Generación de Tareas Periódicas y Aperiódicas para Simulación de Sistemas de Tiempo Real

Gabriela Olguín, Laura Biscayart, José M. Urriza

Universidad Nacional de la Patagonia San Juan Bosco, Puerto Madryn, Argentina.
golguin@unpata.edu.ar, lalibiscayart@gmail.com,
josemurriza@unp.edu.ar

Resumen. En la disciplina Sistemas de Tiempo Real, una forma de comprobar las nuevas teorías o modelos heterogéneos de planificación, es realizar simulaciones masivas con los nuevos algoritmos desarrollados. Estas simulaciones permiten validar los resultados antes de ser implementados. No obstante, es necesario generar millones de sistemas heterogéneos sintéticos. Estos sistemas deben ser generados bajo ciertas premisas y formatos. En este trabajo, se presenta un software para generar conjuntos de sistemas de tiempo real heterogéneos. El software se encuentra desarrollado en el lenguaje Ada 2005, es de código abierto, genera conjuntos de tareas periódicas, conjuntos de tareas aperiódicas y verifica que los sistemas periódicos sean planificables por *Rate Monotonic*, *Deadline Monotonic o Earliest Deadline First*. Se adapta fácilmente a diferentes requerimientos de los simuladores y dispone de una interfaz gráfica.

Palabras Clave: Generación de tareas, Simulación de Sistemas, Tiempo Real, Sistemas de Tiempo Real Heterogéneos, Tareas periódicas, Tareas aperiódicas, Planificabilidad.

1 Introducción

En la disciplina Sistemas de Tiempo Real (*STR*), la simulación de sistemas permite comprobar el desempeño de nuevos algoritmos surgidos de nuevas teorías o modelos.

Algunos de estos algoritmos pueden ser los que determinen si un sistema es factible, o permitan planificar un conjunto de tareas heterogéneas, o sean utilizados para ahorro de energía en dispositivos móviles o para incrementar la robustez mediante técnicas de tolerancia a las fallas, etc. Estos deben ser exhaustivamente comprobados desde distintos puntos vista, como su rendimiento en comparación con otros, o sus costos computacionales temporales y espaciales.

Para lograr esto, es necesario realizar simulaciones sobre millones de sistemas sintéticos en diversas condiciones de carga. De esta manera es posible determinar cuál es el comportamiento del algoritmo.

Consecuentemente, es necesaria una herramienta que permita generar automáticamente estos sistemas sintéticos. Para ello se requiere la construcción de

una aplicación específica, que hasta la actualidad es *ad-hoc*. Debido a que la herramienta es *ad-hoc*, resulta poco flexible para futuras investigaciones.

En los grupos de investigación científica de *STR*, en donde se investigan varias sub-disciplinas, contar con una aplicación que permita generar conjuntos de sistemas sintéticos de manera rápida y flexible, permite simplificar los experimentos.

Hasta la redacción de este trabajo, no fue posible encontrar aplicaciones para generación de conjuntos de *STR* de código abierto y flexible. Sin embargo, se encontró un reporte interno ([1]), donde se presenta un marco de trabajo para la generación de conjuntos de tareas de tiempo real sintéticos. El mismo se encuentra realizado en .NET y cabe destacar que utiliza un modelo de tareas más simplificado que el que se presenta en este trabajo.

Consecuentemente, teniendo como principales objetivos la flexibilización para aplicarse a distintas subdisciplinas, la velocidad de generación y no seguir creando aplicaciones que generen conjuntos de tareas *ad hoc*, sino que dar una solución amplia a este problema, es que se optó por construir un software de generación adaptable.

El modelo de tarea mínimo utilizado está basado en el presentado por Liu y Layland ([2]), el cual es un trabajo liminar de la disciplina. En 1993, Audsley ([3]) presentó un modelo ampliado de cómo caracterizar una tarea de tiempo real. Luego numerosos trabajos aportaron características adicionales al modelo, entre los cuales se puede citar a [4], [5] para computación imprecisa, etc. El modelo ampliado desarrollado en este trabajo cubre la mayoría de las simulaciones requeridas en la disciplina. Sin embargo, a diferencia de lo presentado en [6], se busca no limitar el hiperperíodo del sistema.

El objeto de este trabajo es presentar cómo fue diseñada y construida una herramienta de generación, que brinda la mayor flexibilidad en la creación de los conjuntos de tareas. La posibilidad de que los conjuntos sean configurables simplifica la generación de los datos de entrada para cada una de las simulaciones, las cuales suelen tener requerimientos diferentes para distintas sub-disciplinas de Tiempo Real.

El software desarrollado[1] es configurable, de uso libre y de código abierto. Está desarrollado en Ada 2005 ([7]). Es sencillo de utilizar y genera millones de sistemas en un tiempo razonablemente pequeño. Para facilitar la integración con cualquier simulador, la salida puede guardarse en archivos con formato *.xml,* en archivos de texto plano *.txt* y en un formato vertical *.vert* (ver sección 3.2).

En las siguientes secciones se presentará cómo el software fue desarrollado y qué posibles conjuntos pueden ser generados con el mismo. Este trabajo está organizado como se describe a continuación. En la sección 2 se presenta el Modelo utilizado para la generación, donde se describe cómo está compuesto cada sistema, cómo se obtienen cada uno de sus componentes, el cálculo del hiperperíodo, la verificación del factor de utilización obtenido y la generación de las tareas aperiódicas. En la sección 3 se describe el generador implementado, explicando sus entradas y salidas. En la sección 4 se presentan los experimentos realizados. Por último en la sección 5 se presentan las conclusiones y se mencionan los trabajos futuros.

[1] El software se puede encontrar en http://www.rtsg.unp.edu.ar/

2 Modelo Utilizado para la Generación

A continuación se detalla el modelo de sistemas y el marco de trabajo utilizado. El mismo es similar al presentado en los trabajos de [2, 3]. Además se detalla el modelo de tareas aperiódicas. En la Tabla 1 se define la terminología utilizada para la definición de las tareas del sistema y de las tareas aperiódicas.

Tabla 1. Terminología

Tiempo de ejecución (C_i)	Máximo tiempo que podría tomar la ejecución de una tarea. Normalmente se define como el *peor caso de tiempo de ejecución* (*WCET*). También se pueden generar el mejor tiempo de ejecución (BC_i) y el tiempo de ejecución promedio (AC_i) de la tarea.
Período (T_i)	Intervalo de tiempo entre arribos de una tarea periódica. Para las tareas aperiódicas, este tiempo significa el instante de arribo de la tarea (T_j).
Vencimiento (D_i)	Instante de tiempo máximo, relativo a la instanciación de la tarea, en que la misma debe completar su ejecución.
Tiempo de bloqueo (B_i)	Máximo tiempo en que una tarea podría estar demorada por tareas de menor prioridad con las que comparte recursos.
Retardo de instanciación (*Jitter*) (J_i)	Diferencia entre el tiempo de arribo esperado de la tarea y el tiempo en que está lista para su ejecución.
Offset (Of_i)	Máximo desplazamiento del tiempo de arribo de una tarea respecto al tiempo de referencia del sistema.
Tiempo de ejecución opcional (Co_i)	Máximo tiempo que podría ejecutarse una instancia opcional de una tarea, luego de completar su tiempo de ejecución mandatorio.
Hiperperíodo	Es el mínimo común múltiplo de los períodos de las tareas.

Cada sistema **S** generado contiene *n* tareas periódicas. Cada tarea posee su peor caso de tiempo de ejecución (C_i), el mejor (BC_i) y el promedio (AC_i), el período (T_i), el vencimiento (D_i), el tiempo de bloqueo (B_i), el retardo de instanciación (J_i), el tiempo opcional de ejecución (Co_i) y el offset (Of_i).

Las tareas aperiódicas no se asocian a ningún sistema en particular, y se describen por su instante de arribo (T_j) y su peor tiempo de ejecución (A_j).

En la Tabla 2 se listan las variables y notaciones utilizadas.

Tabla 2. Notación

AC_i	Tiempo promedio de ejecución de la tarea *i*, aleatorio entre BC_i y C_i.	BC_i	Mejor tiempo de ejecución de la tarea *i*.
A_j	Peor tiempo de ejecución de la tarea aperiódica *j*.	BC_{max}	Valor máximo del porcentaje de variación de BC_i respecto de C_i.
A_{jmax}	Máximo tiempo de ejecución para una tarea aperiódica.	BC_{min}	Valor mínimo del porcentaje de variación de BC_i respecto de C_i.

B_i	Máximo tiempo de bloqueo de la tarea *i*.
$B_{\max}$	Valor máximo del porcentaje de variación de B_i respecto de C_i.
$B_{\min}$	Valor mínimo del porcentaje de variación de B_i respecto de C_i.
C_i	Peor tiempo de ejecución de la tarea *i*.
Co_i	Tiempo opcional de ejecución para la tarea *i*.
Co_{max}	Valor máximo del porcentaje de variación de Co_i respecto de C_i.
Co_{min}	Valor mínimo del porcentaje de variación de Co_i respecto de C_i.
D_i	Vencimiento de la tarea *i*.
$D_{\max}$	Valor máximo del porcentaje de variación de D_i respecto de T_i.
$D_{\min}$	Valor mínimo del porcentaje de variación de D_i respecto de T_i.
D_{rel}	Relación de D_i respecto de T_i (<=, >=, =, <=>).
DE	Función de distribución de probabilidad exponencial.
DU	Función de distribución de probabilidad uniforme.
i	Una tarea ($i = 1, ..., n$).
IAT_{max}	Máximo tiempo entre arribos de tareas aperiódicas.
J_i	Retardo de instanciación de la tarea *i*.
$J_{\max}$	Valor máximo del porcentaje de variación de J_i respecto de T_i.
$J_{\min}$	Valor mínimo del porcentaje de variación de J_i respecto de T_i.
mcd	Función para calcular el máximo común divisor entre dos números naturales.
mcm	Función para calcular el mínimo común múltiplo entre dos números naturales.
min (a,b)	Función para calcular el menor entre dos valores.
mod	Función para calcular el resto de la división entre dos números naturales.
n	Cantidad de tareas en cada sistema.
Of_i	Máximo tiempo de offset de la tarea *i* respecto al instante de referencia.
$Of_{\max}$	Valor máximo del porcentaje de variación de Of_i respecto de T_i.
$Of_{\min}$	Valor mínimo del porcentaje de variación de Of_i respecto de T_i.
R	Cantidad de rangos para valores de los T_i.
$R_{(i \bmod R) \max}$	Valor máximo del rango al que pertenecerá el T_i de la tarea *i*.
$R_{(i \bmod R) \min}$	Valor mínimo del rango al que pertenecerá el T_i de la tarea *i*.
random	Función para generar un valor aleatorio con una *DU* entre 0 y 1.
random (*min,max*)	Función para generar un valor aleatorio con una *DU* entre los valores min y max.
S(*n*)	Sistema de *n* tareas.
T	Máximo tiempo de arribo para cualquier tarea aperiódica.
T_i	Período de la tarea *i*.
T_j	Instante de arribo de la tarea aperiódica *j*.
U	Factor de utilización de cada sistema (en porcentaje).
U_{final}	Factor de utilización del sistema generado.
U_i	Factor de utilización de la tarea *i*.
α, β	Variables auxiliares.
ε	Error porcentual admitido para cada *U*.
λ (*Lambda*)	Tasa de arribo de tareas aperiódicas para una distribución exponencial.
μ (*Mu*)	Tasa de servicio de tareas aperiódicas para una distribución exponencial.

En las subsecciones siguientes se expone la forma en que se obtiene cada uno de los parámetros de las tareas. Luego se agregan otras funciones de validación y observación de los sistemas obtenidos.

2.1 Generación del Tiempo de Ejecución, el Período, y el Factor de Utilización de cada tarea (C_i, T_i, U_i)

Para comenzar con cada conjunto de tareas, se generan valores para todos los T_i. Los T_i pueden pertenecer a diferentes rangos (R), para casos en que se necesita simular tareas con períodos en rangos muy diferentes. Por ejemplo algunas tareas con período entre 10^1 y 10^2, y otras entre 10^3 y 10^4. La cantidad de rangos y el valor mínimo y el máximo para cada uno son configurables. La cantidad de tareas en cada rango será n/R, quedando mayor cantidad de tareas en los rangos inferiores si n no fuera exactamente divisible por R.

Los valores para T_i se generan aleatoriamente con una distribución de probabilidad uniforme (*DU*) o exponencial (*DE*) dentro de cada rango. Enumerando los rangos de 0 a *R-1*, para generar cada T_i:

$$\alpha = \mathrm{R}\left(i \bmod R\right)_{\min}, \beta = \mathrm{R}\left(i \bmod R\right)_{\max} \Rightarrow T_i = random(\alpha, \beta)$$

Para generar los U_i se utiliza el algoritmo *UUniFast* propuesto por Bini en [8]. Con esto se obtienen factores de utilización para cada tarea distribuidos uniformemente. El algoritmo que se presenta fue extraído del trabajo presentado en [9]. Una vez generados U_i y T_i, se calcula C_i como $C_i = U_i.T_i$.

2.2 Generación del Mejor Tiempo de Ejecución (BC_i) y del Tiempo de Ejecución Promedio (AC_i)

El mejor tiempo de ejecución (BC_i) de una tarea periódica es el mínimo tiempo que podría tomar la ejecución de la tarea. Se genera como un valor aleatorio entre un mínimo y un máximo porcentaje respecto de C_i. A partir de esos dos valores (C_i y BC_i) se genera el tiempo promedio de ejecución (AC_i), con una *DU* entre BC_i y C_i.

$$BC_i = random\left(BC_{\min}.\ C_i, BC_{\max}.C_i\right) \qquad AC_i = random(BC_i, C_i)$$

2.3 Generación de los Vencimientos (D_i)

El vencimiento de cada tarea, D_i, se genera como un porcentaje de T_i. Son configurables los porcentajes mínimo y máximo, y la relación entre D_i y T_i (si es igual, menor o igual, mayor o igual, o arbitrario). En los modelos utilizados en [2] D_i se considera igual a T_i, pero en otros contextos, como en [10], esta relación podría ser arbitraria. Los cálculos se realizan siguiendo el algoritmo:

```
α = random( Dmin, Dmax )
case Drel
   si "<=>" → seleccionar aleatoriamente con una DU una de las otras
   opciones
    si " = " → Di = Ti
    si "<= " → Di = Ti – α*Ti
    si ">= " → Di = Ti + α*Ti
```

2.4 Generación de los Tiempos de Bloqueo (B_i)

El peor tiempo de bloqueo para cada tarea, B_i, es opcional, se genera como un porcentaje de C_i y pueden configurarse los porcentajes mínimo y máximo. Se genera un valor α con una *DU* entre el mínimo y el máximo porcentaje configurado de C_i, y luego se calcula B_i como:

$$\alpha = random(B_{\min}, B_{\max}) \Rightarrow B_i = \alpha.C_i$$

2.5 Generación de los Jitters (J_i)

El tiempo de retardo de instanciación (Jitter), J_i, es el tiempo en el cual el sistema no realiza ninguna clase de tarea, esperando que la tarea i se encuentre disponible o se instancie. Este tiempo no depende directamente del periodo de la tarea, y suele ser muy pequeño. A fines de la simulación se genera como un porcentaje de T_i. Puede configurarse el porcentaje mínimo y el máximo, y si se genera o no. Se genera un valor α con una *DU* entre el mínimo y el máximo porcentajes respecto de T_i:

$$\alpha = random(J_{\min}, J_{\max}) \Rightarrow J_i = \alpha.T_i$$

2.6 Generación del Offset (Of_i)

El tiempo de offset (Of_i), al igual que el tiempo de ejecución opcional (Co_i), es un parámetro de generación optativa. Se utiliza en casos en que se desea simular el arribo de tareas con retraso. El Of_i se genera como un porcentaje de T_i. Primero se genera un valor α aleatoriamente con una *DU* entre el valor mínimo y el máximo, luego se obtiene Of_i:

$$\alpha = random(Of_{\min}, Of_{\max}) \Rightarrow Of_i = \alpha.T_i$$

2.7 Generación del Tiempo de Ejecución Opcional (Co_i)

El tiempo de ejecución opcional (Co_i) es un valor de generación optativa que se utiliza en el estudio de modelos de computación imprecisa como los presentados en [11, 12]. Se genera aleatoriamente con una *DU* como un porcentaje de C_i. Pueden configurarse los porcentajes mínimo y máximo y la precisión de salida.

$$Co_i = random(Co_{\min}.C_i, Co_{\max}.C_i)$$

2.8 Cálculo del Hiperperíodo

Dada la periodicidad de las tareas, un instante cualquiera de carga se repite nuevamente transcurrido el hiperperíodo del sistema. Se calcula el mínimo común múltiplo (*mcm*) entre todos los T_i, siguiendo el algoritmo:

```
mcm_parcial = T1
for i = T2 to Tn
```

```
    mcm_parcial = mcm( Ti, mcm_parcial )
end for
hiperperiodo = mcm_parcial
```

El Mínimo Común Múltiplo (*mcm*) se calcula a partir del Máximo Común Divisor (*mcd*) entre *a* y *b*, siendo *a* > *b*:

$$mcm(a,b) = \frac{a.b}{mcd(a,b)} \qquad mcd(a,b) = a \bmod b \mid b \bmod (a \bmod b) = 0$$

Como básicamente el *mcm* es el producto de la factorización en números primos de los periodos de las tareas, rápidamente el hiperperíodo puede llegar a valores muy grandes. Por eso se utiliza un entero de 18 dígitos. Cuando el valor del hiperperíodo requiere un número mayor, el valor que se muestra es 999.999.999.999.999.999. Como en los simuladores suele utilizarse un formato que permite representar valores positivos de hasta 2^{31} (2.147.483.648), el valor retornado por el generador no es utilizable, así que se retorna 2^{31}.

2.9 Verificación del Factor de Utilización (*U*)

Al terminar de generar todos los valores, se verifica que el factor de utilización del sistema obtenido se acerque al objetivo, considerando un margen de error ε (configurable). Para esto se vuelve a calcular como:

$$U_{final} = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

Y el resultado obtenido se valida mediante:

$$U_{final} - \varepsilon \le U_{final} \le U_{final} + \varepsilon$$

2.10 Verificación de planificabilidad por *RM* o *DM*

Los sistemas generados son planificables por *EDF* (*Earliest Deadline First*) si en la generación de los sistemas el $U_{final} + \varepsilon$ no supera al 100% ([2]).

Para determinar si son planificables por algoritmos con prioridades fijas como *RM* o *DM*, se utiliza el algoritmo de búsqueda de un punto fijo ([13]). Si se encuentra un punto fijo tal que en ese instante todas las tareas se cumplan antes de su vencimiento, el sistema es planificable. Si se encuentra alguna tarea que se vence, se deja de buscar y se determina que el sistema no es planificable.

2.11 Generación de Tareas Aperiódicas (A_j y T_j)

Las tareas aperiódicas se utilizan para simular sistemas heterogéneos. Poseen dos parámetros, que son su instante de arribo (T_j) y su tiempo de ejecución (A_j). El instante de arribo se cuenta desde el inicio de la ejecución del sistema. El máximo tiempo entre arribos (*IAT*) es configurable. La duración de cada tarea (A_j) se genera

aleatoriamente con una *DU* o una *DE* de tasa μ. Puede configurarse la duración máxima (A_{jmax}), y siempre será un valor mayor a 0 y menor o igual que A_{jmax}.

$$A_j = random(0, A_{j\max})$$

Los tiempos de arribo se generan con una *DE* o una *DU*, a partir del arribo de la última tarea generada, considerando el máximo tiempo entre arribos (IAT_{max}) y el mayor tiempo de arribo configurado (T). Cuando se selecciona una *DE* esto representa una cola M/M/1 con distribución de *Poisson* en el arribo de las tareas aperiódicas donde λ es la tasa de arribos en el intervalo.

$$\alpha = 1 - e^{-\lambda.random(0, IAT\max)}, \beta = \alpha.IAT_{\max} \Rightarrow \quad T_j = T_{j-1} + \alpha$$

La cantidad de tareas generada depende de los valores obtenidos para tiempos de arribo.

3 Descripción del Generador

3.1 Configuración de los parámetros

Por medio de la interfaz gráfica ([14]) se ingresan los valores para determinar el comportamiento del generador. Cuenta con dos secciones principales: Tareas Periódicas y Tareas Aperiódicas, y una sección secundaria, para seleccionar los formatos de salida. Dentro de la definición de las Tareas Periódicas hay tres paneles, para determinar: características generales, características de los sistemas y características de las tareas. Dentro de la definición de las Tareas Aperiódicas hay un panel para indicar sus parámetros de generación.

Al iniciarse, el programa lee y muestra los valores por defecto. Éstos están en el archivo *setup.txt*, el cual debería encontrarse en la misma carpeta que el programa ejecutable. De no ser así, todos los valores serán cero o falsos.

Para generación de Tareas Periódicas puede configurarse: la cantidad máxima de sistemas a generar, el factor de utilización deseado (U), el margen de error porcentual (ε), la cantidad de tareas (n) en cada sistema, la forma de ordenar las tareas (*RM* o *DM*), el criterio de planificabilidad a utilizar y las características de cada variable que describe a las tareas, incluyendo si se generan o no, con cuánta precisión, valores mínimo y máximo de porcentaje si se generan en función de otra variable y, en el caso del período, con qué tipo de distribución.

Para generación de Tareas Aperiódicas puede configurarse: los valores máximos de tiempo de arribo (T), tiempo entre arribos (IAT_{max}) y tiempo de ejecución (A_{jmax}), la distribución de probabilidad para generar cada valor y, en el caso de la *DE*, el valor de la tasa de arribo y de servicio. Para el A_j puede indicarse también la precisión.

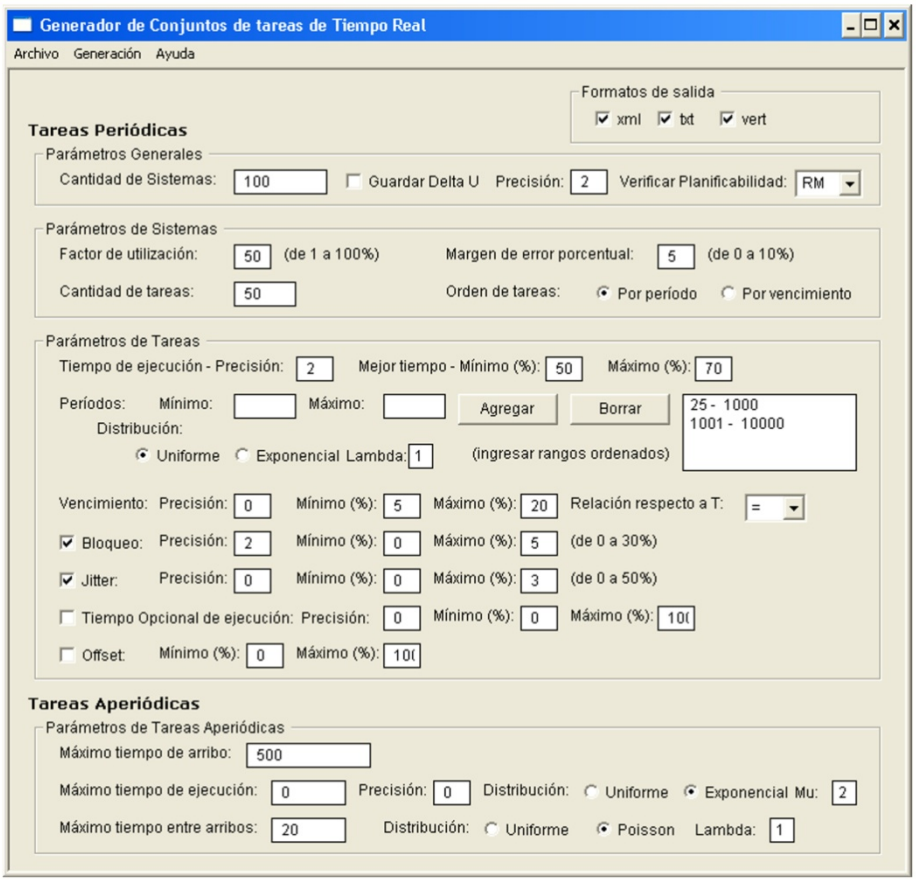

Fig. 1. Interfaz gráfica.

3.2 Descripción del Menú

En el menú *Archivo* se encuentran las opciones para crear, abrir y guardar un archivo de configuración. En el menú *Generación* se encuentran las opciones para iniciar la generación de los sistemas o las tareas aperiódicas de acuerdo a los valores de la interfaz. En el menú *Ayuda* se accede al contenido de la ayuda.

3.3 Archivo de Configuración

La interfaz gráfica está disponible en sistema operativo Windows. Para utilizar el generador en otros sistemas puede configurarse mediante el archivo *setup.txt*. Este contiene todas las variables configurables.

Para facilitar la lectura y mantenimiento del archivo de configuración se permite el uso de comentarios de línea, indicados con un carácter # al comienzo de la línea, y se ignoran los espacios y líneas en blanco.

Cada parámetro tiene un nombre fijo, seguido de un signo = y su valor en números naturales. Se define un parámetro por línea y se ignora todo lo que se escriba a continuación del valor. Puede modificarse el orden de los parámetros, agregarse o quitarse, aunque se ignoran también los parámetros nuevos, y los valores que no se hayan asignado se inicializan en cero.

3.4 Los Archivos de Salida

Cada archivo *xml* con conjuntos de tareas periódicas tiene el siguiente formato:

```
<?xml version="1.0" ?>
<Set size="cantidad_de_sistemas" n="n">
  <S count="contador_de_sistemas" U="U" mcm="hiperperíodo">
    <i nro="i" C="Ci" BC="BCi" AC="ACi" T="Ti" D="Di" B="Bi" J="Ji"
       Of="Ofi" Co="Coi" />
    ...
  </S>
  ...
</Set>
```

En el encabezado se muestra la cantidad de sistemas solicitados (*size*) y la cantidad de tareas en cada sistema (*n*). Luego se listan los sistemas, con las tareas que los componen. De cada sistema (***S***) se muestra su número de orden dentro del archivo (*count*), su factor de utilización (*U*) y su hiperperíodo (*mcm*). Si el hiperperíodo no fuera representable en el formato numérico utilizado, se muestra el mayor valor representable con 18 dígitos.

En cada línea dentro de la etiqueta `<S> … </S>` se muestran los valores de los parámetros para cada tarea. Los parámetros de generación opcional que no hayan sido solicitados se muestran con valor cero.

Dentro de cada sistema las tareas están ordenadas de menor a mayor por períodos o por vencimientos, según lo solicitado en la configuración. Para contemplar los casos de conjuntos de tareas donde el D_i es arbitrario respecto del T_i, si se pide ordenar por vencimiento, se toma $D_i - J_i$ (como lo realizado en [10]).

A continuación se incluye un fragmento del archivo principal de salida en formato *xml* para 10 sistemas de 3 tareas con factor de utilización del 80%:

```
<?xml version="1.0" ?>
<Set size=" 10" n=" 3" >
  <S count=" 1" U="80.0" mcm=" 353580" >
    <i nro=" 1" C=" 15.55" BC="  7.34" AC=" 11.55" T=" 60" D=" 60"
B="  0.37" J=" 1" Of=" 0" Co=" 0" />
    <i nro=" 2" C=" 27.87" BC=" 11.36" AC=" 13.50" T=" 71" D=" 71"
B="  1.35" J=" 0" Of=" 0" Co=" 0" />
    <i nro=" 3" C=" 12.28" BC="  5.86" AC=" 12.19" T=" 83" D=" 83"
B="  0.14" J=" 2" Of=" 0" Co=" 0" />
  </S>
```

El tipo de archivo *txt* muestra los mismos datos, pero reemplazando las etiquetas *xml* por el caracter ":". A continuación se presenta el ejemplo anterior en formato *txt*:

```
10: 3
 1: 80.0: 353580
   1: 15.55:  7.34: 11.55: 60: 60:  0.37: 1: 0: 0
```

```
2: 27.87: 11.36: 13.50: 71: 71:  1.35: 0: 0: 0
3: 12.28:  5.86: 12.19: 83: 83:  0.14: 2: 0: 0
```

El tipo de archivo en formato *vert* (vertical) contiene solamente el hiperperíodo de cada sistema, su factor de utilización y el período y duración de cada tarea en un formato de tipo lista, dejando una línea vacía entre cada grupo de tareas.

El nombre de estos archivos se construye con el prefijo *rtts_* seguido del factor de utilización solicitado, un guión bajo y la cantidad de tareas en cada sistema. El nombre del archivo de respaldo tiene además el sufijo *-bkp*. La nomenclatura elegida tiene por finalidad facilitar la programación de un paquete de pruebas, cuya salida quedaría en archivos separados.

El formato del archivo de salida de las tareas aperiódicas muestra, en cada línea, el número de tarea dentro de la lista generada, su tiempo de arribo y su duración.

En *xml* sería:

```
<?xml version="1.0" ?>
  <Aperiodics>
  <J count="número_de_tarea" T="tiempo_de_arribo" A="duración" />
  …
  </Aperiodics>
```

En formato *txt* el orden de los parámetros es el mismo, reemplazando las etiquetas *xml* por el carácter ":". La nomenclatura de los archivos de salida se construye con el prefijo *apt_* seguido del máximo tiempo entre arribos configurado y el valor de λ utilizado.

4 Resultados Experimentales

4.1 Cálculo de la Disparidad de los U_i

De la misma manera que lo presentado por Bini en [8], las diferencias entre los factores de utilización de las tareas de un sistema deben ser pequeñas. Esto garantiza una distribución uniforme en la utilización del recurso entre las tareas. Cuanto mayor es la cantidad de tareas en el sistema, mejor distribuidos estarán los factores de utilización.

Sea ΔU la diferencia entre el máximo y el mínimo U_i en cada conjunto:

$$\Delta U = \frac{\max_{i=1}^{n} U_i - \min_{j=1}^{n} U_j}{U} \quad \text{con } i \neq j \tag{1}$$

para observar la distribución de los factores de utilización de los conjuntos generados, se calcula la función de densidad de probabilidad de los ΔU.

Más adelante en esta sección, se grafica la función de densidad de probabilidad de ΔU, para observar dónde se centra la mayor parte de la generación de los sistemas.

4.2 Experimentos

Para evaluar la performance desde el punto de vista de la calidad de los conjuntos de tareas, de la misma manera que en el trabajo de Bini ([8]), se realizaron generaciones de conjuntos de distintos tamaños y distintos factores de utilización. En este caso, se realizaron exhaustivas corridas del generador para 100.000 sistemas para cada factor de utilización en conjuntos de 10, 20 y 50 tareas, variando el factor de utilización de diez en diez. En total fueron 30 ejecuciones del generador, logrando así tres millones de sistemas distintos, planificables por *RM* excepto los de 50 tareas con factor de utilización del 100%. En este último caso no fue posible generar conjuntos de tareas que pasaran los tests de planificabilidad por *RM* o *DM*, debido a que estas disciplinas no siempre pueden planificar sistemas para valores más altos que el 88% del factor de utilización ([15]).

La calidad del conjunto de tareas está determinada por el valor de ΔU (Ecuación (1)). Este valor precisa la diferencia existente entre el mayor y menor factor de utilización de las tareas de cada sistema. Consecuentemente, para cada sistema generado se obtuvo el valor de ΔU y se calculó su función de densidad de probabilidad.

En los siguientes gráficos se observará que a medida que se incrementa el tamaño de cada conjunto, los valores de ΔU son menores y menos dispersos, lo cual indica una mejor distribución del factor de utilización entre las tareas de cada sistema.

A continuación se presentan los gráficos para cada factor de utilización. En el eje x se representan los ΔU calculados, y en el eje y su función de densidad de probabilidad. En cada gráfico, el ΔU para conjuntos de 10 tareas alcanza su máximo alrededor de 0,24. Esto indica que existe una diferencia centrada en el 24%. Para conjuntos de veinte tareas, ΔU tiene valores centrados en 0,15 y para conjuntos de 50 tareas, ΔU está centrada en 0,08.

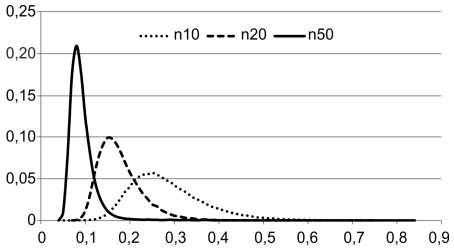

Fig. 2. *U*= 10%. Conjuntos de 10, 20 y 50 tareas

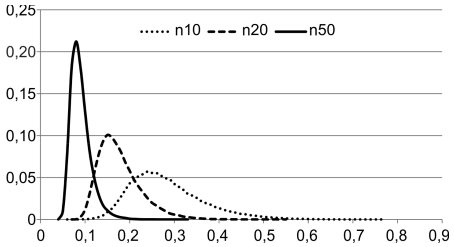

Fig. 3. *U*= 20%. Conjuntos de 10, 20 y 50 tareas

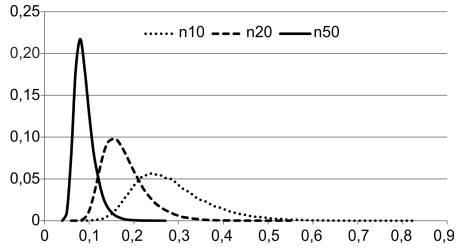

Fig. 4. *U*= 30%. Conjuntos de 10, 20 y 50 tareas

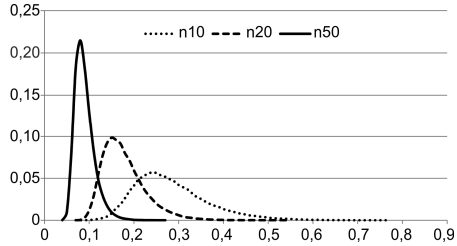

Fig. 5. *U*= 40%. Conjuntos de 10, 20 y 50 tareas

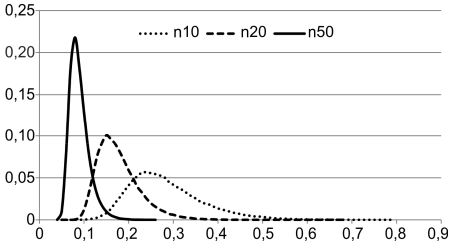

Fig. 6. *U*= 50%. Conjuntos de 10, 20 y 50 tareas

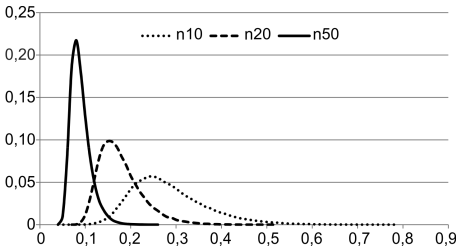

Fig. 7. *U*= 60%. Conjuntos de 10, 20 y 50 tareas

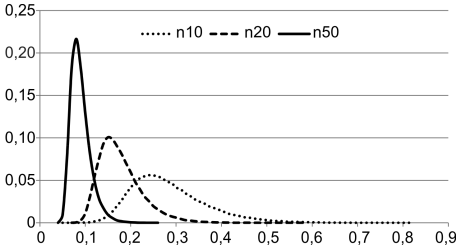

Fig. 8. *U*= 70%. Conjuntos de 10, 20 y 50 tareas

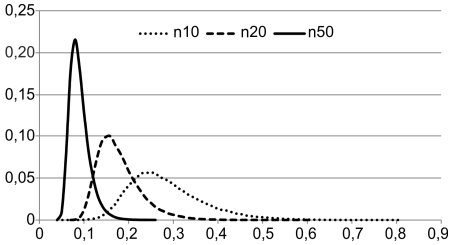

Fig. 9. *U*= 80%. Conjuntos de 10, 20 y 50 tareas

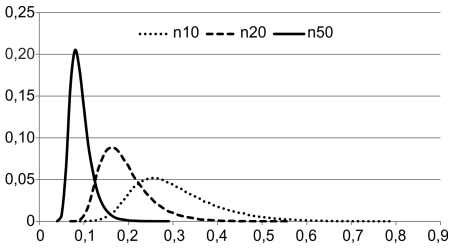

Fig. 10. *U*= 90%. Conjuntos de 10, 20 y 50 tareas

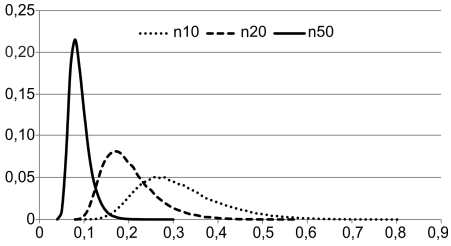

Fig. 11.*U*= 100%. Conjuntos de 10, 20 y 50 tareas

Para evaluar la calidad de los tiempos de arribo de las tareas aperiódicas se realizaron generaciones de tareas con distribución exponencial, con el máximo *IAT* igual a 100, 200, 500 y 1000, con λ igual a la tasa de 1, 2, 3 y 4 sobre el intervalo elegido. Se realizaron en total doce generaciones, con más de un millón de tareas cada una. Los gráficos se generaron a partir de la frecuencia relativa observada normalizada de arribos (*FRONA*) en el intervalo (eje *y*) respecto del tamaño del *IAT* (eje *x*) normalizado. Se incluye en cada gráfico todos los valores de λ utilizados para cada tamaño de intervalo entre arribos.

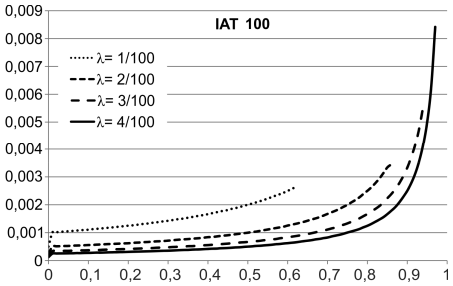

Fig. 12. *FRONA* en el intervalo de 100.

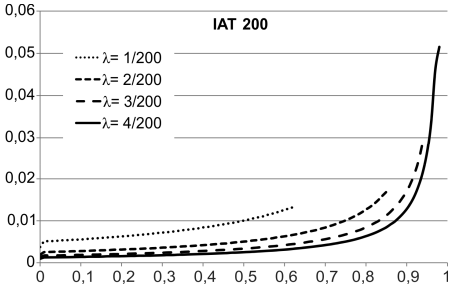

Fig. 13. *FRONA* en el intervalo de 200.

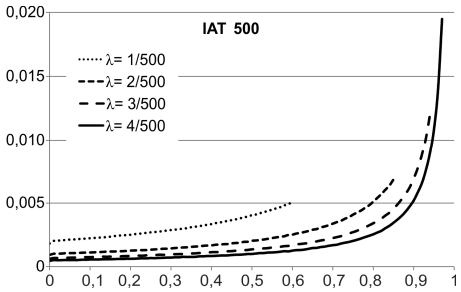

Fig. 14. *FRONA* en el intervalo de 500.

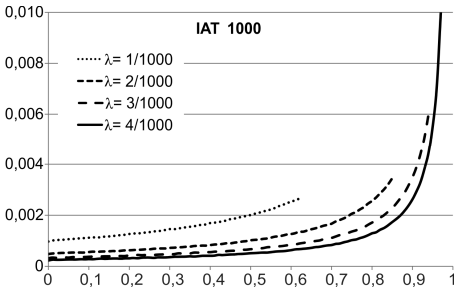

Fig. 15. *FRONA* en el intervalo de 1000.

5 Conclusiones y Trabajos Futuros

Se rediseñó, mejoró y se agregaron nuevas funcionalidades al software de generación de tareas presentado en [16]. Estos cambios aportaron una mayor flexibilidad, para que se pueda adaptar a una mayor variedad de subdisciplinas de *STR*.

Su flexibilidad se basa en la posibilidad de definir su configuración según la aplicación que lo va a utilizar, dado que cada una de ellas suele tener requerimientos diferentes. Además verifica que los sistemas periódicos sean planificables por *RM, DM* o *EDF* y puede generar conjuntos de tareas aperiódicas. Este software también garantiza una distribución pareja de la utilización del recurso entre las tareas periódicas, como se demostró con los gráficos (Figuras de la 2 a la 11). Permite generar tareas aperiódicas con distribución exponencial y uniforme. Es sencillo de utilizar, genera millones de sistemas en un tiempo razonablemente pequeño, fue escrito en *ADA 2005* y dispone de una interfaz gráfica para Windows.

Quedan como trabajos futuros: permitir indicar los nombres y ubicaciones de los archivos de salida, traducir al idioma inglés para facilitar su uso por personas no hispanoparlantes, construir interfaz gráfica independiente del sistema operativo y verificar la planificabilidad de los sistemas incluyendo los tiempos de bloqueo.

Referencias

[1] Chaitanya Belwal and Albert M.K. Cheng, "An Extensible Framework for Real-time Task Generation and Simulation using Object and Reflection Oriented Programming," University of Houston, Houston UH-CS-11-04, June 8, 2011 2011.

[2] C. L. Liu and James W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *Journal of the ACM,* vol. 20, pp. 46-61, 1973.

[3] N. C. Audsley, Alan Burns, M. F. Richardson, K. Tindell, and A. J. Wellings, "Applying New Scheduling Theory to Static Priority Preemptive Scheduling," *Software Engineering Journal,* vol. 8, pp. 284-292, 1993.

[4] W. Feng and J. W. S. Liu, "An extended imprecise computation model for time-constrained speech processing and generation," in *Real-Time Applications, 1993., Proceedings of the IEEE Workshop on*, 1993, pp. 76-80.
[5] Hakan Aydin, Rami G. Melhem, Daniel Mosse, and Pedro Meja-Alvarez, "Optimal Reward-Based Scheduling for Periodic Real-Time Tasks," *IEEE Transactions on Computers,* vol. 50, pp. 111-130, 2001.
[6] Joel Goossens and Christophe Macq, "Limitation of the Hyper-Period in Real-Time Periodic Task Set Generation," in *9th International Conference on Real-Time Systems - Embedded System* Paris, France, 2001, pp. 133-148.
[7] AdaCore, "Ada 2005 - http://libre.adacore.com/libre/," 2005 ed, 2010.
[8] Enrico Bini and Giorgio C. Buttazzo, "Biasing Effects in Schedulability Measures," presented at the Proceedings of the 16th Euromicro Conference on Real-Time Systems, 2004.
[9] Robert I. Davis and Alan Burns, "Priority Assignment for Global Fixed Priority Pre-emptive Scheduling in Multiprocessor Real -Time Systems.," University of York, York 2009.
[10] Robert Davis and Alan Burns, "Response Time Upper Bounds for Fixed Priority Real-Time Systems," presented at the The 29th IEEE Real-Time Systems Symposium, Barcelona, Spain, 2008.
[11] Rodrigo M. Santos, José M. Urriza, Jorge Santos, and Javier D. Orozco, "Heuristic use of Singularities for On-Line Scheduling of Real-Time Mandatory/Reward-Based Optional Systems," in *14th Euromicro Conference on Real-Time Systems*, Vienna, Austria, 2002, pp. 103-110.
[12] Rodrigo M. Santos, José M. Urriza, Jorge Santos, and Javier D. Orozco, "Diagramación on-line de sistemas de tareas de tiempo real periódicas mandatorias duras/opcionales basadas en recompensas con factor de depreciación," in *31 JAIIO AST2002*, Santa Fe, Argentina, 2002, p. 148.
[13] José M. Urriza, "Factibilidad de Sistemas de Tiempo Real con Requerimientos Heterogéneos," Doctor, Departamento de Ingeniería Eléctrica y Computadoras, Universidad Nacional del Sur, Bahía Blanca, 2008.
[14] John English. (2000, *JEWL (John English's Window Library)*. Available: http://www.it.bton.ac.uk/staff/je/jewl/
[15] John P. Lehoczky, Lui Sha, and Ye Ding, "The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior," Department of Statistics, Carnegie-Mellon, Pitsburg, USA, Internal Report1987.
[16] Gabriela Olguín, Laura Biscayart, and José M. Urriza, "Generador de Conjuntos de Tareas para Simulación en Sistemas de Tiempo Real," presented at the JAIIO 39 - Jornadas de Informática Industrial (JII) 2010, Buenos Aires, 2010.