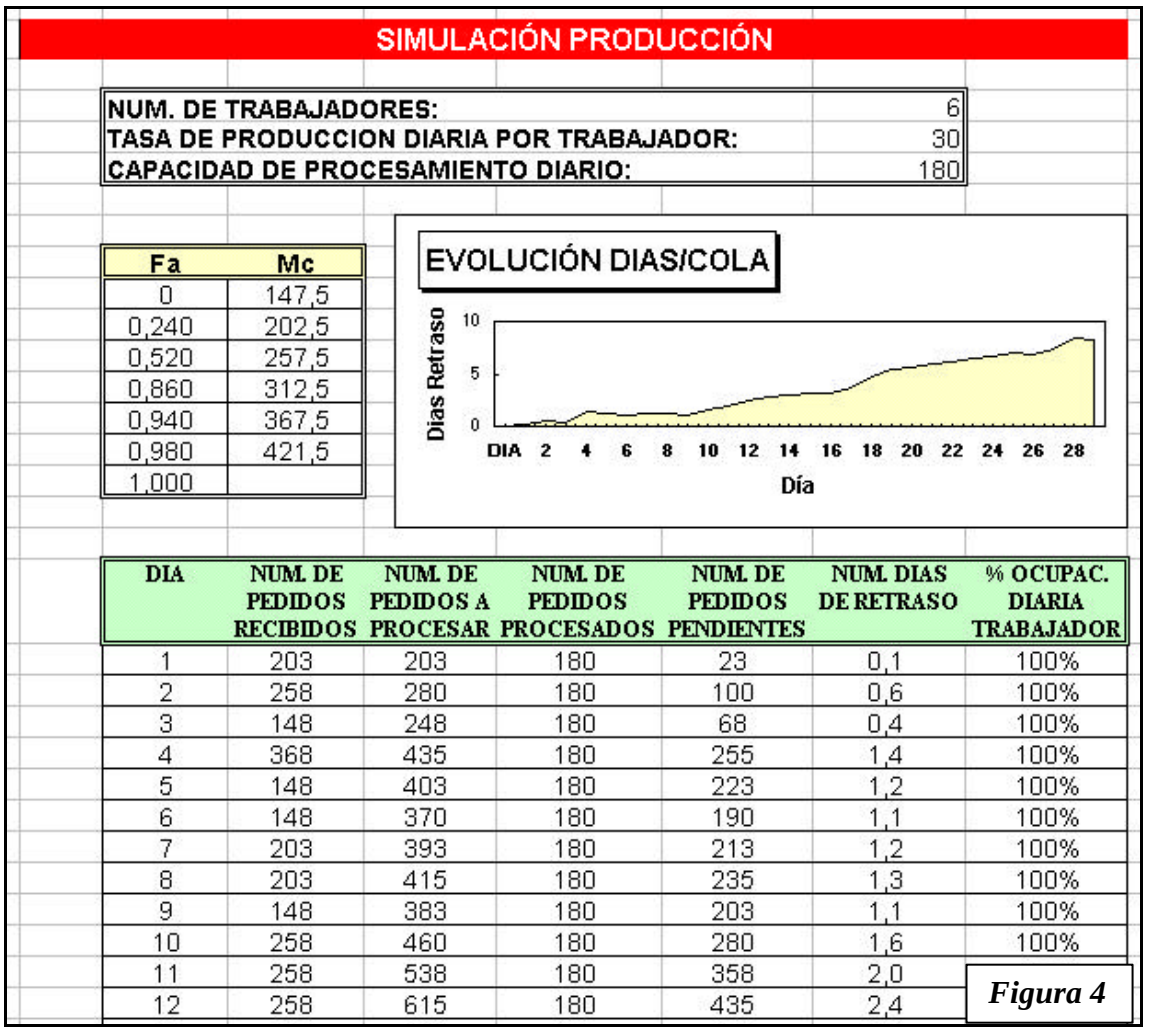


Figura 3

Por último se simula simultáneamente la demanda de un mes, donde se incorporan nuevas variables como son el número de trabajadores ocupados en la producción del artículo considerado, así como su tasa de producción. De esta forma podemos determinar si se produce desabastecimiento o si por el contrario surge demasiado tiempo ocioso para los trabajadores de forma que sabiendo el coste de pérdida de clientes por desabastecimiento y el coste de las horas ociosas se pueda determinar cuál debe ser el número óptimo de trabajadores que permita maximizar el beneficio de la empresa, de acuerdo con que la cola de pedidos (o el número de día de retraso), no exceda de un valor dado (*Figura 4*).



SIMULACIÓN PRODUCCIÓN

NUM. DE TRABAJADORES:	6
TASA DE PRODUCCION DIARIA POR TRABAJADOR:	30
CAPACIDAD DE PROCESAMIENTO DIARIO:	180

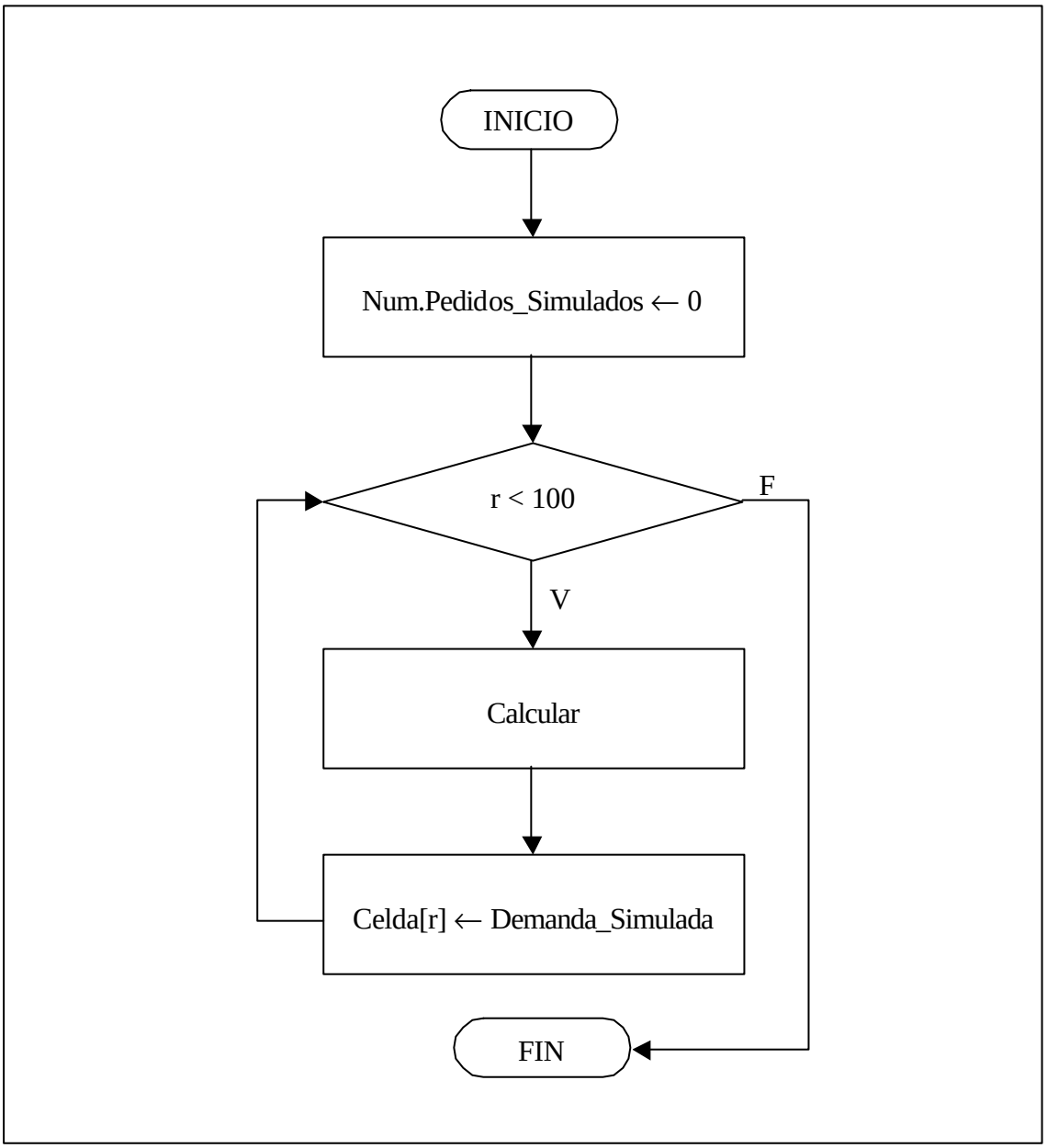
Fa	Mc
0	147,5
0,240	202,5
0,520	257,5
0,860	312,5
0,940	367,5
0,980	421,5
1,000	

DIA	NUM. DE PEDIDOS RECIBIDOS	NUM. DE PEDIDOS A PROCESAR	NUM. DE PEDIDOS PROCESADOS	NUM. DE PEDIDOS PENDIENTES	NUM. DIAS DE RETRASO	% OCUPAC. DIARIA TRABAJADOR
1	203	203	180	23	0,1	100%
2	258	280	180	100	0,6	100%
3	148	248	180	68	0,4	100%
4	368	435	180	255	1,4	100%
5	148	403	180	223	1,2	100%
6	148	370	180	190	1,1	100%
7	203	393	180	213	1,2	100%
8	203	415	180	235	1,3	100%
9	148	383	180	203	1,1	100%
10	258	460	180	280	1,6	100%
11	258	538	180	358	2,0	
12	258	615	180	435	2,4	

Figura 4

Para realizar las tiradas aleatorias utilizamos un módulo bastante sencillo que utiliza rangos nombrados, de forma que vamos copiando en cada tirada la celda "DEMANDA_SIMULADA" en la tabla de salida "NUM._PEDIDOS_SIMULADOS" hasta llegar a llenar el rango "NUM._PEDIDOS_SIMULADOS", que en el ejemplo tiene un total de 100 celdas, lo que implica que vamos a efectuar 100 tiradas, de acuerdo con el organigrama de flujo adjunto.

Veamos a continuación la programación del mismo en los distintos programa objeto de la comparación:



2.1. **Microsoft Excel**

La programación necesaria para realizar las tiradas aleatorias utilizando VBA es la que se presenta a continuación:

```
Sub Tirada()
    Dim r As Integer

    Range("NUM._PEDIDOS_SIMULADOS").Value = ""

    Application.ScreenUpdating = False

    For r = 1 To Range("NUM._PEDIDOS_SIMULADOS").Count
        Calculate
        Range("NUM._PEDIDOS_SIMULADOS").Cells(r, 1).Value = _
            Range("DEMANDA_SIMULADA").Value
    Next
End Sub
```

El módulo creado utiliza el objeto *Range* con rangos nombrados. En primer lugar se borra el rango de salida de las tiradas, para posteriormente ser rellenado fila a fila con los valores que se obtienen de la demanda simulada, hasta completar el número de tiradas previsto, el cual se obtiene en función del tamaño del rango de salida.

2.2. **Lotus 1-2-3**

Lotus 1-2-3 en su edición Millennium incorpora un lenguaje de programación similar a VBA denominado Lotus Smart Script, que es compatible con éste. La diferencia principal no se encuentra por tanto en el lenguaje de programación sino en la jerarquía de objetos que le diferencia de Excel.

Por una parte, la forma de nombrar los rangos difiere notablemente ya que no se utilizan las comillas ni los paréntesis, sino que se utilizan directamente corchetes además de que no existe el objeto *Range* como en Excel. Así mismo, las palabras clave para recalcular la hoja, copiar, pegar y desplazarnos por un rango difieren notablemente, de forma que prácticamente hay que volver a programar el módulo en su totalidad.

El script resultante se puede ver en el siguiente cuadro:

```
Sub Tirada
      Dim r As Integer

      UpdateSheetDisplay = False
      ShowSheetFrame = False

      [NUM._PEDIDOS_SI].Clear ClearData

      [TIRADAS:B4].Select

      For r=1 To 100
            CurrentApplication.Calc
            [DEMANDA_SIMULAD].CopyToClipboard
            Selection.Paste ,False,PasteData + PasteFormulas,,,,
            [TIRADAS].MoveCellPointer $Down,1
      Next r

      'Calcular la frecuencia
      [].Distribution [NUM._PEDIDOS_SI],[TIRADAS:I4..TIRADAS:I9]
End Sub
```

Adicionalmente, y dado que en Lotus no existe una función FRECUENCIA que se pueda recalcular con el resto de la hoja, sino que hay que utilizar un comando de menú para ello, hemos necesitado crear un Script para calcular las frecuencias. Este

Script lo hemos utilizado en el cálculo de las 100 tiradas aleatorias para comparar la distribución simulada con la teórica.

Quattro Pro

Al igual que ocurre con Lotus 1-2-3 la diferencia la encontramos en la jerarquía de objetos. El lenguaje utilizado por Corel en su aplicación no es que sea parecido al VBA sino que es el propio VBA. Sin embargo, la parte negativa la encontramos en los objetos, ya que sólo existe un objeto denominado *PerfectScript* que en lugar de tener propiedades sólo admite procedimientos, los cuales son los mismos que se utilizan en la tradicional programación de macros, heredada de Lotus 1-2-3, y en la grabación de macros Perfect Script, formato propietario de Quattro Pro. Por tanto, nos encontramos ante una programación híbrida que recoge el lenguaje de macros que tanto éxito tuvo en Lotus 1-2-3 junto a la programación VBA propia de Excel.

La programación mediante macros la podemos ver en el siguiente cuadro:

	J	K	L
3	**MACRO**		
4	{SeleccionarBloque TIRADAS:B4..B103}		
5	{BorrarContenidos 0}		
6	{SeleccionarBloque TIRADAS:B4..B4}		
7	{Desde J13;1;100;1;J15}		
8	{SeleccionarBloque TIRADAS:A4..A103}		
9	{EdiciónCopiar}		
10	{SeleccionarBloque TIRADAS:B4..B4}		
11	{PegarEspecial Propiedades}		
12			
13	101		
14	**LLENAR**		
15	{Cálculo}		
16	{BloqueValores B2}		
17	{Abajo}		
18			

siendo la programación VBA necesaria la siguiente:

```
Sub Tirada()
    Dim r As Integer

    PerfectScript.SelectBlock ("NUM._PEDIDOS_SIMULADOS")
    PerfectScript.ClearContents (No)

    For r = 1 To 100
        PerfectScript.Calc
        PerfectScript.SelectBlock ("TIRADAS:B" & (r + 3))
        PerfectScript.PutCell2 _
            (PerfectScript.GetCellValue("DEMANDA_SIMULADA"))
    Next r
End Sub
```

Como se puede observar hay variaciones en el nombre de los comandos utilizados, ya que no existen objetos ni propiedades de los mismos.

2.3. **StarOffice**

StarOffice tiene la ventaja de que soporta de forma nativa los formatos de Excel hasta su versión 2000, aunque lamentablemente no recupera la programación hecha en VBA. En su lugar utiliza un lenguaje BASIC.

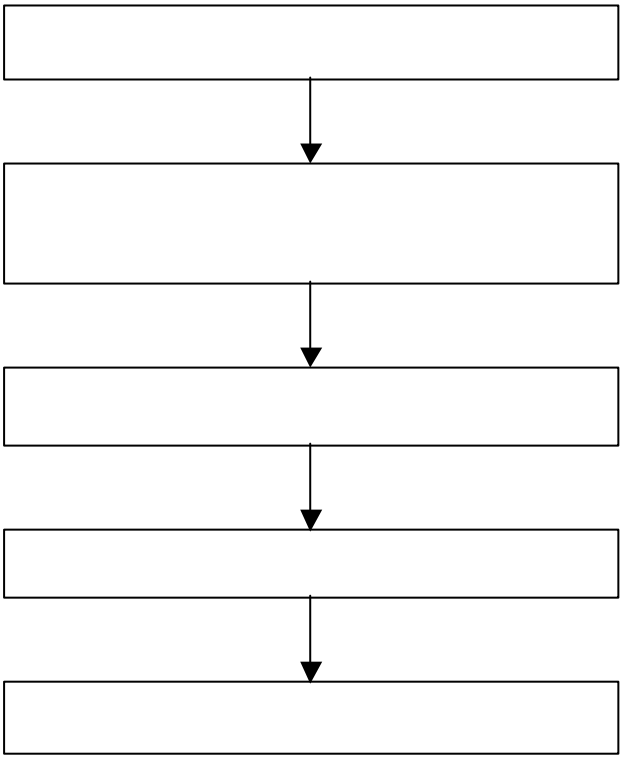
```
Sub Tirada
      Dim r As Integer

      Range("TIRADAS.$B$4:$B$103").Clear( "SVDFN" )

      Range("TIRADAS.$B$4").Select
      For r = 1 To 100
            ActiveDocument.Calculate()
            Range("SIMULACIÓN.$E$14").Copy()
            Selection.InsertContents( "V" )
            Selection.GoDown( 1, FALSE )
      Next r
End Sub
```

Como se puede observar los comandos son similares a los vistos en las otras aplicaciones de hoja de cálculo, pero con ligeras diferencias que obligan a un esfuerzo de traducción desde Excel.

3. Metodología propuesta

En vista de los problemas que presenta el pasar un modelo creado con Excel a cualquier otra hoja de cálculo, hemos creado una metodología que nos permite portar el modelo directamente a un formato que es independiente de las aplicaciones de hoja de cálculo, con lo que adicionalmente conseguiremos implantar el modelo en cualquier

empresa o entidad aunque no dispongan de una aplicación de este tipo. Esta metodología se basa en la utilización de un formato estándar de intercambio de ficheros como es el HTML (HyperText Markup Language), que puede ser leído por cualquier aplicación tanto en Windows como en otros sistemas operativos como Unix, Linux, BeOS, y MacOs entre otros.



Lamentablemente, el formato HTML es un lenguaje más similar al formato de un procesador de texto que al de una hoja de cálculo por lo que no incorpora funciones matemáticas ni estadísticas como la hoja. Esa deficiencia se puede solucionar utilizando varios lenguajes Script, es decir lenguajes sencillos de comandos, entre los que encontramos Visual Basic Script, Jscript y JavaScript, siendo este último el más compatible entre aplicaciones y sistemas operativos. Por último, la aplicación con la que utilizaremos el modelo creado de acuerdo con este formato ya no va a ser una hoja de cálculo (aunque podría serlo), sino que será habitualmente un navegador de páginas Web, como Internet Explorer, Netscape u Opera. Para comprobar que nuestra metodología es lo suficientemente compatible hemos comprobado el modelo resultante con estos tres navegadores en sus versiones Windows y Linux, si bien en el caso de Internet Explorer sólo no hemos podido probarlo en su versión Windows (tanto Win9x como Windows 2000), ya que Microsoft no ofrece productos para sistemas operativos gratuitos como Linux. Además de probarlo con navegadores de Internet, también hemos testado el modelo final con el paquete integrado StarOffice ya que éste, además de incorporar la herramienta de hoja de cálculo que hemos comentando anteriormente, también funciona como navegador Web.

Adicionalmente, el formato HTML presenta la ventaja de que es soportado de forma nativa por Excel 2000, de manera que no hay ningún problema ni al guardar un

libro de Excel en este formato ni al recuperarlo, de forma que mantiene todas sus características como fichero de hoja de cálculo.

3.1. Grabación del fichero HTML con Excel

Utilizando el comando GUARDAR COMO del menú archivo de Excel conseguimos grabarlo como un fichero HTML, el cual al tratarse de un modelo multihoja se genera como 4 ficheros HTML que se graban en una carpeta en la ubicación indicada más 1 fichero índice. Si abrimos el fichero índice con cualquier navegador de páginas Web obtenemos la siguiente imagen (*Figura 5*).

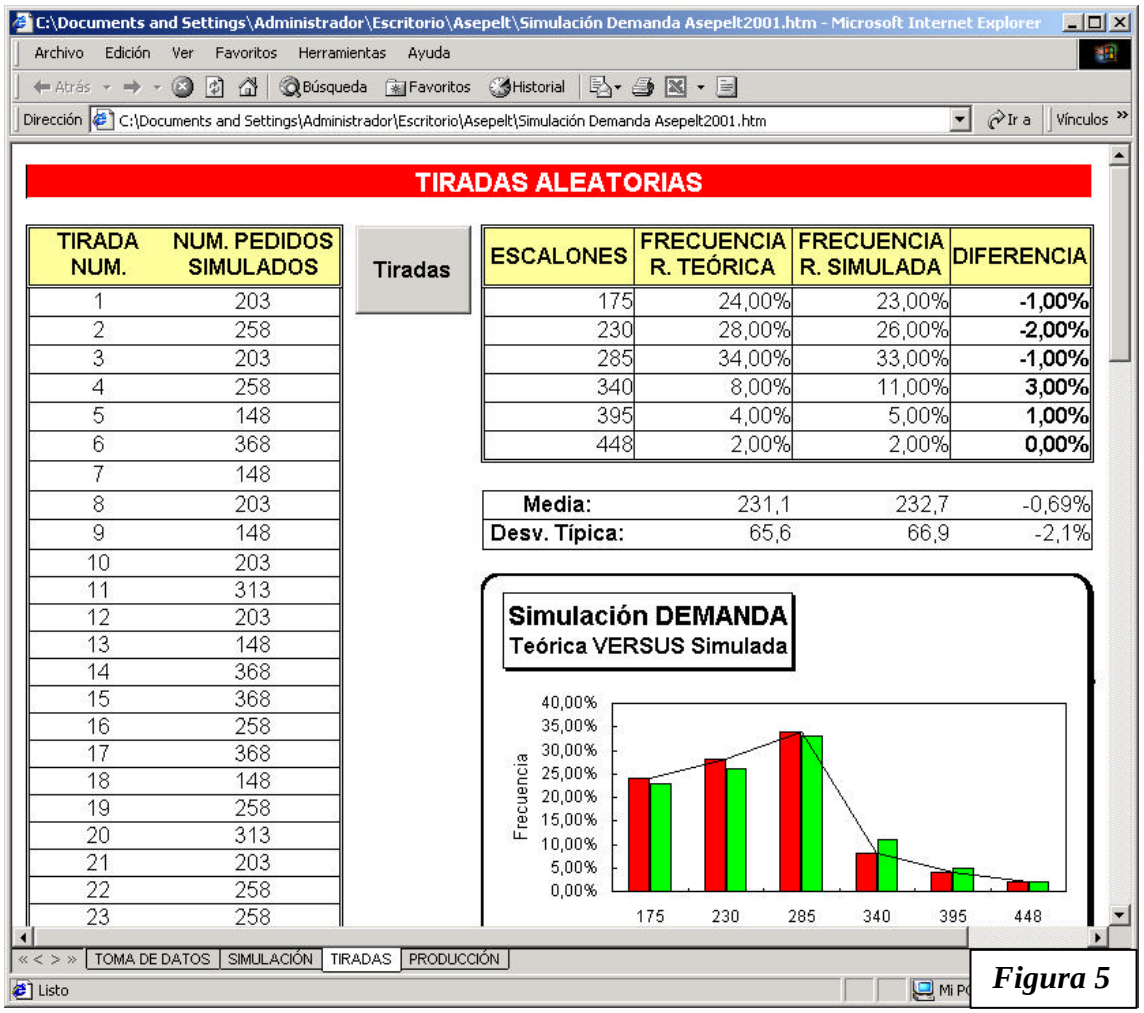


Figura 5

Como se puede observar existen elementos que no son directamente soportados por el formato HTML, como el gráfico que es realmente una imagen que genera automáticamente Excel según el formato GIF, y que por lo tanto no soporta la interactividad propia de un modelo. El segundo elemento que pierde su función original

es el botón de macro *Tiradas* que no realiza ningún tipo de acción al ser pulsado. Además la tirada que aparece es totalmente estática, y por lo tanto no tiene ninguna operatividad.

Por tanto, antes de archivar el modelo en formato HTML hemos de eliminar aquellos elementos que no van a ser interactivos y que solamente pueden inducir a un error ya que por más que se calcule el modelo su aspecto va a ser siempre el mismo.

3.2. Incorporación de las funciones necesarias en JavaScript

Para suplir el problema del recálculo de las cantidades que se suponen deben variar cada vez que carguemos el modelo, o cada vez que pulsemos en el botón de tiradas aleatorias, tendremos que construir las correspondientes funciones, propias de hoja de cálculo, utilizando para ello un lenguaje de programación compatible con el formato HTML. De todos los lenguajes existentes para este fin hemos elegido el *JavaScript* por el hecho de no precisar la instalación de ningún programa añadido como el runtime de Java y porque se puede incorporar dentro de la propia página HTML sin necesidad de necesitar ficheros adicionales. Pese a este último hecho, recomendamos encarecidamente la creación de un fichero de JavaScript (extensión JS) en el que vayamos almacenando todas las funciones que creemos, de forma que cualquier modelo incorpore dicho fichero entre su código, y nos permita depurar funciones sin necesidad de rescribir la página HTML que las llama.

En los modelos desarrollados de esta forma, en lugar de utilizar botones para que se produzca el recálculo del modelo, preferimos utilizar la opción *Actualizar* que todo navegador posee, de forma que al actualizarse el modelo se recalculan todas las fórmulas.

En la hoja número 2 del modelo, en la que se observa la generación de la demanda simulada, simplemente hemos sustituido la función ALEATORIO() de Excel por la función MATH.RANDOM() de *JavaScript*, y a continuación la consulta en lugar de efectuarla mediante la función BUSCARV() de Excel, la realizamos filtrando el valor aleatorio anteriormente obtenido mediante sentencias IF THEN ELSE. Finalmente el aspecto de esta hoja es el mismo que si hubiéramos efectuado los cambios señalados (*Figura 6*).

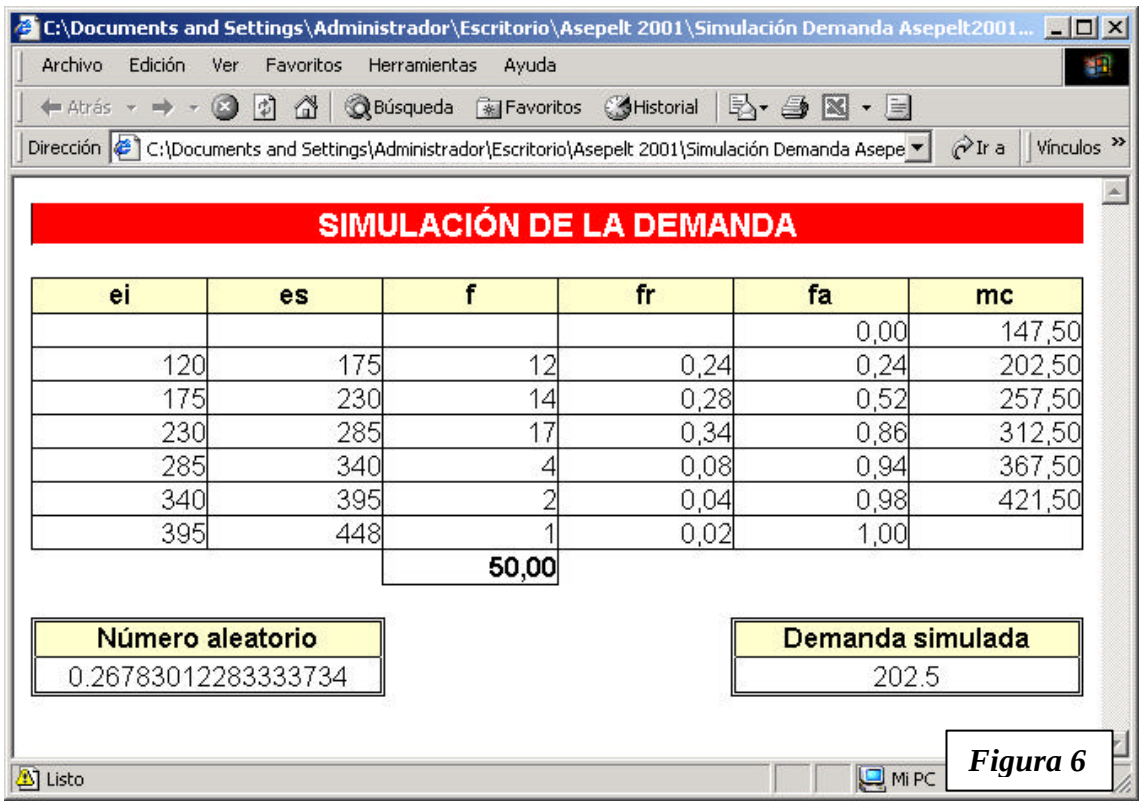
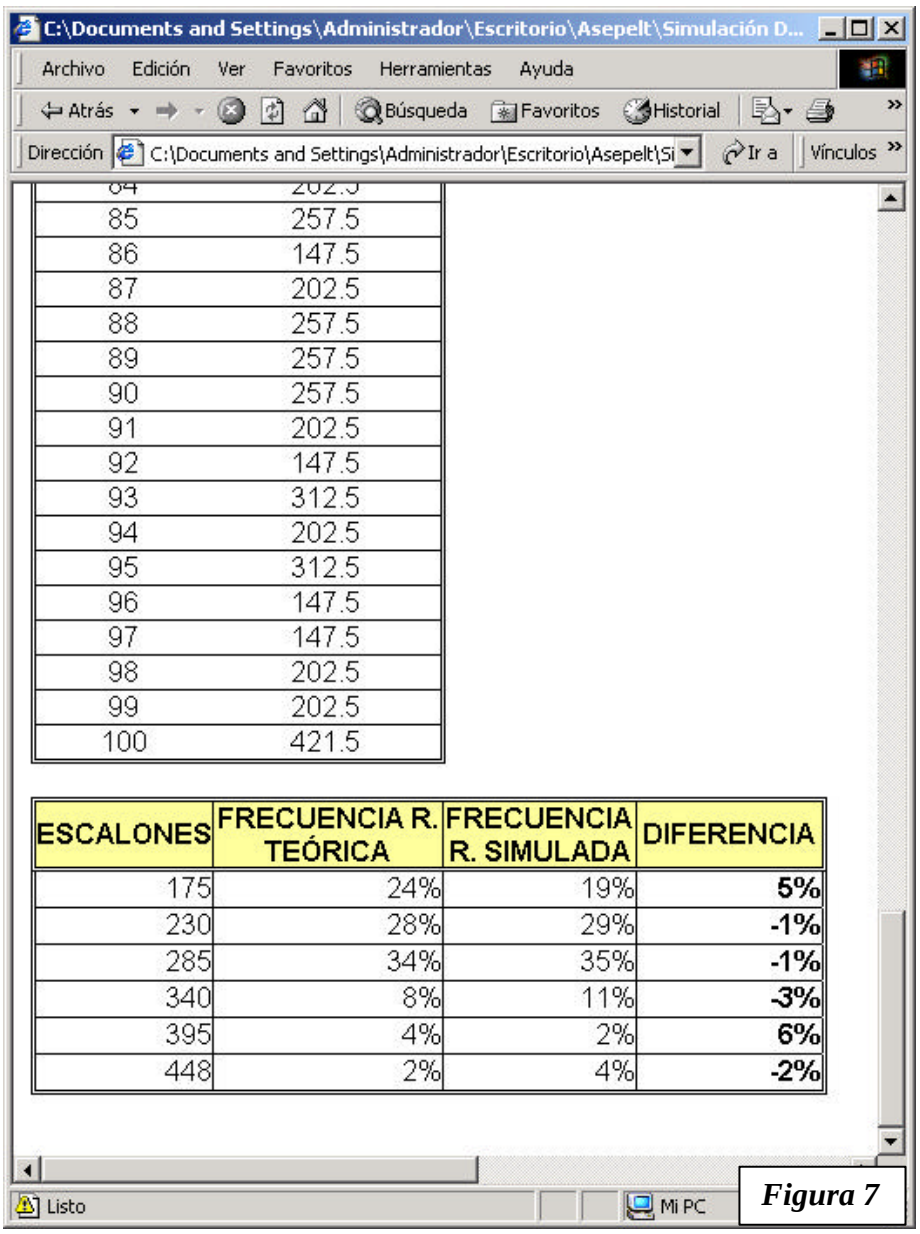


Figura 6

En la hoja correspondiente a las 100 tiradas aleatorias, en la que comparamos las frecuencias teóricas con las simuladas, hemos tenido que modificarla adicionalmente, ya que para calcular la tabla que compare dichas frecuencias es preciso calcular previamente éstas. Con el fin de no hacer todos los cálculos y después presentar entera la hoja, hemos presentado las tiradas según se calculaban y hemos dejado para el final la tabla resumen (*Figura 7*).



85	257.5
86	147.5
87	202.5
88	257.5
89	257.5
90	257.5
91	202.5
92	147.5
93	312.5
94	202.5
95	312.5
96	147.5
97	147.5
98	202.5
99	202.5
100	421.5

ESCALONES	FRECUENCIA R. TEÓRICA	FRECUENCIA R. SIMULADA	DIFERENCIA
175	24%	19%	**5%**
230	28%	29%	**-1%**
285	34%	35%	**-1%**
340	8%	11%	**-3%**
395	4%	2%	**6%**
448	2%	4%	**-2%**

Figura 7

En el caso de la hoja correspondiente a las tiradas mensuales, la variación ha consistido en la incorporación mediante el comando *prompt* de los valores correspondientes al número de trabajadores y a su tasa de producción diaria para que dichos valores puedan ser distintos entre una tirada y otra, de forma que no perdamos en ningún momento la interactividad propia del modelo planteado. A continuación y en base a dichos resultados se genera la tabla correspondiente (*Figura 8*).

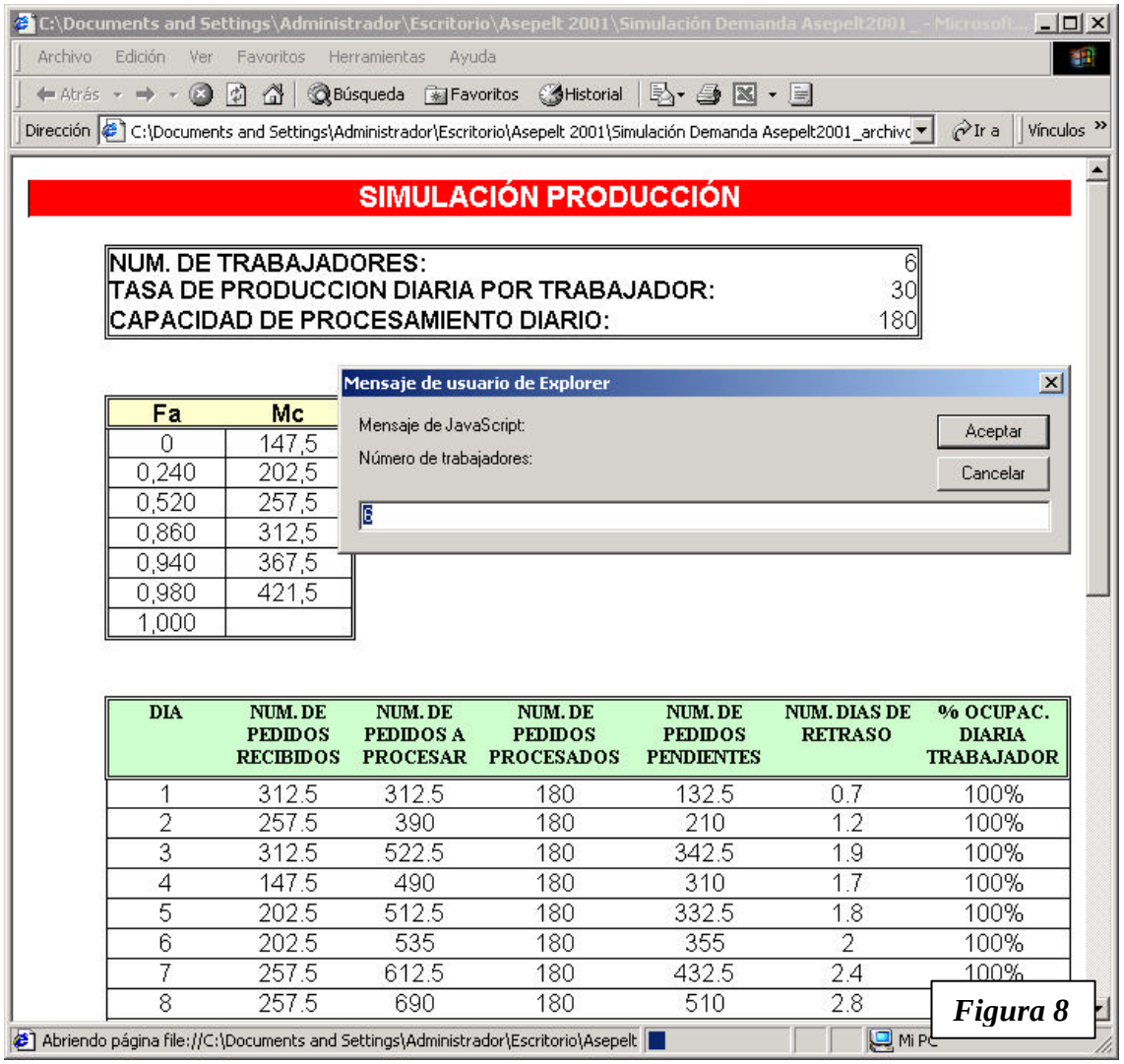


Figura 8

Para ilustrar adecuadamente el funcionamiento de *JavaScript* hemos creído oportuno incluir como ejemplo la programación mediante la cual realizamos la creación de la tabla con la simulación de los 30 días de un mes, en la que podemos ver el uso del comando WRITE() del objeto DOCUMENT con el cual escribimos directamente en la propia página HTML los valores que vamos calculando junto con los comandos propios de dicho lenguaje, obteniéndose finalmente la hoja completa.

```
<script>
pedidos_recibidos = pedidos_procesar = pedidos_procesados =
pedidos_pendientes = retraso = ocupacion = 0
fa1 = fa2 = fa3 = fa4 = fa5 = fa6 = 0

for (r=1;r<=30;r++)
{
        aleatorio=(Math.random())

        if (aleatorio<0.24) pedidos_recibidos=147.5
        if ((aleatorio>=0.24) & (aleatorio<0.52)) pedidos_recibidos=202.5
        if ((aleatorio>=0.52) & (aleatorio<0.86)) pedidos_recibidos=257.5
        if ((aleatorio>=0.86) & (aleatorio<0.94)) pedidos_recibidos=312.5
        if ((aleatorio>=0.94) & (aleatorio<0.98)) pedidos_recibidos=367.5
        if (aleatorio>=0.98) pedidos_recibidos=421.5

       pedidos_procesar = pedidos_pendientes + pedidos_recibidos;
       if (pedidos_procesar > capacidad)
              pedidos_procesados = capacidad;
       else
              pedidos_procesados = pedidos_procesar;

       pedidos_pendientes = pedidos_procesar - pedidos_procesados;
       retraso = Math.round(pedidos_pendientes / capacidad * 10)/10
       ocupacion = Math.round(pedidos_procesados / capacidad * 100)
       document.write('<tr height=20 style='
                     +'\''
                     +'mso-height-source:userset;height:15.0pt'
                     +'\''
                     +'><td height=20 class=xl79 style='
                     +'\''
                     +'height:15.0pt'
                     +'\''
                     +'></td>'
                     +'<td class=xl104 x:num>'
                     +r
                     +'</td>'
                     +'<td class=xl105 x:num="202.5"
x:fmla="=VLOOKUP(RAND(),$B$9:$C$15,2)">'
                     +pedidos_recibidos
                     +'</td>'
                     +'<td class=xl105 x:num="202.5" x:fmla="=C19+0">'
                     +pedidos_procesar
                     +'</td>'
                     +'<td class=xl105 x:num="180" x:fmla="=MIN(D19,$G$5)">'
                     +pedidos_procesados
                     +'</td>'
                     +'<td class=xl105 x:num="22.5" x:fmla="=D19-E19">'
                     +pedidos_pendientes
                     +'</td>'
                     +'<td class=xl106 x:num="0.125" x:fmla="=F19/$G$5">'
                     +retraso
                     +'</td>'
                     +'<td class=xl107 x:num="1" x:fmla="=E19/$G$5">'
                     +ocupacion
                     +'%</td> </tr>')
}
</script>
```

3.3. Pruebas de funcionamiento del modelo

Una vez que hemos terminado nuestro modelo, lo hemos sometido a diversas pruebas de tiradas con el fin de ver si los resultados obtenidos eran similares a los obtenidos con la hoja de cálculo, encontrándonos con un comportamiento muy similar en lo que se refiere a las tiradas aleatorias y a la simulación de la producción.

3.4. Pruebas de compatibilidad

Los resultados comentados anteriormente han sido también validados desde otros navegadores. En este sentido hemos hechos las pruebas oportunas con Netscape (*Figura 9*) StarOffice obteniendo resultados análogos a los obtenidos con Microsoft Internet Explorer. Este hecho debe ser tenido en cuenta ya que demuestra que el modelo y la información generada con Excel puede ser perfectamente accesible desde aplicaciones no desarrolladas por Microsoft, con idénticos resultados.

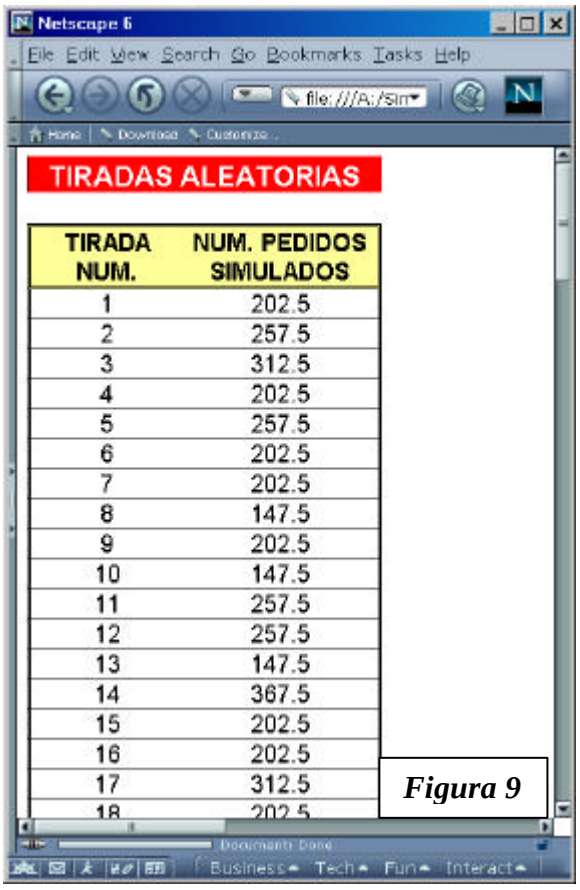


Figura 9

Además de utilizar distintos navegadores también lo hemos llegado a otro sistema operativo como es Linux, con su navegador Konqueror (*Figura 10*).

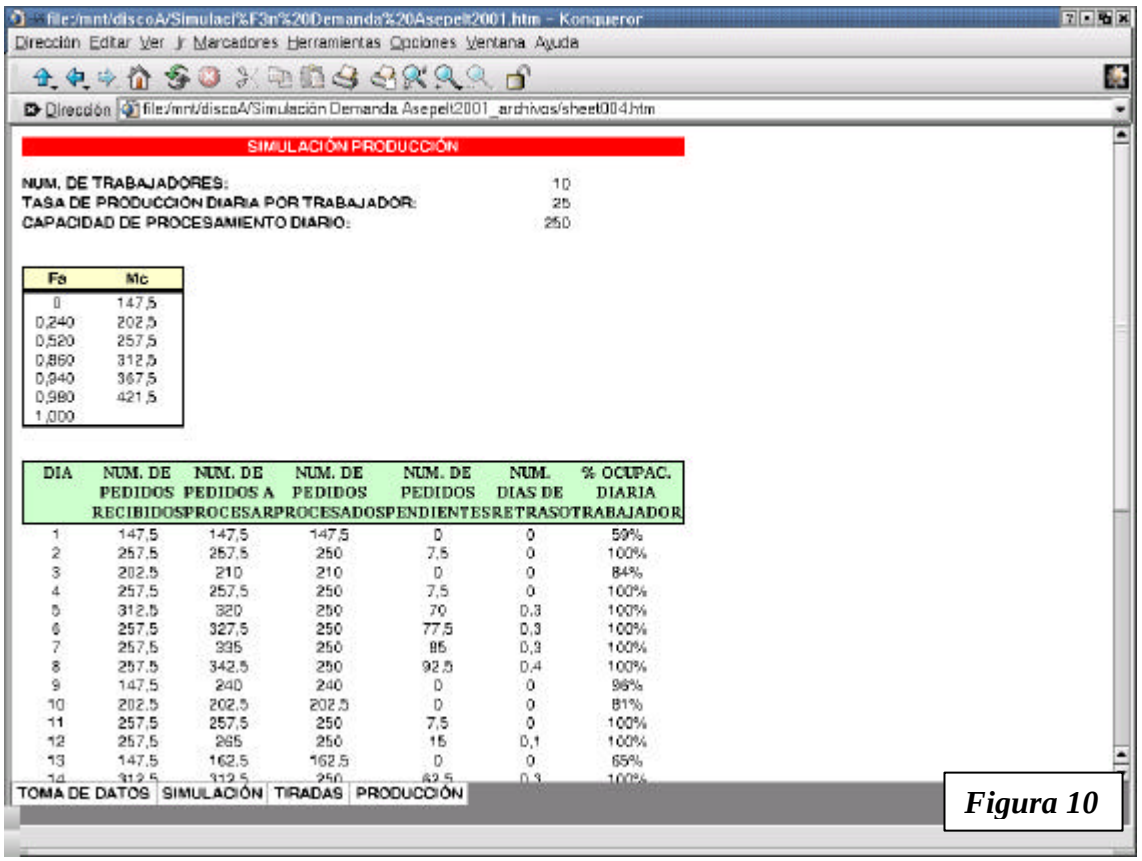


Figura 10

3.5. Modelo final

Una vez hechas todas las pruebas oportunas el modelo puede ser finalmente implantado en la empresa o institución a la que va dirigido, sin tener que preocuparnos nada más que del mantenimiento adecuado de los ficheros creados, con lo cual creemos que la metodología planteada puede aportar una solución viable a los problemas de compatibilidad que dificultan la migración de unos programas a otros.

4. Bibliografía

- Martín, M.; Hasen, S.M.; Klingher, B. *LA BIBLIA DE EXCEL 2000*. Anaya Multimedia. Madrid, 1999.
- Kolbeck, R. *EL GRAN LIBRO DE JAVASCRIPT*. Marcombo. Barcelona, 1997.
- Walkenbach, J. *PROGRAMACIÓN EN EXCEL 2000 CON VBA*. Anaya Multimedia. Madrid, 2000.
- *CURSO DE JAVASCRIPT*. http://www.redestb.es/javaaula/cursjava.htm