Generación Automática de Entornos de Simulación con Interfaces Inteligentes

Alfonseca, M., García, F., de Lara, J., Moriyón, R.
{Manuel.Alfonseca, Federico.Garcia, Juan.Lara, Roberto.Moriyon}@ii.uam.es
Universidad Autónoma de Madrid, Departamento de Ingeniería Informática

Resumen

Este artículo describe cómo generar automáticamente entornos de simulación con interfaces de usuario inteligente, a partir de la descripción del modelo de simulación. Para ello hemos utilizado nuestro propio lenguaje de simulación continua orientado a objetos, OOCSMP, un compilador realizado para este lenguaje, capaz de generar código en C++ para las simulaciones y la interfaz, y un gestor de tareas de usuario, ATOMS, también construido por nosotros, capaz de ofrecer facilidades run-time a los usuarios de las aplicaciones.

Palabras clave Simulación continua, orientación a objetos, educación por ordenador, interfaces inteligentes, modelos de tareas.

1. Introducción.

La simulación de sistemas [1] es una herramienta estándar de las aplicaciones multimedia y la educación por ordenador. En la simulación continua, la descripción del modelo se realiza mediante un conjunto de ecuaciones diferenciales-algebraicas. Estos modelos de simulación continua se programan normalmente mediante un lenguaje de propósito general, o mediante uno de propósito especial. El lenguaje que usamos para generar nuestras aplicaciones educativas, OOCSMP, es una extensión del lenguaje CSMP de IBM, que, entre otras cosas, le hacen orientado a objetos. En artículos previos [2], se han descrito características del OOCSMP que lo hacen muy útil para la generación de cursos educativos para internet en distintos dominios tales como ecología [3], gravitación [4], electrónica [5], etc...

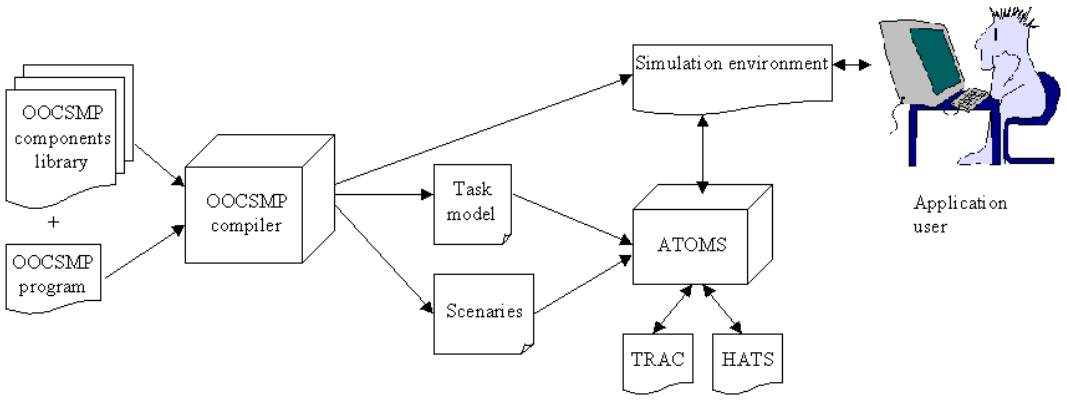
Sin embargo, los sistemas de construcción de entornos de simulación suelen producir interfaces que no permiten explotar al máximo las capacidades de dichos entornos, ya que son producidos de manera automatizada y no pueden ajustarse ni a las características del sistema simulado ni a los objetivos que los usuarios persiguen, disminuyendo así el potencial de dichos sistemas. En este artículo se presenta un trabajo cuyo objetivo es que los usuarios se beneficien plenamente de los sistemas de simulación generados, mediante la modelización de las tareas de las interfaces de usuario generadas.

Las interfaces generadas por los entornos de simulación más extendidos actualmente tienen un problema fundamental: las interfaces resultan poco atractivas a los usuarios desde el punto de vista de la riqueza de la interacción con el sistema. Las interfaces habituales no cubren el hueco existente entre las tareas conceptuales que los usuarios conocen y cómo esas tareas se deben realizar con las aplicaciones que ellos utilizan. Así, dichas interfaces no ofrecen soporte para técnicas avanzadas de interacción con el usuario, tales como automatización de tareas repetitivas o servicios de ayuda al usuario. El primer tipo de soporte permite, cuando el usuario está realizando

secuencias de tareas similares, inferir las siguientes acciones y realizarlas automáticamente, ahorrando así esfuerzos al usuario. Dentro del segundo tipo de soporte, pueden destacarse la generación automática de ayuda sobre las tareas de la interfaz, y la visualización del proceso de realización de *escenarios* -tareas comunes del tipo 'how to...'-. El primer servicio se encargaría de guiar a los usuarios en cómo realizar las tareas que la aplicación provee, mientras que el segundo permitiría enseñar al usuario cómo realizar, en la interfaz, acciones que tendrían determinados efectos sobre los parámetros y comportamientos del sistema que se está simulando.

2. Arquitectura propuesta.

Nuestro compilador del lenguaje OOCSMP permite la generación de código C++ que utiliza el toolkit AMULET [6], desarrollado por la Universidad de Carnegie Mellon. Este código contiene la rutina de simulación y la interfaz para que el usuario interaccione con el problema. Además el compilador genera un modelo de tareas de la interfaz, de forma que ATOMS (Advanced Task Oriented Management System), un sistema de gestión de tareas de usuario, sea capaz de ofrecer ayuda automática para la interfaz, ejecución automática de escenarios y soporte para macros. Estos conceptos se



explicarán en la sección 2.2.

Figura 1. Arquitectura propuesta.

2.1. El lenguaje OOCSMP.

El OOCSMP surge como una extensión orientada a objetos del lenguaje de simulación continua CSMP de IBM. También se han incorporado otras posibilidades, como la resolución de ecuaciones en derivadas parciales, mediante elementos o diferencias finitas, con posibilidad de mezclar ambos métodos en la resolución de una ecuación.

Además, el compilador genera una interfaz flexible para el problema, desde la que se pueden añadir nuevos objetos a la simulación, modificar el contenido de los vectores de objetos, modificar parámetros de la simulación (lo que permite responder a preguntas del tipo *what if...?*)

El compilador también ofrece la elección y combinación de diversos formatos de salida de datos muy flexibles, tales como listados, gráficos lineales, con o sin

animación, gráficos 3D, icónicos e interactivos con la estructura gráfica de las ecuaciones, que son muy útiles, por ejemplo, a la hora de representar tanto circuitos analógicos como digitales.

2.2. El modelo de tareas de ATOMS.

El modelo de tareas de una interfaz gráfica interactiva es una representación jerárquica, utilizando un lenguaje declarativo, de las tareas que el usuario puede realizar con cierta aplicación. Esta representación consiste básicamente en la definición de un conjunto de tareas y las relaciones entre ellas, de modo similar a los símbolos y las reglas de las gramáticas usadas en el tratamiento del lenguaje natural. ATOMS [7][8] es un sistema de gestión de tareas de usuario que se encargará de interpretar el modelo de tareas de usuario y ofrecer servicios adicionales en tiempo de ejecución.

El modelo que representa las tareas que el usuario puede realizar con la aplicación es generado por el compilador, junto al código de la simulación y el de la interfaz. Dichas tareas pueden ser *atómicas*, que representan acciones que el usuario puede realizar directamente interaccionando sobre la aplicación, y que por tanto incluyen una descripción declarativa de a qué interacción se refieren, o *compuestas*, que definen tareas conceptuales de más alto nivel.

Las tareas atómicas se corresponderían con los símbolos terminales de las gramáticas, mientras que las tareas compuestas serían sus símbolos no terminales. Un ejemplo de tarea de tipo atómico sería confirmar los valores de un cuadro de diálogo utilizando el botón de etiqueta 'Ok', mientras que una de alto nivel sería la de hacer que la masa del Sol tomara cierto valor. En la sección 3 se dará un ejemplo de la definición de una parte del modelo de tareas de una interfaz.

Cada tarea, tanto si es atómica como si es compuesta, necesita de una descripción de su semántica, y puede llevar parámetros asociados, que funcionan como información contextual de la tarea. Además de las tareas, el modelo tiene que incluir las relaciones existentes entre dichas tareas, lo cual se hace mediante la definición de reglas. Cada regla relaciona una tarea compuesta dada, que cumple el papel de un símbolo no terminal en la parte izquierda de la regla, con cierto conjunto de símbolos, terminales o no terminales, que actúan como parte derecha. Además, cada regla definida incluye otro tipo de información, entre la que destaca la siguiente:

- La relación de ejecución entre subtareas. Esta relación puede ser de 'orden estricto' (secuencia), de 'todas en cualquier orden' (and) o de 'alguna de ellas' (or).
- Descripción del flujo de parámetros entre tareas. Esta descripción determina cómo se obtienen los valores de los parámetros y cómo se transmiten entre tareas.
- Qué acciones pueden tener ejecución opcional y/o múltiple y, en este último caso, bajo qué condiciones.

ATOMS, en tiempo de ejecución, se encarga de, a partir de las interacciones del usuario y utilizando las reglas del modelo de tareas, actualizar el estado dinámico del sistema, lo que permite realizar un seguimiento de las tareas que el usuario está llevando a cabo. Asimismo, sobre ATOMS pueden montarse distintos servicios de valor añadido

que utilicen el estado de la aplicación para ofrecer distintos soportes automáticamente y sin coste adicional, como se verá en las siguientes secciones.

2.2.1. Macros.

Las interfaces generadas sobre ATOMS ofrecen soporte para un gestor de macros: TRAC [9]. TRAC es un sistema que permite automatizar secuencias repetitivas de tareas, actividad muy común en entornos de simulación, en los que los usuarios con frecuencia formulan sus hipótesis a base de repetir experiencias del tipo '*what if...*' y contrastar los resultados con experiencias similares realizadas con anterioridad.

TRAC realiza inferencia a dos niveles: inferencia de tareas repetitivas e inferencia de nuevos valores de contexto para las tareas a automatizar. El primer nivel se encarga de la detección de patrones de tareas repetidos a lo largo del historial de tareas recientemente realizadas por el usuario. Una vez ese primer nivel infiere cuáles deberían ser las siguientes acciones a realizar por el usuario, se lleva a cabo el segundo nivel de inferencia, esta vez sobre los parámetros con los que el usuario ha llevado a cabo las tareas a partir de las cuales se infirieron las futuras. Cuando ambos niveles de inferencia tienen éxito, en el sentido de que infieren completamente las siguientes acciones, entonces el sistema se ofrece al usuario a realizar automáticamente las acciones inferidas. Por supuesto, el usuario puede no aceptar dicho ofrecimiento e, incluso, puede descartar los valores de los parámetros inferidos y aportar él los deseados.

A diferencia de otros sistemas de automatización de tareas, TRAC no garantiza el acierto en sus predicciones, como ningún sistema basado en la inferencia, pero a cambio es enormemente más sencillo de manejar que aquellos. Esta gran sencillez se debe a que los usuarios no tienen que grabar previamente las macros y después ejecutarlas, ya que simplemente se limitan a aceptar o no las sugerencias que el sistema ofrece.

Otra importante ventaja es el hecho de que TRAC razone sobre tareas de alto nivel, ya que ofrece una gran independencia del procedimiento seguido por el usuario para llevar a cabo sus tareas, pues puede darse cuenta de que el usuario está realizando las mismas tareas siguiendo distintos caminos.

Otra ventaja adicional que soportan las interfaces generadas sobre ATOMS es la facilidad de *undo múltiple orientado a tareas* –cuando lo normal es la inexistencia de undo o, a lo sumo, undo de acciones simples-. Esta facilidad cobra una importancia vital cuando las interfaces permiten automatizar tareas, pues podría suceder que las tareas que se han realizado automáticamente no son las que el usuario esperaba, y por tanto deberían poder ser deshechas.

2.2.2. Ayuda.

En multitud de ocasiones sucede que las personas que utilizan entornos de simulación, sobre todo cuando lo hacen con fines pedagógicos, no consiguen obtener de ellos el rendimiento deseado, porque no tienen a su lado un tutor que complemente el poder de experimentación que el entorno de simulación les facilita. Esto se debe a que muchas veces ellos no saben cómo facilitar los parámetros al sistema, o qué parámetros

han de proveer. HATS (Help for ATOMS Task System), el subsistema de ayuda de ATOMS, está enfocado a resolver el primero de los problemas.

HATS es capaz de guiar al usuario a través del aprendizaje de la interfaz, indicando qué pasos se han de dar para hacer algo o qué alternativas tiene el sistema disponibles en cada paso. Esto ayuda a los usuarios a, por una parte, aprender la manera de llevar a cabo ciertas tareas de *manera genérica*, es decir, con los parámetros que el usuario desee (cambiar las propiedades solares) y, por otra, a llevar a cabo *instancias* de dichas tareas, es decir, llevar a cabo las mismas tareas pero con determinadas restricciones (por ejemplo, dar a la masa del sol la mitad de su valor actual).

HATS, por estar apoyado en una descripción jerárquica de tareas, permite que las explicaciones puedan darse con distintos niveles de abstracción, según interese a cada usuario. Esto, por sí mismo, es un gran avance frente a las ayuda en forma de recetas que ofrecen los actuales sistemas basados en hipertexto o hipermedia. Este enfoque ya fue utilizado por Pangoli y Paternó [10].

Por otra parte, al estar construido con el soporte de ATOMS, la ayuda que se facilita se adapta dinámicamente según el usuario va completando las tareas que HATS le indica, con lo cual se evita el saturar al usuario con multitud de explicaciones que no son necesarias en un cierto momento, sino varios pasos más adelante. Según el usuario vaya completando las tareas con alguna de las alternativas que se le ofrecen, los mensajes de ayuda son actualizados teniendo en cuenta la alternativa seleccionada.

Por último, otra de las características más importantes de HATS es la posibilidad de que se ofrezcan referencias gráficas a los usuarios, en forma de parpadeos de objetos, a fin de permitir localizar instantáneamente los elementos gráficos a los que las explicaciones de la ayuda se refieren, o los objetos con los que el usuario debe interaccionar. Para ver cómo los mensajes de ayuda son generados a partir de las descripciones de las tareas, y muchos más detalles de HATS, acudir a [11].

2.2.3. Escenarios.

Como se indicó anteriormente, uno de los mayores problemas de los usuarios de entornos de simulación es el no saber cómo realizar las tareas que se proponen hacer. A veces, aún sabiendo cómo hacer dichas tareas, no saben qué valores suministrar al sistema para que este se comporte de tal manera que sus experimentaciones ofrezcan resultados útiles para su aprendizaje.

Los escenarios son conjuntos de tareas que el módulo de emulación de tareas que contiene ATOMS es capaz de ejecutar automáticamente sin intervención del usuario. La característica fundamental de este módulo es que es capaz de realizar las tareas del mismo modo que si las llevara a cabo el usuario, es decir, un cursor simulado aparece en la interfaz manipulando sus elementos de la misma manera en que el usuario debería actuar para llevar a cabo esas tareas.

Los escenarios se definen mediante un lenguaje de programación declarativo. Los escenarios deben ser definidos por el desarrollador del modelo de simulación mediante extensiones de OOCSMP, que son compiladas por el compilador de este lenguaje para crear sentencias equivalentes en el citado lenguaje declarativo,

interpretadas por ATOMS. De esta forma el usuario aprende mediante observación a manejar la interfaz y a establecer valores significativos a los parámetros de la simulación.

El listado 4, explicado en la siguiente sección, ofrece un ejemplo de cómo OOCSMP da soporte para incluir la declaración de escenarios.

3. Un ejemplo : simulación del sistema solar interior.

Supongamos que se quiere preparar una simulación del sistema solar interior. Un programa OOCSMP podría ser el siguiente:

```
TITLE GRAVITATION

* ************************************************************
* Universal data                                             *
* ************************************************************

DATA G:=0.00011869, PI:=3.141592653589793
* Sun data
DATA MS:=332999
INCLUDE "PLANET.CSM"

* ************************************************************
* Actual planets                                             *
* ************************************************************

Planet Mercury("Mercu",0.055271,-0.3871, 0,     2.078, -9.892, 7.004)
Planet Venus ("Venus",0.81476,  0.7233, 0,     0.051,  7.39,  3.394)
Planet Earth ("Earth",1,        0,     1,     -6.2899, 0.107, 0    )
Planet Moon  ("Moon", 0.01235,  0,     0.9975,-6.0783, 0.107, 0   )
Planet Mars  ("Mars", 0.10734, 1.5233, 0, 0.476,  5.071, 1.85)
Planet Jupiter("Jupit",317.94,   0,     -5.2028, 2.754,  0.131, 1.308)
Planet InnerSys :=Mercury,Venus,Earth,Moon,Mars,Venus, Jupiter
InnerSys.STEP()
InnerSys.ACTION(InnerSys)

* ************************************************************
* Time intervals and other data                              *
* ************************************************************

TIMER delta:=.001, FINTIM:=2, PRdelta:=.1, PLdelta:=.01
METHOD RKSFX
```

Listado 1. Modelo OOCSMP para la simulación del sistema solar interno.

Donde el módulo Planet.csm contiene la clase Planet que modeliza el comportamiento de un planeta, así como las interaccioners de un planeta con el resto (llamada al método ACTION). El código de la clase Planet se detalla a continuación:

```
* ************************************************************
* Definition of Planet class                                 *
* ************************************************************

CLASS Planet {
 NAME name
 DATA M, X0, Y0, XP0, YP0, FI
 INITIAL
```

```
FIR:=FI*PI/180
CFI:=COS(FIR)
SFI:=SIN(FIR)

* ************************************************************
* Calculations for this planet                                *
* ************************************************************

DYNAMIC
* Distance to the Sun
R2   := X*X+Y*Y
R    := SQRT(R2)
Y1   := Y*CFI
Z    := Y*SFI
* Mutual influences
 * The Sun on this planet
APS := G*MS/R2/R
 * This planet on the Sun
ASP := G*M/R2/R
XPP := -(ASP+APS)*X
YPP := -(ASP+APS)*Y
XP  := INTGRL(XP0,XPP)
YP  := INTGRL(YP0,YPP)
X   := INTGRL(X0,XP)
Y   := INTGRL(Y0,YP)

* ************************************************************
* Mutual actions of two planets                               *
* ************************************************************

ACTION Planet P
* Distance to another planet
DPP2 := (P.X-X)*(P.X-X)+(P.Y-Y)*(P.Y-Y)+(P.Z-Z)*(P.Z-Z)
DPP  := SQRT(DPP2)
* Influences
 * The other planet on the Sun
ASP1 := G*P.M/P.R2/P.R
 * The other planet on this planet
APP1 := G*P.M/DPP2/DPP
* Coordinate conversion
Y2 := P.Y*COS(P.FIR-FIR)
* Actual action of the planet
XPP += APP1*(P.X-X) - ASP1*P.X
YPP += APP1*(Y2-Y) - ASP1*Y2

* ************************************************************
* Other data                                                  *
* ************************************************************

PRINT R
PLOT Y,X
FINISH R=.0001}
```

Listado 2. Código de la clase Planet.

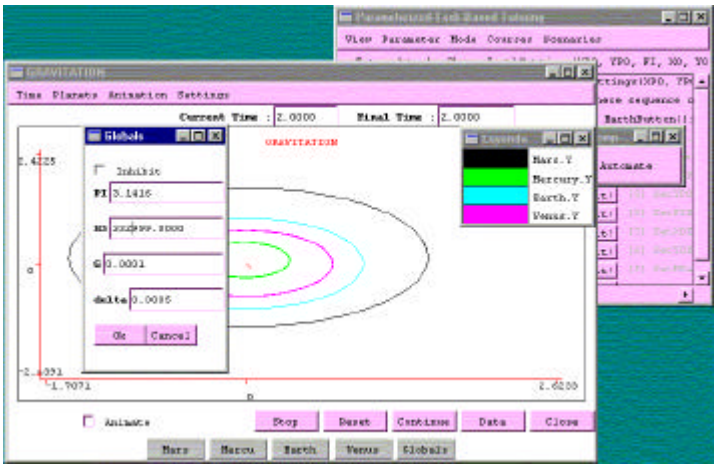


Figura 1: Una pantalla del ejemplo expuesto.

El usuario puede cambiar los parámetros de los objetos que intervienen en la simulación (se genera un botón para cada objeto), variables globales, parámetros de simulación, añadir y quitar objetos a/de los vectores existentes. También se pueden añadir nuevos objetos (en este caso planetas) a la simulación (botón "nuevo objeto", se rellenan los parámetros y se añade al vector "InnerSys"), tras lo cual aparecerá un nuevo botón en la interfaz para poder modificar las propiedades de ese objeto durante la simulación.

El compilador de OOCSMP generará, como se ha dicho, el modelo de tareas de la interfaz. En la siguiente figura se puede ver qué tareas (atómicas y compuestas) y qué reglas define el compilador para la tarea de cambiar el valor de la masa solar:

```
//Atomic task for global settings button
ATOMIC GlobalsButton
BUTTON "Globals"
DESCRIPTION Begin global properties editing by clicking on Globals button
END

//Atomic task for global settings menu option
ATOMIC GlobalsMenu
MENU "Globals"
DESCRIPTION Begin global properties editing by selecting Globals menu item
END

// Bring up the globals dialog
COMPOSED ChangeGlobals
DESCRIPTION Brings up the globals' dialog
END

// The dialog may be poped-up following these two methods:
RULE ChangeGlobals
SUBTASKS GlobalsButton AS Button GlobalsMenu AS Menu
SEQUENCING OR
END


//Accept global values
ATOMIC OkGlobalsButton
BUTTON "Ok"
DIALOG_TITLE == "Globals"
DESCRIPTION Accept new summass value
```

```
END

// Modify the text input box
ATOMIC SetSM
TEXT_BOX  == "SM"
DIALOG_TITLE == "Globals"
PARAMETER SM = TEXT
DESCRIPTION Sets SM property value
END

// High-level task of modify sun mass value
COMPOSED ChangeSMGlobalSetting
DESCRIPTION Allows the SM property global setting to be changed
END

RULE ChangeSMGlobalSetting
SUBTASKS ChangeGlobals AS Change SetSM AS SetValues OkGlobalsButton AS Accept
SEQUENCING SEQ
MULTIPLE SetValue
PARAMETER SM = SetValues.Params.SM
END
```

Listado 3. Parte del modelo generado para un conjunto de tareas.

Como se observa, en el lenguaje se indica que la primera acción para modificar la masa del sol se puede realizar tanto apretando el botón 'Globals' –constantes globales- como mediante la opción equivalente del menú. Igualmente, la tarea de modificar sus parámetros necesita que se complete la secuencia de acciones señaladas: seleccionar la opción de modificar los valores del sol independientemente del procedimiento utilizado-, introducir el valor deseado en el campo destinado a tal efecto -acción que puede realizarse una o más veces, para permitir la corrección de errores al teclear-, y confirmar la operación. En cuanto a los parámetros, el único de esta tarea es el nuevo valor de la masa solar, que se obtiene a partir del objeto sobre el que el usuario ha interaccionado al teclear el nuevo valor y se transforma en el parámetro de la tarea de nivel superior sin sufrir modificación alguna.

Es posible preparar situaciones interesantes en la simulación, tales como aumentar la masa del Sol, añadir un nuevo planeta, etc. Para ello se usa el comando "\" de OOCSMP, seguido de las acciones que se quieren preparar.

```
\ APOLLO
Planet Apollo ("Apolo",1957E-14, 0, 1.4849,-4.253,  2.915, 6.4  )
Planet InnerSys :=Mercury,Venus,Earth,Moon,Mars,Venus, Apollo, Jupiter
\ SUNMASS
MS := 400000
Mars.M := 0.15
```

Listado 4. Código OOCSMP correspondiente a la definición de un escenario.

En el listado anterior se han preparado dos escenarios, uno en el que se añade el cometa Apolo a la simulación del sistema, y otro en el que se aumenta la masa del Sol y del planeta Marte. El compilador de OOCSMP en este caso, generaría dos ficheros (APOLLO y SUNMASS) con las tareas (del modelo de tareas generadas anteriormente) que habría que realizar sobre la interfaz para realizar las acciones anteriores. Además la ejecución de estos escenarios se realizaría de forma animada, de forma que el usuario

puede ver qué interacciones hay que hacer sobre los elementos de la interfaz para llevar a cabo las acciones.

4. Conclusiones y trabajo futuro.

Las interfaces generadas han mostrado importantes ventajas respecto a su interactividad sobre las obtenidas con entornos similares comerciales. Además, es destacable la importancia de obtener buenas interfaces para entornos educativos, en los que el problema del usuario no debería ser cómo usar la interfaz, sino qué representa el modelo de simulación y cómo interpretar su funcionamiento.

Se han resuelto en parte los problemas de usabilidad y de aprovechamiento de la fase de aprendizaje, se ha facilitado el uso de la herramienta de simulación, todo ello a bajo coste, ya que el modelo de tareas que usa ATOMS es generado automáticamente, y con este modelo de tareas ATOMS puede complementar la aplicación con servicios de automatización y guía del usuario.

Como líneas de investigación futuras, destacan por su importancia dos. La primera es la de dotar a las interfaces de capacidades de modelización de usuarios, de tal manera que los sistemas generados sean capaces de darse cuenta de qué tipo de usuario está utilizando el sistema, para así personalizar su comportamiento. Las posibilidades de esta modelización incluyen desde la adaptación de los mensajes de guía en la ayuda hasta la relocalización de los elementos más comúnmente utilizados de la interfaz, según el tipo de usuario. La otra línea apunta hacia la adaptación de ATOMS hacia los sistemas distribuidos, como ya está enfocado el sistema de gestión de tareas ACCEL [12], más específicamente los entornos orientados a la WWW, dada la importancia de los entornos de simulación para la generación de cursos educativos a distancia.

5. Agradecimientos.

Este trabajo ha sido parcialmente subvencionado por el Plan Nacional de Investigación, proyecto número TIC96-0723-C02-01/02.

6. Bibliografía.

[1] Y.Monsef, "Modelling and Simulation of Complex Systems", SCS Int., Erlangen, 1997.
[2] Manuel Alfonseca, Estrella Pulido, Ricardo Orosco, Juan de Lara (1997). "OOCSMP : an object-oriented simulation language", ESS'97. Passau.
[3] Manuel Alfonseca, Rosa Carro, Juan de Lara, Estrella Pulido (1998). "Education in Ecology at the Internet with an Object-Oriented Simulation Language". EUROSIM'98, Helsinki.
[4] Manuel Alfonseca, Juan de Lara, Estrella Pulido (1998). "Semiautomatic Generation of Educational Courses in the Internet by Means of an Object-Oriented Continuous Simulation Language". ESM'98, Manchester.
[5] Manuel Alfonseca, Juan de Lara, Estrella Pulido (1998).“Generación semiautomática de cursos de Electrónica para Internet mediante un lenguaje de simulación continua orientado a objeto”. TAEE’98, Madrid (en prensa).
[6] OpenAMULET Home Page. http://www.scrap.de/html/amulet.htm.
[7] Rodríguez, P., García, F., Contreras, J. and Moriyon, R., (1997) "Parsing Techniques for User-Task Recognition," Proceedings of the Fifth International Workshop on Functional Modelling of Complex Technical Systems, Paris-Troyes.
[8] García, F., Rodríguez, P., Contreras, J. and Moriyon, R.(1998) “Gestión de Tareas de Usuario en ATOMS”, IV Jornadas sobre Tecnología de objetos, Bilbao. Pendiente de aceptación.
[9] García, F. and Moriyon, R., “TRAC: Task Repetition And Completion”. En preparación.
[10] Pangoli, S. and Paternó, F. (1995) "Automatic Generation of Task-oriented Help," Proceedings UIST'95, Pittsburgh, ACM Press, New York, pp.181-187.

[11] García, F., Contreras, J. Rodríguez, P. and Moriyon, R., (1998)"Help generation for task based applications with HATS", IFIP Working Conference on Engineering for Human-Computer Interaction, Creta (en prensa).
[12] Zeiliger, R. and Kosbie, D. (1997) "Automating Tasks for Groups of Users: A System-Wide "Epiphyte" Approach", in INTERACT'97, ed. S. Howard, J. Hammond and G. Lyndgaard, Chapman & Hall Press, IFIP, Sydney.