ReVisión
Revista de Investigación en Docencia Universitaria de la Informática
AENUI Asociación de Enseñantes Universitarios de la Informática

ToolboX: Una estrategia transversal para la enseñanza de la programación en entornos educativos

Francisco Vico
Dpto. Lenguajes y Ciencias de la Computación
ETS Ingeniería Informática
Universidad de Málaga

Resumen

Subestimar la importancia del aprendizaje de la programación de ordenadores, en una sociedad que tiende a articularse en torno al *software* y la producción robotizada, puede ser un error político de consecuencias impredecibles, tanto para el desarrollo y la integración de las personas, como para la competitividad científica e industrial. Las iniciativas para promover la programación entre la población infantil y juvenil son escasas y torpes, lastradas por una inexistente metodología para estas edades, en lo que se refiere a estándares en herramientas didácticas y métodos de evaluación. Es necesario, por tanto, explorar estrategias que eliminen las barreras de entrada y que integren la programación en el ámbito escolar y extraescolar. En este sentido se propone ToolboX, un entorno de desarrollo orientado al aprendizaje de un lenguaje de programación científico (GNU Octave) que es abierto y cubre las necesidades académicas, desde la educación primaria hasta el desarrollo profesional. Esta herramienta incluye problemas de programación extraídos de las materias del currículo docente, y puede utilizarse sin necesidad de mentorización, para facilitar su implantación en centros educativos. Se describe la motivación, estructura, funcionamiento e implementación de esta herramienta, que viene a ocupar un particular nicho, aún vacío, en la enseñanza de la programación.

Palabras clave: Entorno de desarrollo integrado, enseñanza de la programación, *scripting*, GNU Octave, Linux.

1. Introducción

El paradigma de organización social al que converge la civilización tecnológica asigna a los sistemas informáticos un papel central, tanto en el sector productivo, como en el funcionamiento de las instituciones y en los medios de información y comunicación entre las personas. Desde las plantas robotizadas hasta los ubicuos dispositivos móviles, pasando por vehículos sin conductor y servidores de Internet, comprender la arquitectura y funcionamiento de estos sistemas, será uno de los aspectos más relevantes de la formación del ser humano en la era digital. Desde hace años, motivado principalmente por la demanda de desarrolladores en la empresa, varias iniciativas (entre las que destaca *Hour of code* (`https://hourofcode.com/`)) promueven la enseñanza de la programación a nivel escolar [4]; incluso comienzan a incluirse estas materias en los últimos cursos de la formación preuniversitaria. La modificación del sistema educativo es, no obstante, un proceso lento, que requiere sensibilidad política y la implicación de numerosos actores. Por otra parte, y a pesar de que la utilización de dispositivos electrónicos y la programación de ordenadores resulta intuitiva para los niños ya desde el inicio de su periodo escolar, comprobamos a diario que los dispositivos actuales fomentan su uso principalmente como plataformas de videojuego y sistemas de información e interacción social. En efecto, a un lado queda la aridez de la programación, que fuera tan fomentada y popular en los ordenadores personales de las décadas de los 80 y 90. Vemos, por tanto, que el conjunto de los obstáculos institucionales y de mercado descritos dificultan que la enseñanza pública de la programación llegue a todos los estratos sociales y complemente el curriculum formativo preuniversitario.

Podemos identificar también limitaciones de carácter coyuntural. Si bien la enseñanza universitaria de la informática se ha generalizado y asentado en España en los últimos tres decenios, existe un vacío curricular y metodológico para su enseñanza preuniversitaria (sólo cubierto parcialmente por los ciclos de formación profesional), del que sólo comenzamos a

tener consciencia y que genera tímidas medidas en el marco regional. Sabemos cómo enseñar a los niños a leer y a escribir (incluso a edades, quizás, demasiado tempranas), ahora bien: ¿cómo enseñarles a programar? Escasean los estudios en este sentido [2, 5] y, por lo general, implican muestras y metodologías limitadas. Así, por ejemplo, a pesar de su popularidad, aún es cuestionable la eficacia de Scratch[1] para enseñar a programar correctamente a niños de entre 8 y 16 años (adquisición de malos hábitos [10], no afecta a la capacidad de resolver problemas [8] y algunos resultados positivos sobre muestras no representativas [3, 9]), rango de edades donde se concentra su uso [13]. Ahondando en lo mismo, el docente carece de estándares para evaluar el grado de aprendizaje de las habilidades en programación; estándares que, por otro lado, abundan para profesionales del sector informático, dada su importancia para la contratación de programadores [7, 11], en tanto que para niveles escolares las primeras propuestas son recientes, escasamente contrastadas y asociadas a organizaciones sin ánimo de lucro o productos comerciales.[2] Por último, a diferencia del resto de áreas, la enseñanza de la programación precisa de ordenadores y personal técnico dedicado a su instalación y mantenimiento. Si bien los ordenadores constituyen un elemento docente en la mayoría de los centros educativos, la obsolescencia del equipamiento, la fiabilidad del acceso a Internet y la alta demanda hacen que la implantación de la programación dependa fuertemente de un presupuesto complementario para dotar del equipamiento necesario.

Este escenario ha llevado a considerar el aprendizaje de la programación más como una actividad extraescolar, que como una competencia esencial en el currículo de la educación primaria y secundaria. De igual modo que se recomienda el aprendizaje de la práctica musical por sus efectos beneficiosos sobre el desarrollo cerebral y el rendimiento académico [1, 6], persiguiendo similares objetivos, se corre el riesgo de delegar el pensamiento computacional y la práctica de la programación, a un nivel complementario. Éste sería un camino equivocado para resolver el problema de la alfabetización digital, no sólo por el carácter elitista con que se adoptaría la solución, sino, principalmente, por el rol infravalorado que asumiría la programación en una sociedad cada vez más dependiente (en todos los niveles de la vida) del *software*.

Desde un punto de vista meramente educativo, podemos preguntarnos si la programación merece el carácter de materia curricular. Para dar respuesta a esta pregunta se ha elaborado una relación de las competencias que se adquieren en el aprendizaje de la programación (basada principalmente en `http://resources.kodable.com/public/cs_standards_2017.pdf` y la experiencia docente del autor) y se ha establecido la relación con los objetivos y estándares del sistema educativo español (cuadro 1). En concreto, se incluyen las disposiciones del currículo básico de la educación primaria a nivel nacional[3] y en el correspondiente currículo de la Comunidad Autónoma de Andalucía[4], manteniendo la nomenclatura de cada texto, para una identificación más ágil. De todas las áreas propias de la educación primaria, vemos que la programación guarda relación principalmente con elementos curriculares de las áreas de lengua y matemáticas: por la naturaleza lingüística de los lenguajes de programación, en el primer caso, y por los componentes lógico, numérico y de resolución de problemas de la computación, en el caso de la matemática. El cuadro 1 muestra que la consecución de muchos de los objetivos y estándares del sistema educativo español se verían reforzados con la inclusión de asignaturas de programación.

Tal y como se ha expuesto anteriormente, no es de esperar que argumentos como la presión del sector informático (tanto por su vertiente empresarial[5] como por la colegiada), o la evidente complementariedad curricular, provoquen los cambios legislativos necesarios para introducir la programación en escuelas e institutos. Una estrategia alternativa pasaría por modificar este *statu quo* adoptando herramientas que perfundan los conceptos de la programación, integrándose en el sistema educativo actual y eliminando las barreras que los actuales ordenadores plantean al aprendizaje de estas técnicas. ToolboX se propone como una solución transversal, alineada con este tipo de estrategia. Técnicamente, ToolboX convierte el intérprete de GNU Octave, un lenguaje de programación científica en código abierto y equivalente a Matlab[6], en un entorno de programación, que permite al alumno resolver problemas de las asignaturas que cursa, principalmente de matemáticas, física, química o tecnología (por ejemplo, problemas de cálculo, de física clásica, disoluciones o circuitos eléctricos), pero también de biología o lengua (como genética mendeliana, o análisis sintáctico). El objetivo es dotar al docente de una herramienta que permita la resolución de problemas de las asignaturas que imparte, para que los alumnos

[1] Lenguaje de programación visual desarrollado por el MIT [12], accesible en `https://scratch.mit.edu/`.

[2] Véanse los documentos: `https://www.iste.org/standards/standards/standards-for-computer-science-educators`, `https://c.ymcdn.com/sites/www.csteachers.org/resource/resmgr/Docs/Standards/2016StandardsRevision/INTERIM_StandardsFINAL_07222.pdf`, `https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-computer-science-principles-course-and-exam-description.pdf`, `https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-computer-science-a-course-description.pdf`

[3] Ministerio de Educación, Cultura y Deporte. Gobierno de España (2014). Real Decreto 126/2014, de 28 de febrero, por el que se establece el currículo básico de la Educación Primaria. Boletín Oficial del Estado, 1 de marzo de 2014, 52:19349-19420. `https://www.boe.es/boe/dias/2014/03/01/pdfs/BOE-A-2014-2222.pdf`

[4] Consejería de Educación, Cultura y Deporte. Junta de Andalucía (2015). Orden de 17 de marzo de 2015, por la que se desarrolla el currículo correspondiente a la Educación Primaria en Andalucía. Boletín Oficial de la Junta de Andalucía, 27 de marzo de 2015, 60:9-696. `http://www.juntadeandalucia.es/boja/2015/60/BOJA15-060-00831.pdf`

[5] El apoyo a la enseñanza de la programación en Estados Unidos procede, en buena medida, de empresas del sector tecnológico, como Google, Facebook o Microsoft.

[6] Marca registrada de The MathWorks, Inc.

los solucionen desarrollando programas en un ordenador, de modo que el aprendizaje de la programación sea paralelo al de otras áreas —como método de resolución, más que como materia—. A continuación se describe la estructura, funcionamiento, implementación y posibilidades de implantación de ToolboX. El apartado de implementación tiene un carácter técnico y su propósito es ilustrar la viabilidad tecnológica de este tipo de herramientas; al margen de los ejemplos que incluye, para apoyar algunos conceptos, su lectura es prescindible. Por último, ToolboX puede utilizarse descargándolo de `http://toolbox.uma.es/` y siguiendo las instrucciones de instalación.

2. Descripción de ToolboX

En este apartado se definen los principales elementos que conforman este entorno de desarrollo integrado, describiendo su funcionamiento y las tareas de programación que incluye.

2.1. Tarea de programación

Cuando se dispone de un ordenador, de un lenguaje de programación y de un entorno de desarrollo, uno de los principales problemas que debe afrontar el docente de la informática (o el aprendiz autodidacta) es: ¿qué programar? Las características del problema a resolver deben adaptarse a los conocimientos ya adquiridos y a un plan formativo. El concepto de tarea de programación suple esta necesidad en ToolboX.

En este contexto, una tarea es un problema que puede resolverse mediante la realización de un programa y que incluye toda la información necesaria para su planteamiento (enunciado y gráficos), resolución (pasos a seguir) y corrección (solución). La tarea se refiere al concepto general de problema, abarcando cualquier ejercicio que puede plantearse en el sistema educativo, como el ejemplo incluido en la figura 1. Sin embargo, algunos tipos de problemas se pueden resolver fácilmente mediante un programa de ordenador, mientras que en otros, aparentemente sencillos, su programación plantea dificultades. Por ejemplo, el problema de la figura 1 se resuelve con expresiones aritméticas sencillas, mientras que una factorización requiere el uso de vectores, en tanto que algunos problemas de matemáticas (resolución de sistemas de ecuaciones), biología (genética mendeliana), química (fórmula empírica) o lengua (ortografía), exigen soltura manipulando cadenas de caracteres. Si bien cualquier problema resoluble es susceptible de ser programado, en adelante denominaremos «problegramas» (acrónimo formado por «problema» y «programa») a los problemas cuya solución, en general, guarda similitud con un programa de ordenador.[7]

La adquisición de competencias en programación tiene lugar en ToolboX indirectamente, a través de la resolución de problegramas. La resolución de las tareas que incluye permite aprender tanto el funcionamiento del entorno y los principales conceptos del lenguaje Octave, como contenidos concretos de las materias propias de la enseñanza preuniversitaria. Con esta finalidad, las tareas se organizan en módulos, que son relaciones de tareas que deben resolverse en secuencia. Los módulos definen itinerarios que cubren unidades didácticas concretas; así, pueden definirse módulos para trigonometría básica, para cónicas, para movimiento rectilíneo o para formulación orgánica, cuyo nivel de dificultad vendrá establecido por el curso al que se dirige dicho módulo.

El enfoque de ToolboX es totalmente abierto, también en cuanto a las tareas, por lo que, aparte de las incluidas en el entorno, el docente puede crear tareas propias para ser resueltas por sus alumnos, como se describirá en el apartado de implementación. El sitio web de ToolboX actuará como servidor para las tareas creadas en los distintos centros, integrándose en versiones sucesivas las nuevas tareas y convirtiendo así en colaborativo el proceso de generación de contenidos.

2.2. Entorno de programación

Por su diseño, y de manera resumida, ToolboX pretende implementar en un ordenador actual una experiencia de usuario similar a la de los ordenadores personales de la década de los 80 (ZX Spectrum, Commodore o Dragon), que sólo ofrecían un entorno básico de programación. Al ejecutarse ToolboX, la pantalla se divide en tres áreas (figura 2): ventana de instrucciones (terminal), ventana gráfica y ventana de programa (editor de texto). Inicialmente las ventanas de instrucciones y gráfica se reparten la mitad izquierda y la de programa la mitad derecha, si bien esta configuración puede modificarse en función del tipo de tarea a realizar. El usuario puede alternar entre las ventanas de instrucciones y de programa presionando la tecla tabulador,[8] haciendo prescindibles los dispositivos apuntadores (ratón y panel táctil) y las teclas especiales (de función, bloqueo numérico y de desplazamiento, pausa, *Super*, *Alt*, *Alt Gr* y *Menu*). Esta limitación en la interfaz de entrada persigue evitar que el usuario inexperto pueda alterar el entorno, cerrando una ventana, o activando una función diferente a la prevista. Con el mismo objetivo, se eliminan de las ventanas las cabeceras y menús, así como la barra principal del escritorio, maximizando al mismo tiempo el área de trabajo, dado que los equipos en centros escolares o de acceso público a Internet pueden utilizar monitores de pequeñas dimensiones (hasta de 10.1 pulgadas). De este modo, las ventanas de ToolboX quedan siempre a la vista y, alternando entre ellas, el usuario realiza tareas de interacción y consulta en la ventana de instrucciones, de programación en el editor, o de visualización de datos y resultados en la ventana gráfica.

Este diseño simplificado, en efecto, evoca a los antiguos

[7] Encontramos abundantes ejemplos de problegramas en la aritmética, por ejemplo, y menos en el cálculo diferencial, ya que la mayoría de los lenguajes de programación están más orientados al cálculo numérico que al simbólico. Más información y ejemplos pueden consultarse en `https://es.wikipedia.org/wiki/Problegrama`.

[8] Se ha elegido 'tabulador' por ser una de las teclas de mayor tamaño, fácil acceso y tener, generalmente, un símbolo impreso con las flechas hacia derecha e izquierda.

Competencia	**Currículo nacional**	**Currículo CCAA**
Comprender que programar es utilizar un lenguaje sencillo para comunicarse con el ordenador	—	O.LCL.1
Asimilar la gramática de un lenguaje de programación	LCL.4.4.6	LCL.STD.35.6
Diseñar, estructurar y escribir programas gramaticalmente correctos, comentando los aspectos computacionales implicados computacionales implicados	MAT.3.4.2 LCL.4.1.1	MAT.STD.24.2 LCL.STD.32.1
Adquirir capacidad para la resolución de problemas mediante la creación de programas	MAT.1.1.1 MAT.1.2.1 MAT.1.2.3	O.MAT.1 MAT.STD.1.1 MAT.STD.2.1 MAT.STD.2.3
Manejar magnitudes numéricas y representar relaciones entre ellas	–	O.MAT.3
Identificar estructuras y patrones en los procesos	MAT.1.3.1	MAT.STD.3.1
Predecir el funcionamiento de un programa	MAT.1.3.2 MAT.1.7.1	MAT.STD.3.2 MAT.STD.7.1
Buscar alternativas de resolución de un programa	MAT.1.4.1	MAT.STD.4.1
Valorar diferentes soluciones en base a criterios de optimización de código y velocidad de cómputo	MAT.1.10.1	MAT.STD.12.1
Expresiones aritméticas simples y complejas: jerarquía de operaciones y paréntesis	MAT.2.5.1 MAT.2.6.8	MAT.STD.16.1 MAT.STD.19.8
Expresiones con operadores relacionales	MAT.2.2.4	MAT.STD.15.4
Expresiones con operadores lógicos[9]	–	–

Cuadro 1: Relación entre las competencias adquiridas en el aprendizaje de la programación y los currículos estatal y de la comunidad autónoma andaluza para la educación primaria. Al referir los estándares se utiliza la nomenclatura propia del boletín oficial correspondiente, para facilitar su localización en el texto oficial; en el caso del BOE: A.B.X.Y, siendo A el área (LCL para lengua castellana y literatura, o MAT para matemáticas), B es el bloque y X.Y el estándar de aprendizaje evaluable; y en el caso del BOJA: A.STD.X.Y (área y estándar de aprendizaje) y O.A.N (objetivo de un área y número del objetivo).

enunciado:	Un cesto de uvas pesa 265 kilogramos. Queremos repartir 5 cestos entre 25 personas. ¿Cuántos kilogramos se llevará cada uno?
método:	Calcula primero cuántos kilos en total pesan los 5 cestos. Debes dividir el total de kilos entre las 25 personas.
solución:	53

Figura 1: Ejemplo de problema de matemáticas de últimos cursos de primaria.

terminales textuales. Sin embargo, la enseñanza de materias como la matemática, la física o la química precisan de recursos que van más allá del código ASCII o el UNICODE. Con esta finalidad, ToolboX puede visualizar lenguaje LaTeX, así como animaciones. LaTeX permite incluir expresiones matemáticas, de cualquier nivel de complejidad, en la definición de las tareas (tanto en el enunciado, como en los mensajes de ayuda), así como diagramas gráficos, lo que convierte ToolboX en un sistema avanzado de edición, capaz de visualizar textos con la calidad de los libros utilizados en docencia (un ejemplo se muestra en la figura 9). Asimismo, el terminal permite la visualización de vídeos y animaciones, como complemento en la definición de las tareas.

Los siguientes apartados describen en detalle la funcionalidad de cada ventana.

2.3. Ventana de instrucciones

La programación en general, y particularmente el *scripting* (la creación de pequeños programas con finalidad concreta o *scripts*[10]) al que está orientado ToolboX, requiere una interacción continua con el entorno, para buscar ayuda, consultar valores de variables y comprender el comportamiento de las funciones. Instrucciones como `help` o `lookfor` sustituyen al clásico manual de usuario. Este tipo de operaciones complementarias de bajo nivel se realizan en la ventana de instrucciones.

El usuario puede ejecutar aquí cualquier función propia del lenguaje Octave (por ejemplo, un bucle, una instrucción, o una operación matricial), así como consultar y manipular el espacio de trabajo (variables), que es independiente del espacio de trabajo del mismo entorno (es decir, el conjunto de variables utilizado para implementar ToolboX). Adicionalmente, la ventana de instrucciones permite, mediante instrucciones de alto nivel, configurar el entorno. De este modo, el conjunto de instrucciones disponible para el usuario abarca los propios del lenguaje de programación (excepto algunos, cuya ejecución está bloqueada para evitar que el usuario pueda alterar la configuración del entorno; como la ejecución de instrucciones en Linux mediante la instrucción `system`), y otros externos al lenguaje, por ejemplo, para consulta y selección de las tareas (`task`), o la ejecución del programa (`go`). El cuadro 2 muestra algunos ejemplos de ambos bloques de funciones.

ToolboX define un superconjunto del lenguaje Octave, en tanto que incluye la sintaxis completa de Octave (salvo las funciones no permitidas por seguridad) y la amplía, con las instrucciones externas para la gestión de tareas y del mismo entorno, así como con nuevas estructuras sintácticas. Por ejemplo, se incluye el bucle definido,[11] con la sintaxis incluida en la figura 3.

La implementación de este tipo de bucle en Octave resulta significativamente más compleja (utilizando una variable contador y un rango de valores numéricos, como muestra la figura 4), lo que representa una limitación para el aprendizaje de una de las estructuras iterativas esenciales en los estadíos iniciales.

2.4. Ventana gráfica

La necesidad de mostrar información esquemática y de interaccionar con entornos complejos para la definición de tareas, justifica la inclusión de una ventana gráfica. En esta ventana se representa, o bien una imagen única que aporta información para la resolución de un problema (complementaria al enunciado), o bien un mundo virtual dinámico, con el que el usuario puede interactuar. Al margen de estas funcionalidades, la ventana gráfica tiene utilidad para que el usuario muestre datos obtenidos con las instrucciones de Octave destinados a representación gráfica, como `plot` (diagramas de dispersión), `hist` (histogramas) o `bar` (diagramas de barras).

2.5. Ventana de programa

La mitad derecha de la pantalla presenta un editor orientado a la programación, con funcionalidades de interés para su integración en el entorno. Este editor, en función del nivel de conocimientos del usuario, podrá utilizar elementos profesionales en la edición de programas: números de línea, plegado

[9] Las aportaciones del componente lógico de la programación están más presentes en la educación secundaria, mientras que en primaria sólo guardan relación con el aprendizaje de la conjunción, disyunción y negación en lenguas extranjeras.

[10] El *script* se referenciará en adelante mediante el término 'programa'.

[11] Esta es la única estructura sintáctica adicional en la versión 1.0; aunque el objetivo es mantener el lenguaje original, no se descartan nuevas estructuras, en la medida en que faciliten la adquisición de la sintaxis de Octave, especialmente para usuarios de corta o muy avanzada edad. (Se utiliza la notación Backus-Naur para representar constructos sintácticos.)

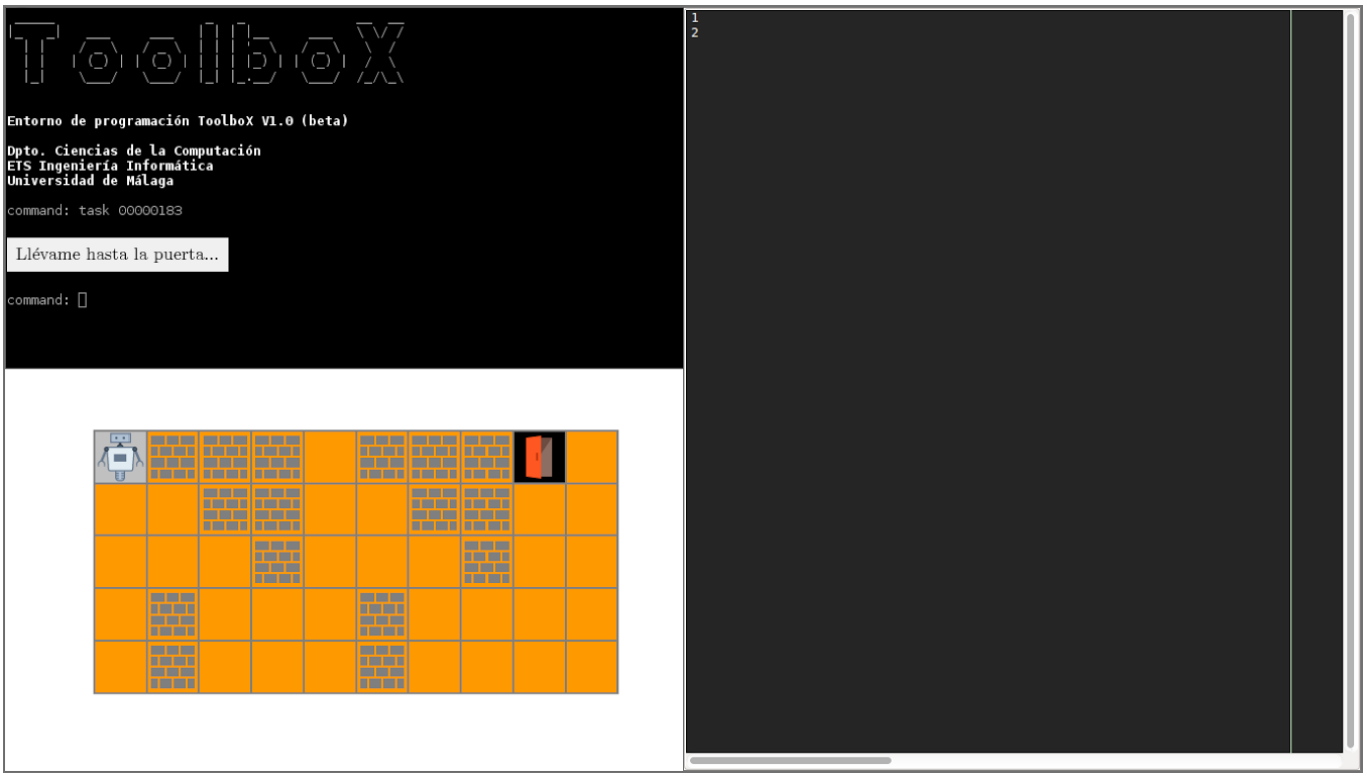

Figura 2: ToolboX divide la pantalla en tres áreas funcionales: ventana de instrucciones (arriba a la izquierda), ventana gráfica (abajo a la izquierda) y ventana de programa (mitad derecha).

Funciones del lenguaje Octave	**Descripción**
`sqrt`	cálculo de la raíz cuadrada
`find`	buscar elementos en un vector
`mean`	cálculo de la media aritmética
`plot`	mostrar datos gráficamente
`printf`	mostrar datos textualmente
Funciones externas al lenguaje	**Descripción**
`task`	enumerar tareas/cargar una tarea
`go`	ejecutar el programa
`help`	ayuda y listado de funciones
`statement`	enunciado de la tarea

Cuadro 2: Ejemplos de funciones utilizadas.

```
repeat <iteraciones>
    <instrucciÇn>+
end
```

Figura 3: Estructura del bucle definido repeat en ToolboX.

```
for <variable> = <valorinicial>:[<incremento>:]<valorfinal>
    <instrucciÇn>+
end
```

Figura 4: Estructura del bucle for en Octave, utilizando un contador de iteraciones.

de bloques de texto, o resaltado sintáctico; elementos que pueden ser activados mediante instrucciones internas.

El entorno ToolboX realiza la gestión de ficheros en base a una estructura de directorios fija, por lo que, en cada momento, el editor sólo permite modificar un programa (un fichero de texto) para la tarea que realiza actualmente el usuario. Este programa se carga en el editor, se salva automáticamente[12] cada vez que el usuario cambia de ventana, y es ejecutado desde la ventana de instrucciones. De este modo, el usuario no tiene acceso al nombre lógico del fichero, ni su ubicación física, sólo puede alterar su contenido. Éste es un aspecto importante, al separar el aprendizaje de la programación de los conocimientos del sistema operativo y la gestión de carpetas y ficheros.

3. Funcionamiento de una sesión

Una sesión en ToolboX consiste en la realización de un conjunto de tareas. Como se ha descrito en el apartado anterior, cada tarea consiste en un problema de programación planteado al usuario. La sesión comienza al iniciarse el entorno ToolboX, y termina con la ejecución de una instrucción de salida (`quit` o `shutdown`).

La instrucción `task` permite enumerar las tareas disponibles (invocada sin argumentos) o cargar una tarea concreta para la resolución de un problegrama. Tras cargarse la tarea aparecerá el enunciado en la ventana de instrucciones (que puede reproducirse posteriormente con la instrucción `statement`). A partir de aquí, la instrucción `tip` permite obtener información sobre cómo resolver el problema planteado (tanto en cuanto a la naturaleza del mismo, por ejemplo, aportando una ecuación para un problema de física; como en cuanto a la programación de la solución, por ejemplo, indicando el tipo de estructura condicional adecuada). En ocasiones, estas pistas pueden llevar al usuario a recabar más información con la instrucción `help`; por ejemplo, `help repeat` muestra la sintaxis de este tipo de bucle, como ilustra la parte superior de la figura 5, siempre mediante ejemplos y evitando formalismos.

Simultáneamente, al cargar la tarea aparece en la ventana de programa el código contenido en la definición de la tarea, que puede ser parte del programa; una estructuración de la solución mediante comentarios; una versión con errores para corregir; o estar vacía, para que el usuario desarrolle todo el programa. La tecla de tabulación activa alternativamente las ventanas de instrucciones y edición, permitiendo al usuario activar la ventana correspondiente para programar o bien consultar información.

Finalmente, cuando el usuario ha concluido el programa, lo ejecutará con la función `go`. Si el programa no es correcto, la ventana de instrucciones mostrará el mensaje correspondiente, que puede ser información para localizar un error sintáctico, o una discordancia en el valor calculado (respecto al campo *solution* de la definición de la tarea). En el primer caso, los mensajes de error de código generados por Octave son convertidos a un lenguaje más cercano al usuario profano, desconocedor de la jerga informática (la parte inferior de la figura 5 muestra un ejemplo de informe ante un error sintáctico). Terminada la fase de depuración de errores, el programa resuelve la tarea planteada y se muestra la conclusión (campo *takehomemessage* de la tarea), tras lo cual, se puede cargar una nueva tarea, o cerrar la sesión con `quit`.

4. Implementación

4.1. Definición e interpretación de tareas

ToolboX almacena la definición de las tareas en notación JSON (*JavaScript Object Notation*), un formato estándar abierto para el intercambio de datos. Para ilustrarlo con un ejemplo, la figura 6 muestra la representación del problegrama de la figura 1: los nombres de los campos están entrecomillados y seguidos por el signo de dos puntos, al que sigue el valor que recibe (una cadena, un número, o un conjunto de valores entre corchetes). Este tipo de representación permite la reutilización de las tareas por otros programas y una creación o edición sencilla mediante un editor de texto.

La definición de una tarea comienza con un campo *class*,[13] seguido del enunciado y la solución del problema (que puede ser un número, una cadena, un vector, etc), así como información relevante para guiar en su resolución (`tip`, o pista) y la conclusión extraída de la realización de la tarea (es decir, lo que ha aprendido mediante su resolución, o *takehomemessage*). Adicionalmente, se incluye metainformación para la gestión de la tarea, como un posible programa Octave que resuelve el problema; palabras clave para su agrupación en módulos (por materia o curso); datos del autor (nombre y URL, del autor o de la página donde se publica el problema) y tipo de licencia *Creative Commons* con que se utiliza en ToolboX.

[12] Liberar al alumno de la operación de salvado evita comportamientos inesperados por la no actualización del programa.

[13] La clase coincide con el nombre del complemento que interpreta la tarea, descrito en el siguiente apartado.

[14] Este complemento se define a partir de una librería (*board*) que implementa las funciones básicas para crear un tablero y las interacciones de las piezas sobre él, lo que facilita la creación de otros complementos, por ejemplo, juegos de mesa.

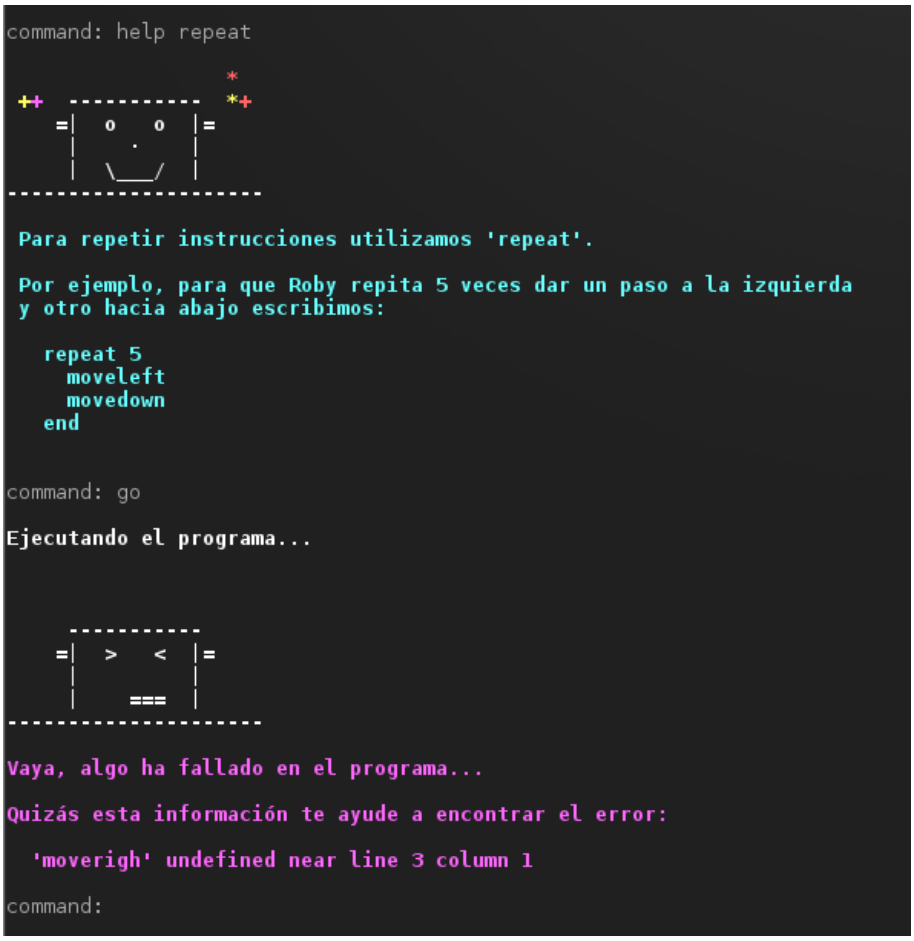

Figura 5: Utilización de la función de ayuda help (arriba) y de un informe de error sintáctico tras ejecución del programa (abajo).

```
{
  "class": "wordproblem",
  "statement":
       "Un cesto de uvas pesa 265 kilogramos. Queremos repartir 5 cestos
       entre 25 personas. ºCuÇntos kilogramos se llevarÇ cada uno?",
  "solution": [53],
  "tip": [ "Calcula primero cuÇntos kilos en total pesan los 5 cestos.",
           "Debes dividir el total de kilos entre las 25 personas." ],
  "takehomemessage": "El verbo 'repartir' se asocia a la operaciÇn 'dividir'.",
  "keyword": [ "matemÇticas",
               "4º primaria",
               "5º primaria" ],
  "program": [ "## peso de un cesto de uvas en kg\\pesocesto = 265" ],
  "author": "Juan A. Ortiz",
  "URL": "http://www.colegiointelhorce.com/",
  "CC": "BY 3.0"
}
```

Figura 6: Definición en formato JSON del problema de la figura 1.

La figura 7 contiene un ejemplo de definición de una tarea para un complemento (*puzzle*[14]) que modela la interacción de un robot con un mundo cuadriculado y en el que puede encontrar otros agentes. El escenario que muestra la parte inferior izquierda de la figura 2 se genera a partir de esta tarea, donde los campos específicos *layout* y *tile* definen la configuración del mundo (10×5 cuadros de color naranja, con líneas separadoras en gris de dos píxeles de ancho) y de los objetos que contiene (el robot en su posición inicial, *roby*; obstáculos en forma de muros, *wall*; y una puerta como objetivo a alcanzar, *doorredopened*),[15] especificando su posición en el tablero,[16] el color de fondo de estas cuadrículas y una etiqueta para identificar el objeto o la casilla.

Dado que algunos problegramas precisan de notación matemática, así como de gráficos de apoyo para comprender el planteamiento, la definición de la tarea puede opcionalmente incluir código LaTeX. La figura 8 muestra un ejemplo de tarea de matemáticas incluyendo un diagrama[17] de la órbita terrestre en torno al sol, así como ecuaciones que ayudan a resolver el problema. La visualización del enunciado y las ayudas correspondientes a esta tarea se muestra en la figura 9.

Por último, las ecuaciones utilizadas son propias de la enseñanza secundaria y están definidas en un fichero interno con formato JSON (figura 10), lo que facilita el trabajo de creación de tareas, pues evita reescribir las fórmulas directamente en código LaTeX para cada tarea, y con la posibilidad de actualizar este fichero con nuevas fórmulas de cualquier área de conocimiento por parte del profesor.

4.2. Integración de utilidades y estructura

ToolboX integra varias aplicaciones Linux de código abierto. Así, la ventana de instrucciones se ejecuta sobre el emulador de terminal que permite visualizar texto en color y también objetos gráficos (figuras y vídeos), lo que lo hace especialmente adecuado para la integración de LaTeX. La ventana de edición se implementa mediante un editor de texto,[18] configurado para que la interacción resulte sencilla para el usuario inexperto. En cuanto a la ventana gráfica, se define a partir del *toolkit* gráfico Qt,[19] implementada como una figura de Octave.

En cuanto al formato de ficheros, todas las imágenes utilizadas en el entorno ToolboX están libres de derechos de autor y se almacenan en el formato abierto *Portable Network Graphics* (PNG). Al margen de las que acompañan a la versión descargable, el docente puede añadir imágenes propias para la definición de tareas. En cuanto a ficheros de vídeo para presentar bloques didácticos o aclarar conceptos concretos (también permite material del docente), el formato abierto utilizado es WebM y pueden reproducirse mediante la instrucción `video`.

La aplicación que gestiona la interacción con el usuario, la realización de tareas y la administración está implementada mediante programas en los lenguajes Bash y GNU Octave 4.0, por lo que algunas de las funciones (como la gestión de ventanas o el bloqueo de teclas) se controlan mediante instrucciones de sistema operativo, y otras (por ejemplo, la gestión de tareas y sesiones) en un lenguaje de alto nivel. La figura 11 muestra la estructura de directorios y algunos de los programas Bash y Octave, así como ficheros de definición de problemas, que componen la aplicación. El programa toolbox crea las ventanas de programa (editor de texto orientado a programación) y de instrucciones, ejecutando un emulador de terminal que, a su vez, ejecuta el programa toolboxloop, encargado éste último de ejecutar toolbox.m, programa en Octave que inicia una sesión de ToolboX; y que será ejecutado nuevamente desde toolboxloop si la sesión termina de manera anómala (por ejemplo, presionando Ctrl-C, para detener la ejecución de un programa).

ToolboX se estructura en cuatro directorios, cuya finalidad es:

- `lib/` almacena todo el código fuente (programas Bash y Octave), separando en el directorio `core/` programas para las funciones principales, de uso interno, y en el directorio `m/` las instrucciones que pueden ser ejecutados por el usuario. Otros directorios contienen librerías de uso común en diferentes módulos (*board* para gestión del tablero gráfico donde se mueve Roby, o JSONlab para la gestión de ficheros JSON).
- Las clases de tareas se definen mediante complementos en el directorio `addon/`. Cada directorio dentro de `addon/` contiene un programa Octave con el mismo nombre (coincide con el campo *class* de la tarea), que inicializa la ejecución de la tarea (por ejemplo, representación del tablero en la ventana gráfica si se trata de una tarea de la clase *puzzle*). Así mismo, un complemento puede aportar nuevas funciones que el usuario puede utilizar para la resolución de la tarea, y que se almacenan en el correspondiente directorio `m/` (como la función `moveright` en el complemento *puzzle*).
- El directorio `util/` almacena los ficheros para la configuración del sistema operativo,[20] mediante el programa `install/toolboxinstall`, y su posterior ejecución (en la distribución instalada en disco duro, o en modo *live* desde una memoria USB).
- `content/` es el directorio donde se almacenan las tareas, módulos y ficheros multimedia, todos dedicados a la realización de tareas. Las tareas se almacenan en el

[15] La quinta columna del campo tile especifica el nombre del fichero gráfico que corresponde al objeto.
[16] Coordenadas horizontal y vertical, siendo (1,1) la esquina superior izquierda.
[17] Los dibujos se realizan mediante el paquete TikZ[14].
[18] La versión 1.0 utiliza el editor Geany.
[19] GNU Octave utiliza, a partir de su versión 3.7, QtHandles.
[20] Lubuntu 16.0.4 en la versión 1.0.

```
{
  "class":     "puzzle",
  "statement": "Debemos alcanzar la puerta.",
  "solution":  "target",
  "layout":  [ 10, 5, "orange",    "gray",   2],
  "tile":    [[ 1, 1, "lightgray", "start",  "roby"          ],
              [ 2, 4, "ground",    "wall",   "wall"          ],
              [ 2, 5, "ground",    "wall",   "wall"          ],
              [ 2, 1, "ground",    "wall",   "wall"          ],
              [ 3, 1, "ground",    "wall",   "wall"          ],
              [ 4, 1, "ground",    "wall",   "wall"          ],
              [ 3, 2, "ground",    "wall",   "wall"          ],
              [ 4, 2, "ground",    "wall",   "wall"          ],
              [ 4, 3, "ground",    "wall",   "wall"          ],
              [ 6, 4, "ground",    "wall",   "wall"          ],
              [ 6, 5, "ground",    "wall",   "wall"          ],
              [ 6, 1, "ground",    "wall",   "wall"          ],
              [ 7, 1, "ground",    "wall",   "wall"          ],
              [ 8, 1, "ground",    "wall",   "wall"          ],
              [ 7, 2, "ground",    "wall",   "wall"          ],
              [ 8, 2, "ground",    "wall",   "wall"          ],
              [ 8, 3, "ground",    "wall",   "wall"          ],
              [ 9, 1, "black",     "target", "doorredopened" ]],
  "tip": [ "Puedes utilizar el bucle 'repeat'." ],
  "keyword": [ "Roby", "bucles" ],
  "program": [ "" ],
  "takehomemessage": "ºUtilizar bucles simplifica el cÇdigo!",
  "author": "ToolboX",
  "URL": "http://toolbox.uma.es/",
  "CC": "BY 3.0"
}
```

Figura 7: Ejemplo de tarea para el movimiento de un robot en un mundo simulado.

```
{
  "class": "wordproblem",
  "statement":
     "La Çrbita que describe la Tierra alrededor del Sol es una elipse de
      excentricidad 0.017 y semieje mayor 149.60 millones de kilÇmetros,
      en la que el Sol ocupa la posiciÇn de uno de los focos.
      \begin{center}
      \begin{tikzpicture}
        \filldraw[orange] (-0.33, 0) circle  (5pt);
        \filldraw[blue]   ( 1.2 , 0) circle  (2pt);
        \draw             ( 0   , 0) ellipse (1.2 and 1);
        \par
      \end{tikzpicture}
      \end{center}
      Calcula la mÇxima separaciÇn entre el Sol y la Tierra.",
  "solution": [ 152.14E06 ],
  "tip": [
     "La excentricidad de una elipse viene dada por $ \ellipseeccentricity $.",
     "La ecuaciÇn general de la elipse es $ \ellipseequation $."
  ],
  "keyword": [ "matemÇticas",
               "1º bachillerato" ],
  "program": [ ],
  "takehomemessage": "La GeometrÇa permite realizar cÇlculos fÇsicos.",
  "author": "ToolboX",
  "URL": "http://toolbox.uma.es/",
  "CC": "BY 3.0"
}
```

Figura 8: Ejemplo de definición de tarea incluyendo código LaTeX.

directorio `task/`, con un fichero de definición en formato JSON para cada tarea. `module/` incluye los ficheros de definición de módulos, a partir de los nombres de definición de tareas. `media/` almacena los ficheros multimedia necesarios para la realización de las tareas. El directorio `content/` almacena todos los ficheros susceptibles de ser modificados para adaptar ToolboX a la actividad docente.

En cuanto a las tareas de administración del entorno, se pueden realizar a nivel local accediendo a la estructura de ficheros del sistema desde un terminal Linux (que se inicia en la ventana de instrucciones con `term`). La ejecución del terminal y la alteración del sistema de ficheros puede realizarse sin clave de superusuario, identificándose con la clave de administrador creada durante la instalación de ToolboX. Por último, algunas tareas de mantenimiento, como la actualización de versión o la salida del sistema, pueden realizarse internamente mediante instrucciones (`update` y `shutdown`, respectivamente).[21] Así, vemos que el docente tiene una funcionalidad especial en la gestión de ToolboX, en tanto que el alumno interactúa a un nivel superior, cargando y realizando tareas, exclusivamente.

La implementación de ToolboX explota los recursos de metaprogramación que facilita GNU Octave como lenguaje de programación dinámico.[22] Así, la ejecución de instrucciones Linux y de programas Octave desde el mismo entorno, permiten que ToolboX sea, a un tiempo, un entorno de programación y un intérprete de código Octave.[23] Igualmente, facilita la creación de espacios de trabajo (*workspaces*) independientes, sin que entren en conflicto las variables utilizadas por el usuario con las del entorno. Es de destacar, así mismo, que ToolboX es un programa de código abierto, y técnicamente no podría serlo de otro modo, ya que Octave es un lenguaje exclusivamente interpretado (no se dispone de compilador): cualquier aplicación en Octave es de código abierto, por la propia naturaleza del lenguaje. Es esta misma característica la que le convierte en un buen candidato para el *scripting*, o desarrollo de pequeños programas para tareas muy concretas y que se ejecutan individualmente. Precisamente, éste es el principal objetivo docente de ToolboX: introducir la programación a partir del *scripting*.

5. Conclusiones

Las características de ToolboX le posicionan en un fragmento de utilidades para el aprendizaje de la programación

[21] Todas las instrucciones y elementos sintácticos del lenguaje de programación se notarán con el tipo de letra Consolas.

[22] Los lenguajes dinámicos permiten en tiempo de ejecución comportamientos que otros lenguajes sólo pueden realizar en tiempo de compilación.

[23] Las funciones Octave eval y run permiten ejecutar instrucciones y programas en lenguaje Octave.

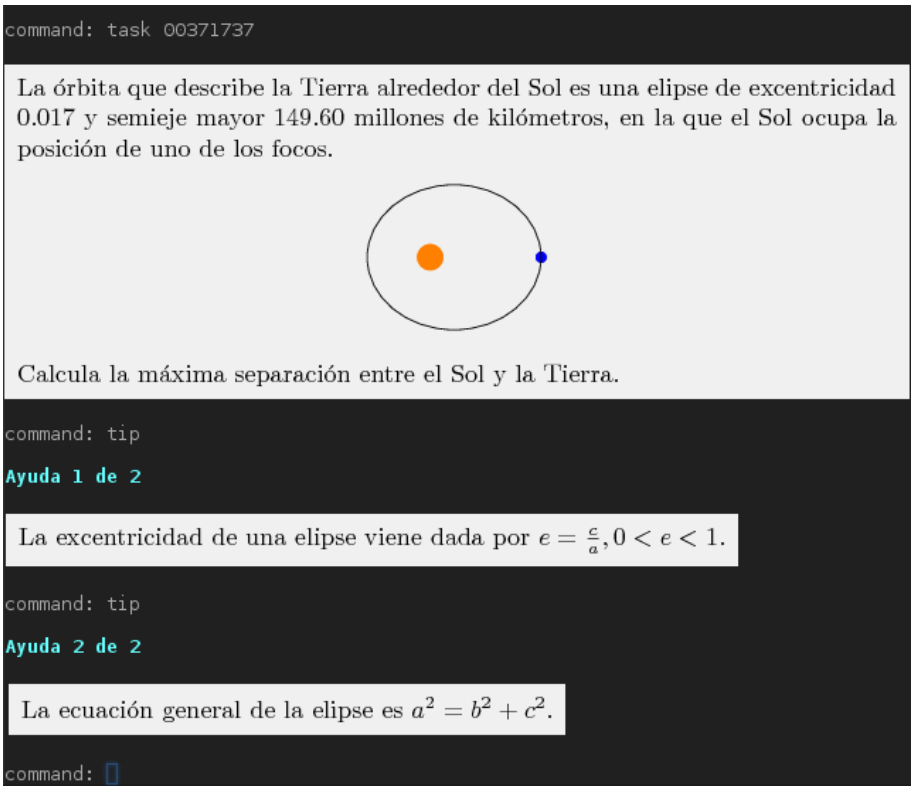

Figura 9: Ejemplo de tarea con ecuaciones y un diagrama en código LaTeX (definida en la figura 8).

```
{
    "integers":              "\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}",
    "positiveintegers":      "\mathbb{Z^+} = \{1, 2, 3, 4, \dots\}",
    "negativeintegers":      "\mathbb{Z^*} = \{-1, -2, -3, -4, \dots\}",
    "nonnegativeintegers":   "\mathbb{Z^*} = \{0, 1, 2, 3, 4, \dots\}",
    "distance2points":       "\sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2}",
    "distance2realnumbers": "\left | a - b \right |",
    "emptyinterval":         "(a,a) = \emptyset \}",
    "degenerateinterval":    "[a,a] = \{ a \}",
    "openinterval":          "(a,b) = \{ x \in \mathbb{R} | a < x < b \}",
    "closedinterval":        "[a,b] = \{ x \in \mathbb{R} | a \leq x \leq b \}",
    "ellipseequation":       "a^2 = b^2 + c^2",
    "ellipseeccentricity":   "e=\frac{c}{a}, 0 < e < 1"
}
```

Figura 10: Extracto del fichero de fórmulas en LaTeX.

```
toolbox

   lib/      bash/         toolboxloop ...
             core/         toolbox.m | execute ...
             board/        makeboard ...
             jsonlab/      jsonload ...
             m/            task | statement | go ...

   addon/    wordproblem/  wordproblem
             puzzle/       puzzle
                           m/     moveright | movedown ...
                           ...

   util/     install/      toolboxinstall ...
             autostart/
             ...

   content/  task/         0000001
                           0000002
                           ...
             module/       trigonometriabasica
                           trigonometriaavanzada
                           conicas
                           ...
             media/        sistemamasamuelle.png
                           trayectoriabalon.png
                           tutorialbucles.webm
                           ...
```

Figura 11: Estructura de archivos de ToolboX: contenido del directorio /opt/toolbox/.

escasamente explotado hasta ahora.[24] El diseño de esta herramienta didáctica está orientado a guiar al usuario en todas las fases del aprendizaje, desde la familiarización con el entorno, hasta el desarrollo de programas complejos (haciendo uso de la vasta librería de funciones de GNU Octave, que abarca campos como el análisis de funciones, el control óptimo, o sistemas de información geográfica,[25] por mencionar sólo algunos ejemplos). Las características de Octave como lenguaje de programación en código abierto, gratuito y dinámico (soporta metaprogramación), y también por su redundancia sintáctica (*legacy syntax*, que permite, por ejemplo, 13 formas diferentes de almacenar datos en una matriz) lo convierten en un candidato idóneo para el aprendizaje de las técnicas de programación.[26] Por otra parte, GNU Octave es la versión abierta de Matlab, el lenguaje de mayor implantación a nivel mundial en ciencia e ingeniería y con una extensa comunidad de usuarios, foros y complementos. En el lado negativo, Octave no es un lenguaje para cómputo intensivo, en cuyo caso se suele utilizar C o Python (incluso Matlab da un rendimiento algo superior a Octave), pero esta limitación no afecta al objetivo de ToolboX, orientado a la realización de programas de escasa complejidad, aparte de ser un lenguaje matricial, notación que acelera extraordinariamente el cálculo sobre grandes volúmenes de datos; con la potencia de los ordenadores actuales, Octave cubre en tiempo real las necesidades de cómputo de cualquier profesional y de estudiantes pre- y universitarios. Pero, sin duda, es la gramática el principal aval de Octave como candidato para el aprendizaje de la programación, ya que elimina las trabas sintácticas que obstaculizan los primeros pasos en el acceso a lenguajes como JavaScript o Python, que arrastran el legado de los lenguajes de programación convencionales, alejándose del lenguaje natural (por ejemplo: importación de librerías, diseño orientado a objetos, o simplemente el uso de símbolos especiales como separadores). Un buen candidato para suceder a Octave, como segunda fase de aprendizaje, es Julia,[27] un lenguaje también dinámico y abierto con un rendimiento comparable a C en la mayoría de test realizados (*benchmarking*), y una comunidad amplia y en crecimiento. No obstante, su sintaxis (similar a la de Octave) incluye elementos que no lo hacen apropiado para la primera toma de contacto del usuario con la programación.

Si bien la elección del lenguaje de programación es crítica, el resto de criterios de diseño incluidos en ToolboX son igualmente esenciales para el éxito de su implantación:

- entorno sencillo, basado exclusivamente en el uso del teclado (sin dispositivos apuntadores o pantallas táctiles), y eliminando todos los elementos externos a la programación (barras de menú y acceso a otros programas);
- implementación sobre la distribución Lubuntu, que permite su utilización en cualquier ordenador, aún con muy escasos recursos de memoria y CPU;
- instalación en disco duro o en memoria USB, para permitir su uso sin interferir con la configuración del ordenador, doméstico o en el ámbito escolar; y
- repositorio de tareas, que incentiven la iniciación a la programación y promuevan su utilización posterior como complemento para otras materias cursadas en la educación primaria y secundaria (incluso en la universitaria).

Todos estos criterios son necesarios para la implantación con éxito de una estrategia de aprendizaje de la programación. Toolbox aún debe considerarse una prueba de concepto, un prototipo cuya competencia para enseñar en todos los niveles educativos e integrarse en los centros está siendo evaluada en el curso 2016–2017 y continuará en los próximos cursos.

6. Discusión

El espectro de soluciones para el aprendizaje de la programación es amplio. Desde juegos de mesa (Robot turtles, Code master o c-jump) a aplicaciones móviles (Daisy the dinosaur o Tynker), pasando por cursos abiertos (Udacity, Open Courseware del MIT) y una amplia oferta de sitios web (Code.org, Scratch 2.0, CodeMonkey, Code Academy, SQLZOO, Treehouse, Khan Academy o CodeCombat). Vídeos (presentados por los gurús de la informática), libros, material en red y, sobre todo, juegos para el aprendizaje interactivo, componen un catálogo diverso y muy completo de soluciones para educadores (tanto en casa, como en el aula); incluso para iniciarse sin la necesidad de mentorización. La mayoría de estas soluciones apuestan por eliminar las barreras de entrada mediante la ludificación: convertir el entorno en un juego, atractivo e intuitivo para el niño (Scratch, Code.org o CodeMonkey). Otras, en cambio, sacrifican este aspecto en favor de la mayor utilidad del lenguaje objetivo (Khan Academy y otras iniciativas proponen el aprendizaje de JavaScript). En ambos enfoques hay un coste inevitable, que o bien afecta a la utilidad de lo aprendido (el lenguaje Scratch está muy limitado fuera del ámbito de la animación; igualmente Code.org, aunque éste ofrece opcionalmente una traducción a JavaScript); o bien impone una curva de aprendizaje difícil de escalar para una mayoría de iniciados (la programación de páginas web es un aspecto limitado de la programación, lo que hace de JavaScript un mal candidato como lenguaje de propósito general[28]).

[24] QtOctave (https://www.openhub.net/p/qtoctave) y Octave Online (https://octave-online.net/) son entornos para la programación en GNU Octave, pero no proporcionan las utilidades y complementos necesarios para el aprendizaje.

[25] Una relación completa de estas librerías puede consultarse en `https://octave.sourceforge.io/packages.php`.

[26] De manera similar, en las décadas de los 70 y 80, BASIC (con todas sus deficiencias como lenguaje, y sin la aridad de FORTRAN) facilitó el acceso a la programación de los primeros ordenadores personales.

[27] `http://julialang.org/`

[28] Al menos, en el lado del cliente; la versión servidor, a través de node.js, proporciona mayor funcionalidad.

También destaca la falta de audacia en este muestrario, que parece conformarse con que el niño conozca los conceptos básicos de la programación,[29] o logre desarrollar un aspecto reducido de las posibles aplicaciones (crear una base de datos, o una página web). Naturalmente, el principal objetivo es proporcionar una toma de contacto, abrir las puertas a aquellos que se sientan más atraídos o capacitados para la programación de ordenadores (los protagonistas del tan anhelado *The next big thing* en los foros de Silicon Valley y *hubs* tecnológicos). Pero, ¿qué ocurrirá con esa gran mayoría que no posee las habilidades, o la vocación, para continuar aprendiendo?[30] Vemos que los niños desarrollan desde corta edad la capacidad de leer un libro o de escribir un documento, habilidades que son importantes para el desarrollo personal y la integración social, y esenciales durante todo el periodo educativo, tanto si termina en la enseñanza obligatoria, como si alcanza el postdoctoral. De igual modo, la enseñanza de la programación no debe orientarse a promocionar la toma de contacto y posterior formación de los desarrolladores, que después engrosarán las plantillas de las EBT y de los cuerpos de funcionarios del sector TIC. Muy al contrario, la programación debe alcanzar a todos, adaptándo la tecnología actual a los sistemas educativos, y proporcionando una herramienta para estimular el aprendizaje y, sobre todo, la capacidad de resolución de problemas. De hecho, el objetivo a largo plazo de ToolboX abarca el perfil completo del programador profesional: la toma de contacto con el sistema operativo Linux[31] y familiarizar al usuario con las técnicas de control de versiones en red, necesarias en el desarrollo profesional de *software*.

Agradecimientos

ToolboX ha nacido en la Universidad de Málaga, a la que le debe el tiempo y los medios dedicados para su desarrollo. Esta aplicación nunca habría sido posible sin GNU Octave 4.0, los componentes de la distribución Lubuntu 16.0.4 y otras aplicaciones de código abierto. La versión actual de ToolboX se debe a colaboraciones estratégicas, entre las que destacan la de Francisco Jiménez Moreno y sus alumnos de la asignatura Proyecto Integrado de Matemáticas, que realizaron las pruebas en alfa en el IES Emilio Prados de Málaga; y a Juan Antonio Ortíz Ramos y sus alumnos del CEIP Intelhorce, como *beta testers*. Igualmente, la versión actual se ha beneficiado de la experiencia de usuarios anónimos de foros (principalmente Stackexchange), que han ayudado a resolver problemas técnicos complejos, acortando los tiempos de desarrollo. El artículo, en su versión actual, incorpora la valiosa contribución de dos revisores anónimos. Por último, si bien no menos importante, merecen mención mis hijas Alba y Laila, por su paciencia para hacerme entender los problemas que tiene la enseñanza de la programación.

Referencias

[1] Carpentier SM, Moreno S, McIntosh AR. (2016). Short-term Music training enhances complex, distributed neural communication during music and linguistic tasks. Journal of Cognitive Neuroscience, 28(10):1603-12.

[2] Fessakis, G, Gouli, E. y Mavroudi, E (2013). Problem solving by 5-6 years old kindergarten children in a computer programming environment: A case study. Computers & Education, 63, 87-97.

[3] Franklin D, Conrad P, Boe B, Nilsen K, et al. (2013). Assessment of computer science learning in a Scratch-based outreach program. In Proceeding of the 44th ACM Technical Symposium on Computer Science Education, 371–376.

[4] García-Peñalvo, F. J., Reimann, D., Tuul, M., Rees, A. y Jormanainen, I. (2016). An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers. TACCLE3 Consortium.

[5] Hsin, CT, Li, MC. y Tsai, CC (2014). The influence of young children's use of technology on their learning: A review. Educational Technology & Society, 17 (4), 85–99.

[6] Hyde KL1, Lerch J, Norton A, Forgeard M, Winner E, Evans AC. y Schlaug G. (2009). Musical training shapes structural brain development. Journal of Neuroscience, 29(10):3019-25.

[7] Joseph S. (2008). Programmer competency matrix. .Net blog of Sijin Joseph [Consulta 15 marzo 2017]. Versión en formato HTML disponible en `http://sijinjoseph.com/programmer-competency-matrix/`.

[8] Kalelioglu, F. y Gülbahar, Y. (2014). The Effects of teaching programming via Scratch on problem solving skills: A Discussion from learners' perspective. Informatics in Education, 13(1), 33-50.

[9] Maloney J, Peppler K, Kafai Y, Resnick M y Rusk N (2008). Programming by choice: urban youth learning programming with Scratch. ACM SIGCSE Bulletin, 40(1):367–371.

[10] Moreno, J. Robles, G. (2014). Automatic detection of bad programming habits in Scratch: A preliminary study. Frontiers in Education Conference, IEEE, 1-4.

[29] En algunos colegios de EEUU, tras la Hour of code se reparte a los niños un panfleto con el eslogan «I'm a coder», con la evidente intención de estimular al niño, aunque también transmite el mensaje de que eso era todo.

[30] Debe recordarse aquí que la población femenina es la más afectada por esta desigualdad, siendo el acceso a las carreras técnicas mayoritariamente masculino, y constituyendo un problema globalizado.

[31] También abierto y que permite un control total de los sistemas informáticos, tanto clientes como servidores.

[11] Poss, R. (2014) How good are you at programming? A CEFR-like approach to measure programming proficiency. [Consulta 15 marzo 2017]. Versión en formato HTML disponible en http://science.raphael.poss.name/programming-levels.html. Versión en formato PNG disponible en http://science.raphael.poss.name/programming-levels/prog-skill-matrix.png.

[12] Resnick M, Maloney J, Monroy-Hernández A, Rusk, N; Eastmond E, Brennan K, Millner A, Rosenbaum E, Silver J, Silverman B y Kafai Y (2009). Scratch programming for all. Communications of the ACM, 52(11), 60-7.

[13] Roque R, Dasgupta S y Costanza-Chock S (2016). Children's civic engagement in the Scratch online community. Social Sciences, 5(4), 55.

[14] Tantau, T. (2015). The TikZ and PGF Packages. Manual for Version 3.0.1a. Versión en PDF disponible en http://www.texample.net/media/pgf/builds/pgfmanual_3.0.1a.pdf.

Francisco Vico es catedrático de Ciencias de la Comptuación e Inteligencia Artificial en la Universidad de Málaga, a la que está vinculado desde 1992. Es profesor de Teoría de Autómatas y Lenguajes Formales en la ETS Ingeniería Informática e investigador en las áreas de modelado biológico, vida artificial y creatividad artificial.