UCShell. A Framework To Development Expert System

M. G. Lezcano, L. Fundora, H. López

***Abstract*— This paper describes the framework UCShell 3.0, a tool for developing expert system that consists of the following modules: a knowledge bases compiler, a knowledge bases editor and an inference engine that lets you work with imprecise rules and imprecise data (uncertainty). The inference engine uses backward and forward chaining searches.**

***Keywords*— Expert system, inference engine, uncertainty, knowledge bases.**

I. INTRODUCCIÓN

LOS sistemas expertos (SE) surgieron a mitad de la década del 60 del pasado siglo y contrariamente a lo que algunos piensan han mantenido su actualidad, incluso sus primeras aplicaciones exitosas están vigentes y sirven de referencia para nuevos desarrollos [1]. De esa primera época deben mencionarse los sistemas Dendral [2] y Mycin [3].

Los SE se utilizan en muchos campos, por ejemplo y solo por citar algunos, se pueden encontrar como auxiliares en: reconstrucción de terrenos [4], orientación vocacional [5], medicina [6], tratamiento de inundaciones [7], en la agricultura [8], etc.

En este artículo se presenta la versión 3.0 [9] del sistema UCShell, un ambiente para desarrollar sistemas expertos que puede usarse en todas las fases que forman parte del desarrollo de un SE. UCShell está constituido por los siguientes módulos:

- Un editor de bases de conocimientos (BC).
- Un compilador de bases de conocimientos.
- Un módulo de puesta a punto.
- Una máquina de inferencia (MI) que se enlaza con un mecanismo de interfaz con el usuario.

II. DESCRIPCIÓN GENERAL

El paradigma de representación del conocimiento utilizado por UCShell se denomina reglas de producción [10] y las bases del sistema están organizadas en forma de bloques que contienen distintas partes sobre las que actúa la máquina de inferencia (solo el bloque de acciones es obligatorio). Los bloques se identifican por una palabra reservada que se sitúa a su inicio, ellos son:

- Bloque de atributos externos. Comienza con la palabra External. En él se declaran todos los atributos que deberán conservar su valor cuando se haga un encadenamiento o cambio desde la BC actual hacia otra BC. Para hacer el cambio entre BC se utiliza la sentencia CHAIN.
- Bloque de atributos que se preguntan. En él se definen todos los atributos que obtendrán valores por medio de preguntas que se le hacen al usuario del sistema. Comienza con la palabra Asks.
- Bloque de reglas. Contiene todas las reglas de la base de conocimientos y se inicia con la palabra Rules. Es el último de los tres bloques opcionales.
- Bloque de acciones. Es el único bloque que no es opcional, comienza con la palabra Actions y constituye el punto de entrada de la base de conocimientos, o sea es el lugar donde la máquina de inferencia comienza su trabajo.

La sintaxis general de una base de conocimiento para UCShell se muestra en la Fig. 1. Pueden observarse los bloques opcionales (External, Asks y Rules) y el bloque obligatorio (Actions). Las bases terminan con la palabra reservada End seguida por un punto.

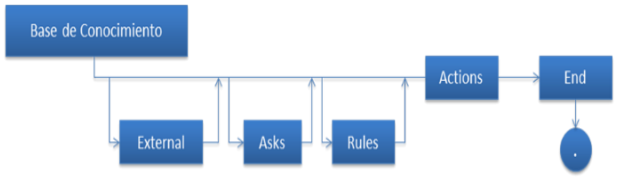

Figura 1. Sintaxis de las bases de conocimientos de UCShell.

La sintaxis para los atributos que se preguntan se aprecia en la Fig. 2. Su definición comienza con la palabra ASK seguida por el nombre del atributo y dos puntos (:) y cierra la parte obligatoria de la sintaxis el texto de la pregunta que se hará, encerrado entre comillas simples. Opcionalmente se pueden asociar opciones de respuestas y una explicación que deberá desplegarse en caso de que el usuario desee saber por qué el sistema le hace la pregunta en cuestión.

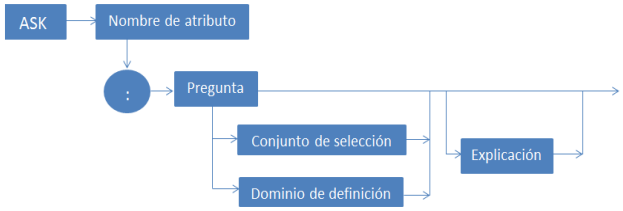

Figura 2. Sintaxis de los atributos que se preguntan (los lexemas aparecen en mayúsculas).

La Fig. 3 muestra la sintaxis para las reglas. La parte *Condición* contiene las premisas necesarias para que una regla sea verdadera y puede estar conformada por una o varias expresiones unidas por los operadores lógicos AND u OR, cada elemento tiene la forma:

<Expresión> Operador Relacional <Expresión>

M. G. Lezcano, Universidad Cooperativa de Colombia, sede Neiva. Universidad Central "Marta Abreu" de Las Villas (UCLV), Cuba, mateo.lezcanob@campusucc.edu.co

L. Fundora, Universidad Central "Marta Abreu" de Las Villas, Cuba, lissettfundora@gmail.com

H. López, Empresa de Tecnologías de la Información para la Defensa, La Habana, Cuba, hlopez@xetid.cu

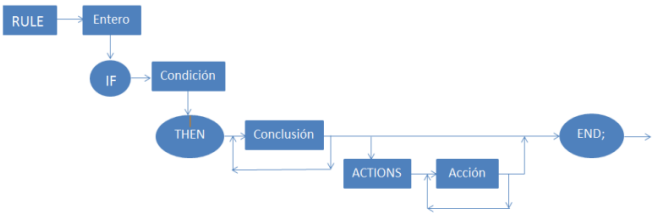

Figura 3. Sintaxis para las reglas de UCShell.

En la Fig. 4 se aprecia la sintaxis para la parte *Conclusión*, en ella se le asignan valores a los atributos usando el operador de asignación (:=).

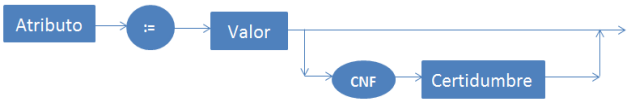

Figura 4. Sintaxis de la parte *Conclusión*.

El fin de una regla se señala con la palabra reservada *END* seguida de un punto y coma (*END;*). Las reglas pueden contener un bloque de acciones (*ACTIONS*) que se ejecutará cuando ellas se cumplan.

La Fig. 5 muestra un ejemplo de una base de conocimientos.

```
//Ejemplo de base de conocimientos
  EXTERNAL        /* Atributos que conservarán su
                     calor cuando se ejecute CHAIN*/
    Concluye      // Concluye conservará su valor
  ASKS            // Se definen las preguntas
    ASK Pre_1: 'Texto de la pregunta'
      DOMAIN 'Sí', 'No'
      BECAUSE 'Le pregunto esto porque …'
    ASK Pre_2: 'Texto de la pregunta'
  RULES           // Se definen las reglas
    RULE 1        // Regla 1
      IF
        Pre_1    = 'Sí'   AND
        Pre_2    > 20     AND
      THEN
        Concluye := 'Conclusión 1'
      ACTIONS
        DISPLAY 'He determinado que …'
    END;
  ACTIONS         // Órdenes a la MI
    FIND Concluye
    CHAIN ot.kbs /* Cambiar a la BC o.kbs, se
                    conservará el valor de Concluye
                    porque se declaró como externo*/
  END.
```

Figura 5. Base de conocimientos típica de UCShell.

En el bloque *EXTERNAL* de la Fig. 5 se ha declarado el atributo *Concluye* porque se desea conservar su valor cuando se realice un cambio hacia otra BC, en este caso cuando la MI ejecute la sentencia *CHAIN ot.kbs* que provocará que la inferencia se cambie de la base actual hacia la base *ot.kbs*.

Dentro del bloque *ASKS* se han definido dos atributos, *Pre_1* y *Pre_2*, que obtendrán valores a través de preguntas formuladas al usuario: para la pregunta Pre_1 se ha especificado un dominio de respuestas posibles a través de la palabra reservada *DOMAIN*, mientras la pregunta Pre_2 no tiene respuestas asociadas debido a que se espera que el usuario responda tecleando algún valor.

En el bloque *RULES*, se ha definido solo una regla y su bloque de acciones (comienza con la palabra reservada *ACTIONS*) mostrará un mensaje para lo cual se usa la sentencia *DISPLAY*.

Por último el bloque de acciones de la BC contiene algunas órdenes para la máquina de inferencia, esas mismas órdenes pueden estar en el bloque de acciones de las reglas. La orden más importante de esta base es *FIND Concluye* que ordena probar el objetivo *Concluye* mediante una búsqueda dirigida por objetivos.

El sistema UCShell permite dos tipos de comentarios que son iguales a los del lenguaje C, estos son: // y /* */, el primero permite poner comentarios de una sola línea mientras el segundo admite comentarios que pueden extenderse por varias líneas, ambos se han usado en la Fig. 5. Se recomienda tabular el contenido de las bases de conocimiento en la forma mostrada en la Fig. 5, lo que da más claridad a cada una de sus partes y facilita los mantenimientos.

III. LA INTERFAZ DE UCSHELL

El ambiente que le presenta el sistema UCShell a los usuarios (su interfaz) se muestra en la Fig. 6.

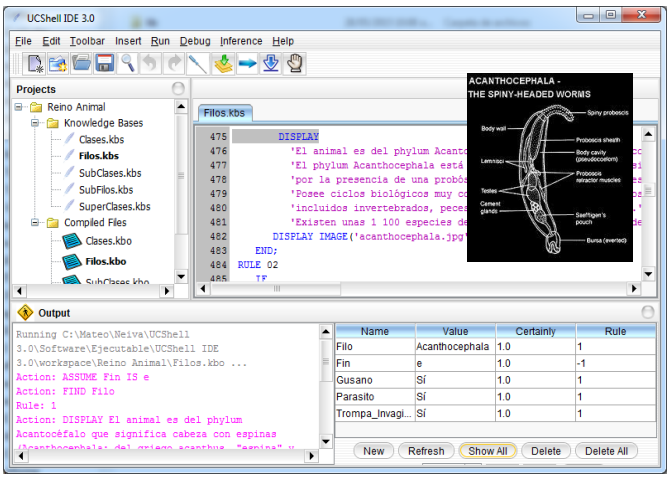

Figura 6. UCShell en ejecución paso a paso.

El panel izquierdo de la Fig. 6 contiene el árbol del proyecto que se está ejecutando, *Reino Animal* en este caso, un sistema experto utilizado para la clasificación de animales que fue hecho con UCShell.

El SE Reino Animal está compuesto por cinco bases de conocimientos, que son archivos de texto con extensión *kbs*: *Clases.kbs*, *Filos.kbs*, *SubClases.kbs*, *SubFilos.kbs* y *SuperClases.kbs*, los cuales se han compilado para obtener sus respectivos archivos objetos (extensión kbo). El sistema resalta, en negrita, la base actual (**Filos**) y su código fuente aparece en el panel de edición.

El panel de edición, localizado en la parte derecha de la figura 6, permite realizar cambios a las bases de conocimientos y dispone de las herramientas clásicas de

cualquier editor de programa.

La vista que se aprecia en la Fig. 6 muestra una instantánea de la ejecución, paso a paso, de la máquina de inferencia; en ese momento se había inferido el *Filo* del animal analizado (*Acanthocephala*) mediante la regla 1. El bloque de acciones de esa regla hace que se impriman las conclusiones a través del uso de dos sentencias *Display*, puede observarse una imagen del animal identificado (recuadro negro) que se imprimió con la sentencia *Display Image 'Acanthocephala.jpg'*.

Los paneles inferiores de la Fig. 6, muestran:

- En la parte izquierda las salidas del sistema.
- En la parte derecha las variables que se infieren con detalles de: los valores alcanzados, la certidumbre con que se llega a esa conclusión y la regla que permitió probarla. El ingeniero de conocimiento puede usar todas estas facilidades para encontrar errores en el funcionamiento lógico del sistema experto.

La Fig. 7, también muestra el proceso de ejecución paso a paso, pero en este caso en forma gráfica; en ese momento se han explorado y probado todos los antecedentes de la *Regla 1* y el sistema lo hace notar resaltándolos en color azul. Al cumplirse todos los antecedentes de la regla el sistema concluirá que el *Filo* del animal es *Acanthocephala* y una vez que se ejecute su bloque de acciones la resaltará en color verde haciendo notar que la regla se probó con éxito. Las reglas no exploradas en la Fig. 7 permanecen sin resaltar y aparecerán resaltadas en rojo cuando fallen.

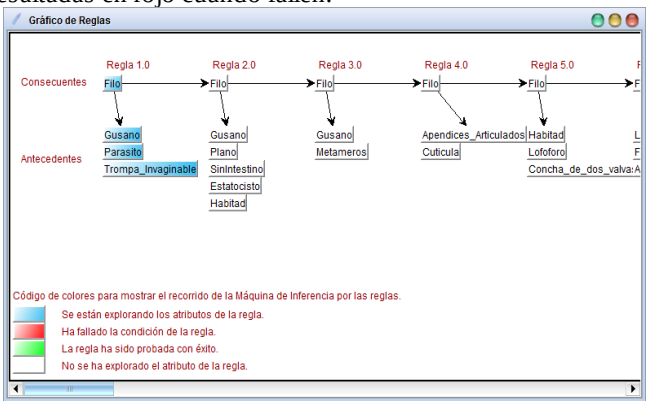

Figura 7. Ejecución de la inferencia en forma gráfica.

Una vez terminada la inferencia también se puede obtener un árbol que se construye de acuerdo al camino recorrido para probar un atributo, la Fig. 8 muestra el árbol después de haber sido probado el atributo *Filo*.

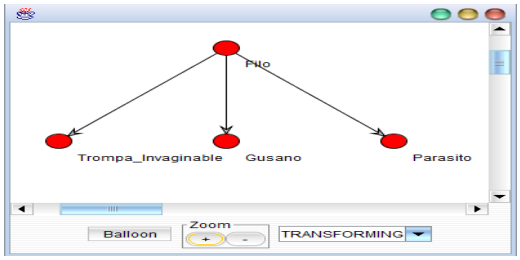

Figura 8. Árbol de prueba para el atributo Filo.

Las facilidades del sistema con relación a la fase de puesta a punto del SE, explicadas en las Fig. 6, 7 y 8, constituyen una buena herramienta en manos de los ingenieros de conocimientos, tanto en la etapa de puesta a punto del SE como en mantenimientos futuros, cuando se necesite agregar nuevas reglas o modificar algunas y también ayudan a la comunicación con los expertos si se usan de manera apropiada

IV. DIRECCIONES DE BÚSQUEDA

UCShell posee las direcciones de búsquedas: dirigidas por objetivos (*backward chaining*) y dirigidas por datos (*forward chaining*).

Para iniciar un proceso dirigido por objetivos se usan las sentencias *FIND* y *FINDALL*, la primera se detiene cuando la máquina de inferencia encuentra una solución, mientras la segunda permite buscar todas las soluciones posibles.

La sentencia *FINDFORWARD* permite iniciar un proceso de prueba dirigido por datos.

Las inferencias se pueden iniciar desde los bloques de acciones de las reglas o desde el bloque de acciones principal, aunque deberá existir al menos una orden de búsqueda (FIND, FINDALL o FINDFORWARD) en el bloque de acciones principal porque de no ser así el SE estaría sintácticamente bien definido pero semánticamente no tendría sentido.

V. MANIPULACIÓN DE LA INCERTIDUMBRE

La manipulación del conocimiento y los datos inciertos, inseguros e inexactos es típica de los SE [11], el mecanismo para manipular la incertidumbre es una parte importante de UCShell.

La incertidumbre se puede expresar por [12]: pesos, medidas, grados de confianza, factores de creencia, probabilidades subjetivas, etc. Esos valores de certeza se expresan dentro del intervalo real (0, 1] en el sistema UCShell [13], donde un valor muy cercano a 0 expresa incertidumbre alta y el valor 1 certidumbre absoluta (es la que se establece por defecto).

La incertidumbre sobre el grado de veracidad de los resultados alcanzados por un sistema experto provine de dos fuentes fundamentales:

- Imprecisión en el valor de los atributos que sirven como hechos iniciales. La interfaz de UCShell ofrece la posibilidad de asignarle a los hechos un valor de incertidumbre dado.
- Existencia de reglas débiles debido a que el experto o constructor del modelo es incapaz de establecer una correlación fuerte entre la condición y la conclusión de la regla, lo cual hace que la implicación no sea categórica y permita la posibilidad de excepciones a la regla.

La Fig. 9 muestra la forma en que se escribe una regla débil.

```
RULE 1
 IF
A = 10      CNF      0.78
THEN
 C:= 19     CNF      0.8
END;
```

Figura 9. Regla débil.

La regla expresa que C es igual a 19 con certeza 0.8 cuando

se conoce, con CNF 0.78, que el valor de A es igual a 10. Este tipo de regla permite que el ingeniero de conocimientos pueda usar datos que no son precisos pero tienen un cierto valor de verdad.

Las generalizaciones de conjunciones y disyunciones juegan un papel vital en el manejo de la incertidumbre en los SE; ellas se usan en la evaluación de la satisfacción de las condiciones, en la propagación de la incertidumbre a través del encadenamiento de las reglas y en la consolidación de la misma conclusión derivada de diferentes reglas.

Las causas de la incertidumbre pueden ser variables. El sistema UCShell da la posibilidad de definir la incertidumbre a los dos actores que usan el sistema:

- El usuario, cuando contesta una pregunta y le asigna un valor de certeza a ese atributo.
- El ingeniero de conocimiento, cuando escribe la regla con algún factor de certidumbre (FC) en cualquiera de sus partes y por tanto la hace débil.

Existen diferentes métodos para calcular la propagación de la certidumbre [14] [15], UCShell utiliza una extensión del Teorema de Bayes [16] para calcularla.

VI. FACILIDADES GENERALES DE UCSHELL

Muchas de las facilidades del sistema UCShell ya se han mencionado en los apartados anteriores y solo se han dejado para esta sección aquellas que tienen que ver con las sentencias que sirven al ingeniero de conocimientos para tomar decisiones de las acciones a seguir.

Las sentencias pueden estar dentro de cualquiera de los dos tipos de bloque de acciones:

- El bloque de acciones de las reglas que debe situarse, opcionalmente, en la parte final de las reglas.
- El bloque de acciones de la BC, que es obligatorio y se localiza como parte final de cualquier BC.

Seguidamente se presentan las sentencias del lenguaje que se usa en las bases de conocimientos de UCShell:

- Sentencia DISPLAY

  Imprime texto e imágenes. Esta facilidad se usa, sobre todo, en las reglas que son conclusivas y también en el bloque de acciones de la BC para informarle al usuario del SE los resultados obtenidos.
- Sentencia FIND

  Inicia una búsqueda dirigida por objetivos que se detendrá cuando se pueda probar el objetivo buscado.
- Sentencia FINDFORWARD

  Inicia una búsqueda dirigida por datos que se detendrá si se logra probar el objetivo.
- Sentencia FINDALL

  Inicia una búsqueda dirigida por objetivos que se detendrá cuando se exploren todas las reglas en donde ese objetivo esté como conclusión. Mientras FIND se detiene cuando logra probar el objetivo por una regla cualquiera, FINDALL sigue tratando de probar las reglas que aún no se han explorado, lo que da la posibilidad de asociar varios valores a una conclusión que se ha obtenido de distintas formas.
- Sentencia ASSUME

  Le asigna un valor a un atributo. Es útil para asignar datos iniciales para una búsqueda dirigida por datos.
- Sentencia REPEAT

  Permite definir un ciclo que se basa en el valor del atributo (desconocido o conocido) que se toma como control.
- Sentencia RESET

  Deja sin valor a los atributos especificados. Su uso habitual está relacionado con la sentencia anterior, sobre todo cuando se ha probado algo y se desea hacer una nueva prueba, posiblemente con valores de respuestas diferentes.
- Sentencia CHAIN

  Permite que la máquina de inferencia cambie su accionar desde la base actual hacia otra base. Es muy útil en sistemas con varias BC, como es el caso del SE Reino Animal [17] que se ha usado como ejemplo en esta artículo.

El sistema UCShell también puede trabajar con expresiones que usan distintos operadores.

Los operadores aritméticos permitidos son: factorial (!), potenciación (^), multiplicación (*), menos unario (-), división (/), resto de la división (%), división entera (Div), resta (-), y suma (+).

Los operadores relaciones que admite el lenguaje son: igual que (=), diferente de (<>), menor que (<), mayor que (>), menor o igual que (<=), mayor o igual que (>=).

También se puede usar una amplia gama de funciones que forman parte del lenguaje y amplían el campo de aplicación del sistema, las funciones son: coseno, arcoseno, coseno hiperbólico, seno, arcoseno, seno hiperbólico, tangente, arcotagente, tangente hiperbólica, logaritmo neperiano, logaritmo natural, raíz cuadrada, potencia de 2, exponencial y valor absoluto.

VII. CONCLUSIONES

Los sistemas expertos han mostrado efectividad desde la época de su surgimiento y aun hoy mantienen su vigencia, por esa razón resulta útil disponer de máquinas de inferencias versátiles y modernas que ayuden a desarrollarlos e implementarlos.

UCShell es más que una máquina de inferencia debido a que integra, en un solo ambiente, un conjunto de facilidades que ayudan a la labor de desarrollar nuevos sistemas expertos, incluyendo todas sus etapas: ingeniería del conocimiento, puesta a punto, prueba, verificación y mantenimiento.

La máquina de inferencia que forma parte del sistema UCShell permite realizar búsquedas dirigidas por objetivos y por datos que se pueden combinar en un SE de acuerdo a las necesidades del sistema.

Con el objetivo de poder usar UCShell sobre diferentes plataformas, la versión actual se programó en el lenguaje Java (las anteriores se hicieron en distintas versiones del lenguaje Pascal), al momento de escribir este artículo se había utilizado sobre los sistemas operativos Windows y Linux.

REFERENCIAS

[1] A. KUMAR MEENA, S. KUMAR. "Study and Analysis of MYCIN expert system". *International Journal of Engineering and Computer Science*, vol. 4, issue 10, pp. 14861-14865, 2015.

[2] R. K. LINDSAY, B. G. BUCHANAN, E. A. FEIGENBAUM, J. Lederberg. "DENDRAL: A case study of the first expert system for scientific hypothesis formation". *Artificial Intelligence*, vol. 61, issue 2, pp. 209-261, 1993. Doi: 10.1016/0004-3702(93)90068-M.

[3] B. G. BUCHANAN, E.H. SHORTLIFFE. *Rule Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project*. Boston: Addison-Wesley, 1984.
[4] R. CORREAL, G. PAJARES, J.J. RUZ. "Automatic expert system for 3D terrain reconstruction based on stereo vision and histogram matching*". Expert Systems with Applications*, vol. 41, pp. 2043-2051. 2014. Doi: http://dx.doi.org/10.1016/j.eswa.2013.09.003
[5] Y. BOUAIACHI, M. KHALDI, A. AZMANI. "A Prototype Expert System for Academic Orientation and Student Major Selection". *International Journal of Scientific & Engineering Research*, vol. 5, issue 11, pp 25-28, 2014.
[6] O. OLADIPO, C.T. OLAYINKA, O.L. POPOOLA. "Mobile Compactable Expert System for the Treatment of Typhoid Fever in Developing Countries". *International Journal of Computer Applications*, vol. 105, issue 2, pp. 16-19, 2014.
[7] M. E., MABROUK, M. EZZIYYANI, Z. A. SADOUQ, M. ESSAAIDI "New expert system for short, medium and long term flood forecasting and warning". *Journal of Theoretical and Applied Information Technology*, vol.78, issue 2, pp. 286-302, 2015.
[8] A. NADA, M. NASR, M. HAZMAN. "Irrigation Expert System for Trees". *International Journal of Engineering and* Innovative *Technology*, vol. 3, issue 8, pp. 170-175, 2014.
[9] H. LÓPEZ-LEÓN (autor), M. G. LEZCANO-BRITO, L. FUNDORA FERNÁNDEZ (asesores). *Implementación de la versión 3.0 de la herramienta UCShell*. Proyecto de grado de la carrera Ciencia de la Computación, Universidad Central "Marta Abreu" de Las Villas, Santa Clara, Cuba, 2013.
[10] N. J. NILSSON. *Principles of artificial intelligence*. Palo Alto: Morgan Kaumann, 1980.
[11] S. DUBEY, R. K. PANDEY, S. S. GAUTAM. "Dealing with Uncertainty in Expert Systems". *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 4, issue 3, pp. 105-11, 2014.
[12] A. JIMÉNEZ-MARTÍN. *Un sistema de ayuda a la decisión multiatributo con asignaciones imprecisas*. Tesis para optar por el grado científico de Doctor en Ciencias, Universidad Politécnica de Madrid, 2002.
[13] M. G. LEZCANO-BRITO. *Ambientes de aprendizaje por descubrimiento para la disciplina Inteligencia Artificial*. Tesis para optar por el grado científico de Doctor en Ciencias, Universidad Central "Marta Abreu" de Las Villas, Santa Clara, Cuba, 1998.
[14] O.P. LE MAÎTRE, O.M. KNIO, H.N. NAJM, R.G. GHANEM. "Uncertainty propagation using Wiener–Haar expansions". *Journal of Computational Physics*, vol. 197, pp. 28-57, 2004. doi:10.1016/j.jcp.2003.11.033
[15] S. POMMÉ, S.M. JEROME, C. VENCHIARUTTI. "Uncertainty propagation in nuclear forensics". *Applied Radiation and Isotopes*, vol. 89, pp. 58-64, 2014. doi:10.1016/j.apradiso.2014.02.005
[16] S. MAHADEVAN, R. ZHANG, N. SMITH. "Bayesian networks for system reliability reassessment". *Structural Safety*, vol. 23, Issue 3, pp. 231-251, 2001. doi:10.1016/S0167-4730(01)00017-0
[17] D. ROJAS-SOSA (autora), M. G. LEZCANO-BRITO, L. FUNDORA FERNÁNDEZ (asesores). *Estudio práctico del sistema UCShell. Mejoras necesarias*. Proyecto de grado de la carrera Ciencia de la Computación, Universidad Central "Marta Abreu" de Las Villas, Santa Clara, Cuba, 2013.

Mateo G. Lezcano. Cibernético-Matemático (1982). Máster en computación Aplicada (1995). Doctor en Ciencias Técnicas (1998). Actualmente Profesor-Investigador de la Universidad Cooperativa de Colombia. Profesor Titular de la Universidad Central "Marta Abreu" de Las Villas, Cuba.

Lissett Fundora. Graduada de Ciencias de la Computación (2011) de la Universidad Central "Marta Abreu" de Las Villas, Cuba. Trabaja actualmente en forma independiente.

Humberto López. Graduado de Ciencias de la Computación (2014) de la Universidad Central "Marta Abreu" de Las Villas, Cuba. Trabaja actualmente en la "Empresa de Tecnologías de la Información para la Defensa".