https://doi.org/10.17979/spudc.9788497498081.1148

DOCTUS, UNA HERRAMIENTA DE E-LEARNING INNOVADORA PARA EDUCACIÓN EN AUTOMÁTICA Y ÁREAS AFINES

Fabio Gómez-Estern
Universidad Loyola Andalucía
C/ Energía Solar 1, 41014 Sevilla, España.
fgestern@uloyola.es

David Muñoz de la Peña
Universidad de Sevilla
Camino de Los Descubrimientos S/N, 41092 Sevilla, España.
dmunoz@us.es

Carlos Sánchez Cazorla
Universidad de Sevilla
Camino de Los Descubrimientos S/N, 41092 Sevilla, España.
cscazorla@gmail.com

Resumen

En este trabajo se presenta una nueva herramienta para su uso en docencia en asignaturas del área de Ingeniería de Sistemas y Automática, y por extensión en cualquier materia científico-técnica. Doctus es una versión completamente renovada de Goodle GMS, una aplicación que en los últimos años ha cobrado relevancia en entornos académicos con la que se ha tratado de llevar el concepto de evaluación automática mucho más allá de los paradigmas tradicionales recogidos en Moodle y herramientas afines. La extensa base de datos de ejercicios creados hasta la actualidad en Goodle y compatibles con el nuevo Doctus, permite un acceso rápido a las funcionalidades de autocorrección. Además de presentar una arquitectura software modular orientada a objetos y una interfaz mejorada, Doctus posee una serie de características innovadoras que resultan de interés práctico en el área de Automática.

Palabras Clave: Evaluación automática, educación en control automático, e-learning, control de sistemas lineales.

1 INTRODUCCIÓN

La plataforma de e-learning Goodle GMS [5,6,7] es una herramienta consolidada y popular en el área de Ingeniería de Sistemas y Automática en España y el extranjero. Ha sido traducida al francés y al inglés, y actualmente ha sido empleado en más de 100.000 pruebas de evaluación, según reflejan sus registros. Inicialmente desarrollada en 2007 en la arquitectura PHP-MySQL-Apache por profesores del área, ofrecía una interfaz sencilla en la que se podía evaluar código compatible MATLAB escrito por los alumnos como solución a un problema técnico o científico planteado. El código MATLAB de cada alumno era anexado a un código suministrado por el profesor, que a su vez se encarga de verificar si la solución proporcionada por el estudiante era correcta, mediante la observación de las variables creadas en MATLAB (que se ejecutaba en una instancia de servidor) como consecuencia de la ejecución del código del alumno.

En definitiva, la primera versión de Goodle GMS ofrecía un *sandbox* de evaluación automática en el que el docente podía definir con libertad programática el proceso de evaluación, aprovechando toda la potencia de cálculo de MATLAB/Simulink. Además, disponía de ciertas características que lo hacían especialmente útil en docencia: capacidad de personalización de los ejercicios basada en el documento nacional de identidad, filtros de sintaxis, plantillas para simplificar la tarea del estudiante, comandos prohibidos, diversos modos de ejecución, etc.

Posteriormente, Goodle sufrió varias revisiones: una revisión mayor del código en 2009 en la que se rehizo el programa con base en un framework PHP (modelo vista-controlador) y se profesionalizó el interfaz, más una serie de revisiones incrementales en las que se fueron añadiendo funcionalidades y mejoras progresivas. Entre estas mejoras destacaron la evaluación en modo competitivo, la generación de enunciados personalizados automáticamente, la adaptación a lenguajes como Java y C++ (para su uso en asignaturas de programación) y el trabajo con Simulink y la Symbolic Toolbox de MATLAB. En este período destacamos, además, dos contribuciones relevantes en el contexto de la educación en

Automática: i) el desarrollo de una interfaz XML para permitir evaluar el trabajo realizado con laboratorios virtuales (por ejemplo Easy Java Simulations) y remotos, de creciente importancia docente e investigadora en la última década, y una extensión dedicada a la generación automatizada de problemas personalizados de diseño de controladores lineales, y sus correspondientes evaluadores automáticos. Ambas contribuciones se enmarcaron en el desarrollo de sendas Tesis Doctorales defendidas en 2012 y 2015 [2,9].

En 2015, tras su uso regular por un importante conjunto de profesores de diversas áreas, y gracias a la experiencia acumulada y los informes de uso recibidos, se ha considerado el momento de realizar una modernización radical de la plataforma. Se ha abordado la tarea en el marco de un proyecto de innovación docente financiado por la Escuela Técnica Superior de Ingeniería de la Universidad de Sevilla, con una modificación posterior dentro de otro proyecto de innovación de la Universidad Loyola Andalucía. La revisión del sistema estaba inicialmente destinada a mejorar la arquitectura software sacando partido de los enormes avances en los frameworks *Open Source* destinados al desarrollo de aplicaciones web multiusuario y multiplataforma, aunque en el proceso se ha aprovechado la oportunidad para introducir nuevas funcionalidades que consideramos suficientemente relevantes para considerarlo una aplicación nueva y novedosa.

En el resto del trabajo describiremos las innovaciones relacionadas con la plataforma en relación con la educación en Automática, empezando por un recordatorio de la filosofía de funcionamiento de Doctus, heredada de Goodle GMS. A continuación presentaremos una descripción de la nueva arquitectura software y de aspectos del interfaz gráfico, por su interés y actualidad desde el punto de vista técnico. Tras ello, describiremos la nuevas funciones de generación de enunciados automatizados, que consideramos fundamental en asignaturas de control con problemas parametrizables.

En la misma sección, enlazando con los enunciados personalizados, presentaremos un nuevo sistema de evaluación basado en Excel, destinado a operaciones sencillas, que palia la necesidad de saber programar en MATLAB para crear un ejercicio en Doctus. Aunque excede el alcance de este trabajo, cabe mencionar otra funcionalidad de Doctus en la que se plasma la aspiración a convertir en una herramienta de calado profundo en la educación: un sistema de seguimiento y análisis de consecución de competencias formativas definidas en los planes de estudio del EEES.

Finalmente, se ilustrará el uso de Doctus en Automática de forma conjunta con una aplicación docente de gran popularidad -las herramientas interactivas para el aprendizaje del control automático desarrolladas por los autores de [4], basadas en Sysquake. En esta aplicación concreta se propone el aprendizaje del ajuste mediante lugar de las raíces mediante dichas herramientas y evaluadas con Doctus.

2 EVALUACIÓN AUTOMÁTICA EN DOCTUS

En esta sección describiremos las generalidades de la nueva aplicación, empezando por explicar el objetivo fundamental: la evaluación automática.

2.1 ¿Por qué usar evaluación automática?

La evaluación automática en aplicaciones de e-learning es un recurso sumamente valioso y usado en la actualidad. Con frecuencia creciente, se emplean clickers, tests online, plataformas como Moodle y otros recursos para obtener una medida inmediata del desempeño de los estudiantes. Los docentes son cada vez más conscientes del valor del aprendizaje y la evaluación continua, tanto para monitorizar y apoyar el progreso de los estudiantes como para medir la efectividad de las actividades formativas a la hora de transmitir conceptos y entrenar habilidades.

La evaluación automática puede ser considerada desde un punto de vista restrictivo, es decir, como evaluación numérica pura, en cuyo caso está sometida a críticas en el sentido de prescindir del criterio humano y de la proximidad al alumno en la formación. O en sentido amplio, es decir, como elemento facilitador de nuevas metodologías docentes, en la que la evaluación solo es una parte de un ciclo de enseñanza-aprendizaje-ensayo-corrección.

En este segundo sentido se han propuesto numerosas estrategias formativas basadas en Goodle GMS [3], la predecesora de Doctus. Concretamente, se han desarrollado laboratorios virtuales en los que el alumno interacciona con un sistema simulado, y en el proceso genera información que puede ser analizada en Matlab bajo la ejecución de un script desarrollado por el docente; se han desarrollado interfaces interactivas para el trabajo con sistemas dinámicos no lineales y funciones de Lyapunov, sistemas de evaluación de algoritmos desarrollados por el estudiante, sistemas competitivos, sistemas cooperativos, evaluación automática personalizada basada en clickers, Excel, etc.

2.2 Arquitectura de evaluación automática

En todos estos casos la filosofía de base es un modelo de caja negra que se describe en unos sencillos pasos:

- El sistema ofrece una interfaz web para el alumno y otra para el profesor.
- Los elementos de ambas interfaces son principalmente texto plano.
- El profesor crea un ejercicio y se lo asigna a un grupo de alumnos. La creación del ejercicio requiere la redacción de un script, llamado *evaluador*. Además, ha de preparar un enunciado con instrucciones precisas.
- El alumno lee el enunciado, y determina las respuestas con el conocimiento de la asignatura. Estas serán entregadas en formato texto en el interfaz web. En el caso más sencillo, la respuesta será un conjunto de valores numéricos. Por ejemplo, supongamos que el alumno ha de proporcionar la masa y la longitud de un cuerpo como solución al problema. Para ello escribiría:

```
m=23.7;
l=14.8;
```

- En un interfaz clásico, esto requeriría dos campos de texto. Sin embargo, en Doctus, el formato de entrega debe ser flexible para acoger cualquier tipo de respuestas, y por tanto se emplea un simple campo de texto multilínea en el que podemos introducir asignaciones (como el ejemplo), comandos, y cualquier operación compatible con la sintaxis Matlab.
- Una vez realizadas las entregas, se almacenan en la base de datos para su evaluación posterior. Cuando el docente lo desee, lanzará un ciclo de evaluación en el que se verificará si la solución es correcta mediante la ejecución encadenada del script del alumno y del script *evaluador*. En el ejemplo anterior, un posible evaluador tomaría la forma siguiente:

```
nota=0;
copmentarios='';
if m==23.7
    comentarios=strcat(comentarios, 'Masa
correcta<br>');
    nota=nota+5;
end
if l==14.8
    comentarios=strcat(comentarios,
'Longitud correcta<br>');
    nota=nota+5;
end
```

En el ejemplo mostrado es fácil observar la lógica de la plataforma Doctus. Tras la ejecución encadenada de la entrega del alumno y el *evaluador* en el servidor, se crean dos variables en el *workspace* de Matlab, que serán recuperadas por el sistema web para su almacenamiento en la base de datos y su visualización por parte del alumno:

a) Variable *nota*. Contiene un valor numérico calculado incrementalmente a partir de todos los apartados correctos del ejercicio.
b) Variables *comentarios*. Contiene un informe textual que se mostrará el alumno con detalles sobre el proceso de evaluación. En este caso se ha empleado para detallar cuáles apartados fueron respondidos correctamente. Esta variable de cadena puede contener etiquetas HTML, que servirán para formatear la visualización de los informes en el navegador.

2.3 Paradigmas de uso.

Lejos de estar agotado el modelo con la descripción realizada, a partir de este punto se abren una serie de posibilidades, que han ido desarrollándose en los últimos años, y que extienden el uso de Doctus a un número amplio de aplicaciones. Para comprender las opciones existentes, se muestra una tabla con las funcionalidades disponibles.

Tabla 1: Modos de funcionamiento de Doctus.

Característica	**Opciones**
Formatos de entrega textual (se entrega en la web de Doctus)	Matlab
	C/C++
	Java
Formatos de entrega no textual (se emplea una aplicación cliente para resolver el ejercicio planteado)	Easy Java Simulations (vía XML)
	Simulador 3D industrial (vía XML)
	Plantillas Excel (entrega de fichero en navegador)
Modos de cálculo de la nota	Individual (cada alumno tiene su calificación independiente del resto)
	Competitiva (la nota se calcula en función de la calidad de la solución comparada con el resto del grupo)
Tipos de problemas	Numérico de valor exacto (ver ejemplo anterior)
	Numérico aproximado
	Numérico evaluado por simulación*
	Algorítmico (el alumno entrega un programa o función)
Tipo de enunciado	Estático
	Dependiente del DNI
	Dependiente del DNI con texto de enunciado variable

Nota: (*) En el problema numérico evaluado por simulación la solución no es única, sino que ha de ser

evaluada en el contexto de una simulación o análisis complejo. El resultado de la simulación dará una medida de la calidad de la solución adoptada. Como ejemplo paradigmático cabe considerar el diseño de controladores lineales, basado en constantes (solución numérica), que han de ser evaluados mediante simulaciones lineales.

2.4 Arquitectura Software

La arquitectura que implementa las funcionalidades resumidas en la Tabla 1 se ilustra mediante la Figura 1. En ella destacamos:

a) Bloques de origen de información (color azul oscuro). Representan las entregas de los alumnos en distintos formatos (textuales y no textuales, ver Tabla 1) en la zona izquierda, y la creación de material por parte del profesor (zona derecha).
b) Bloques de código de alumno (color naranja). Dependiendo de la aplicación origen, el código lo puede escribir el alumno o se puede generar automáticamente en el seno de una aplicación o laboratorio virtual. En este segundo caso, es preciso pasar por un bloque de generación de código automático.
c) Bloques de generación de código automático (color violeta). Se trata de plugins insertados en las aplicaciones clientes y laboratorios virtual, capaces de registrar el trabajo en el laboratorio del alumno, enviarlos en formato XML por procedimiento remoto y convertir los datos en asignaciones Matlab, para que se procesen en el esquema de ejecución encadenada.
d) Sistema de ejecución secuencial (*Sequential execution* al pie a la izquierda de la figura), en el que se funde el código proporcionado por el profesor (azul claro) y el entregado por el alumno (oscuro). En dicho bloque se gestiona también la individualización de los parámetros en función del documento nacional de identidad.

La funcionalidad Excel no aparece reflejada en este esquema, ya que no se basa estrictamente en la ejecución encadenada de código, y se describirá más adelante.

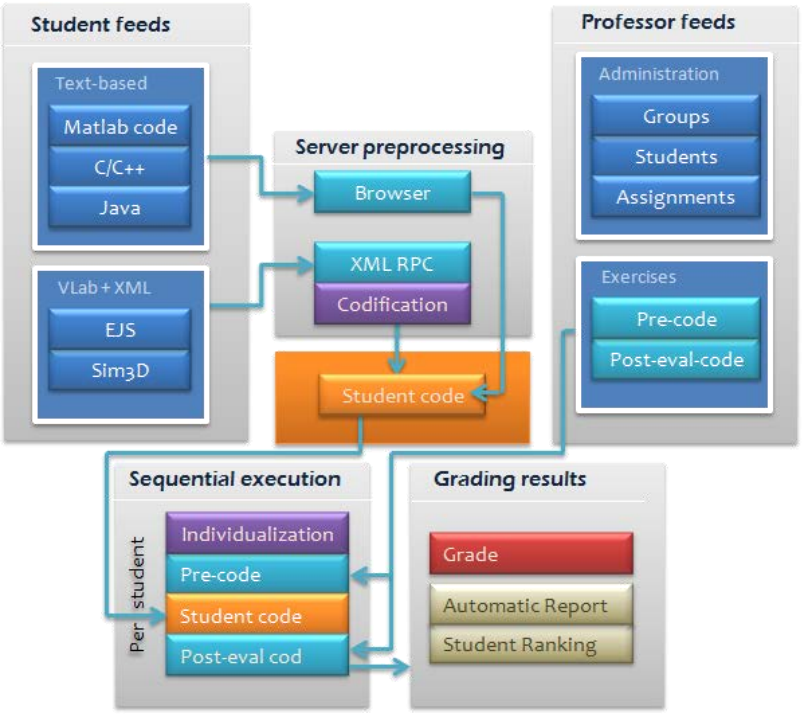


Figura 1: Arquitectura de la aplicación

2.5 Implementación

La arquitectura descrita se ha programado de nuevo apenas empleando código *legacy* de la aplicación predecesora. El objetivo es basar el código en una arquitectura potente basada en un *framework* PHP capaz de gestionar los accesos a bases de datos y al interfaz web con clases abstractas.

En concreto, se ha hecho uso del framework de software libre *Laravel*, con un gestor de código denominado *PHP artisan*, y un gestor de versiones

basado en *GitHub* capaz de mantener actualizadas todas las versiones del servidor mediante una instancia compartida en la nube, manejando un sistema de versiones y subversiones que permite el desarrollo en paralelo de funcionalidades por distintos programadores.

Independientemente del *framework* utilizado, el desarrollo da lugar a una aplicación PHP-MySQL-Apache completamente portable. Actualmente corre en un servidor Windows 2012 Server con una instancia de Matlab automatizada mediante el protocolo ActiveX. La seguridad está a cargo del sistema SSL de Apache, y de los sistemas de autenticación por clave asimétrica y de protección de rutas de Laravel, por lo que la seguridad queda en un nivel muy superior a lo disponible hasta ahora.

La opción de instalar el sistema en un entorno Microsoft se ve justificada por el uso remoto de Matlab mediante ActiveX, aunque si se empleara otro método de llamada a procedimiento remoto no ActiveX –propietario de Microsoft-, la migración a otros sistemas como Linux sería inmediata.

Figura 2: Ventana de login en Doctus

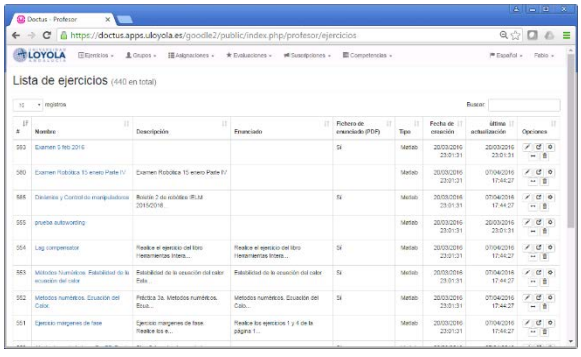


Figura 3: Pantalla de listado de ejercicios usando la plantilla de visualización Bootstrap

La filosofía de implementación ha seguido en todo momento tres principios:

a) Modularidad y facilidad de creación de nuevos tipos de ejercicios y funcioanlidades.
b) Empleo de herramientas Open Source,
c) Interfaz adaptable a distintos tipos de dispositivos.

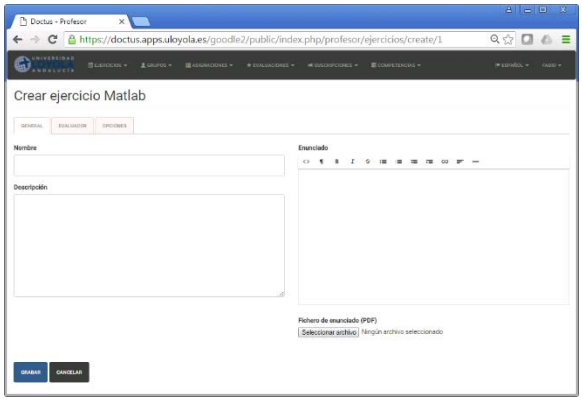


Figura 4: Pantalla de creación de ejercicios de tipo MATLAB usando la plantilla de visualización Sandstone

3 NUEVAS FUNCIONALIDADES DOCTUS

En esta sección describiremos las novedades funcionales aportadas por Doctus respecto a su predecesor, Goodle GMS. Concretamente, se describirá el sistema de enunciados automáticos y la interfaz Excel. Una novedad adicional, el sistema gestión de competencias de aprendizaje, quedará fuera del alcance de este artículo para no excedernos en su longitud.

3.1 Enunciados automáticos

Esta función se empezó a probar en las últimas versiones de Goodle GMS. La idea fundamental consiste en abordar la parametrización de los ejercicios en función del número de documento de identidad del alumno.

En el enfoque tradicional, un alumno tiene que un ejercicio en el que los parámetros dependen de su DNI. Un ejemplo de ello se puede ver en la figura 5, perteneciente a un ejercicio de Teoría de Máquinas y Mecanismos de la Universidad de Sevilla. En dicho ejercicio, el alumno ha de calcular una serie de parámetros específicos para su ejercicio.

Doctus permite simplificar el proceso de personalización mediante el uso de ejercicios automatizados, de dos maneras: basada en HTML y basada en TeX-PDF. En el primer caso, el docente dispone de un campo de texto en el interfaz de creación de ejercicios, donde puede emplear código MATLAB para crear una cadena de texto dinámica

(dependiente del DNI mediante fórmulas similares a las de la figura 5), que se muestre en pantalla cuando se conecta. Por ejemplo, para implementar un enunciado que calcule el parámetro ξ de la figura 5, se escribiría el siguiente código:

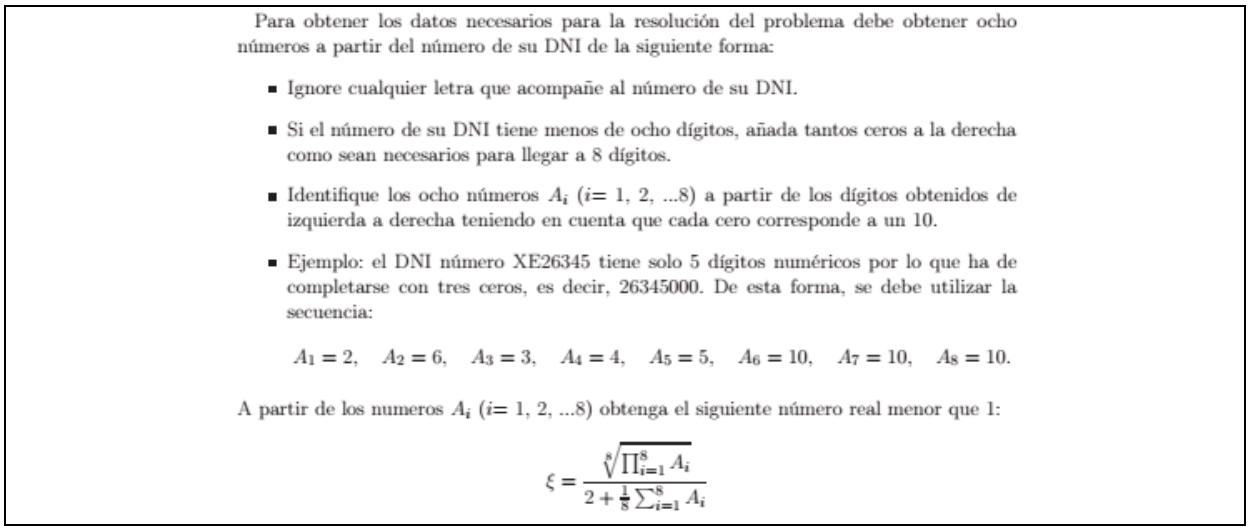
```
A=zeros(1,8);
for indice=1:8
    A(i)=str2num(dni(indice));
end

xi=prod(A)^(1/8)/(2+1/8*sum(A));
htmlautowording=sprintf('<br> El valor de
su parámetro personalizado es: %f <br>',
xi);
```

Como se puede apreciar, la variable `htmlautowording` puede ser creada dinámicamente, usando entre otras cosas la variable `dni` que identifica al alumno (introducida por el sistema). Dicha variable contendrá la cadena HTML que se mostrará al alumno correspondiente al conectarse al sistema.

La segunda opción corresponde a la creación de enunciados PDF, es más sofisticada y produce resultados más atractivos. En resumen, su lógica de funcionamiento es la siguiente:

- El profesor prepara un enunciado TeX con toda la complejidad tipográfica que desee.
- En dicho enunciado el docente inserta una serie de etiquetas en los lugares donde desea que aparezcan los valores personalizados.
- Dichas etiquetas serán procesadas por Doctus para después compilar y generar el enunciado definitivo y en PDF. Las etiquetas serán asociadas a variables MATLAB creadas en el código del profesor de manera similar a como se creó la etiqueta `htmlautowording` en el caso anterior.
- El fichero PDF personalizado queda visible para el alumno, quien no tiene que hacer ningún cálculo para la personalización.



Para obtener los datos necesarios para la resolución del problema debe obtener ocho números a partir del número de su DNI de la siguiente forma:

- Ignore cualquier letra que acompañe al número de su DNI.
- Si el número de su DNI tiene menos de ocho dígitos, añada tantos ceros a la derecha como sean necesarios para llegar a 8 dígitos.
- Identifique los ocho números A_i (i= 1, 2, ...8) a partir de los dígitos obtenidos de izquierda a derecha teniendo en cuenta que cada cero corresponde a un 10.
- Ejemplo: el DNI número XE26345 tiene solo 5 dígitos numéricos por lo que ha de completarse con tres ceros, es decir, 26345000. De esta forma, se debe utilizar la secuencia:

$$A_1 = 2, \quad A_2 = 6, \quad A_3 = 3, \quad A_4 = 4, \quad A_5 = 5, \quad A_6 = 10, \quad A_7 = 10, \quad A_8 = 10.$$

A partir de los numeros A_i (i= 1, 2, ...8) obtenga el siguiente número real menor que 1:

$$\xi = \frac{\sqrt[8]{\prod_{i=1}^{8} A_i}}{2 + \frac{1}{8}\sum_{i=1}^{8} A_i}$$

Figura 5: Ejemplo de un ejercicio personalizado con parámetros basados en el DNI.

3.2 Evaluación en Excel

La experiencia en los cursos impartidos sobre Goodle GMS ha llevado a los autores de este trabajo a considerar la posibilidad de reducir o eliminar la tarea de programación del profesor, permitiendo su uso a un público más amplio. Por primera vez, se implementa una funcionalidad de evaluación automática sin necesidad de escribir código con Doctus. Para ello se usará Excel tanto como plantilla para que el alumno entregue sus soluciones como motor de evaluación y de personalización de los ejercicios.

La idea principal supone que todo ejercicio autoevaluable consiste en una serie de valores de entrada (parámetros del ejercicio) que pueden ser personalizado, y las soluciones consisten en una serie de valores de salida que dependen de los valores de entrada con una relación matemática realizable con fórmulas Excel. En estas condiciones, el profesor debe rellenar una plantilla de autoevaluación en la que se especifica:

- Para cada parámetro de entrada, un valor fijo o, alternativamente, un intervalo dentro del cual el sistema escogerá un valor diferente para cada alumno en cuestión.
- Para cada valor de salida, insertará una fórmula haciendo referencia a los parámetros de entrada que se hayan asignado al alumno.

Para ello, se proporciona al profesor una plantilla Excel como se muestra en la figura 6. En ella se ofrece

una cantidad de filas indefinida para rellenar con los parámetros de entrada p1, p2, etc. (columnas B a H), sus rangos (en el caso que sean aleatorios) y las cifras decimales admitidas. En las columnas M a Q, se introducen los valores solución (s1, s2, etc) expresados como fórmulas Excel en función del valor contenido en las celdas de los parámetros de entrada. Además, se configura la tolerancia de error en las respuestas y el peso (nota) de cada apartado.

El sistema funciona del siguiente modo:

- El profesor crea la plantilla, definiendo los rangos de los parámetros de entrada y el procedimiento de cálculo de la solución. Sube el fichero Excel modificado al servidor Doctus y asigna el ejercicio a los alumnos.

- El alumno descarga el enunciado del ejercicio en formato .xls. Este enunciado contendrá los parámetros personalizados del alumno y no será igual que el de sus compañeros. El alumno realizará los cálculos necesarios, insertará las soluciones en las celdas destinadas a ello, y subirá el fichero .xls resultante a Doctus.
- El profesor lanzará la evaluación de los ejercicios entregados. El proceso de evaluación realiza una fusión entre la hoja del alumno (de la que extrae sus parámetros personalizados y sus soluciones) y la del profesor (de la que extrae las fórmulas de cálculo de las soluciones correctas a partir de los valores personalizados).
- El resultado es una nota calculada automáticamente a partir de la comparación de las soluciones del alumno con las correctas, en la hoja Excel fundida, que quedará almacenada en la base de datos.

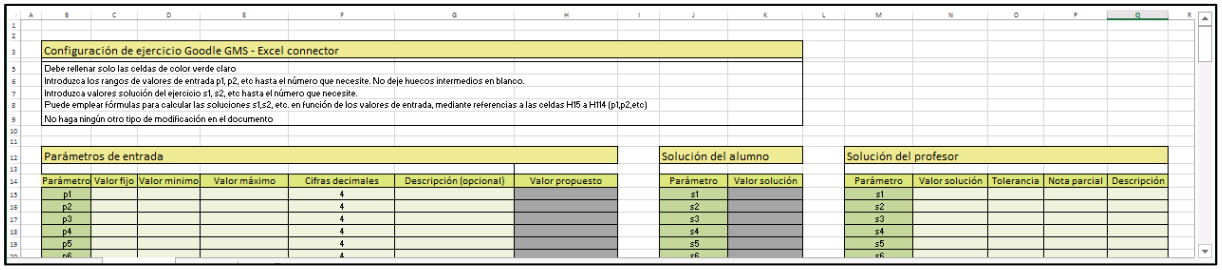


Figura 6: Vista parcial de la plantilla de creación de ejercicios Excel autoevaluados.

4 EXPERIENCIA EN EL AULA: EXPLICANDO EL LUGAR DE LAS RAÍCES CON DOCTUS

A modo de ilustración de las capacidades de Doctus, y sus aplicaciones en el área de Automática, se ilustrará un ejercicio clásico de un curso de Regulación Automática: el ajuste por el lugar de las raíces [8]. Por su extensión, se reducirá su presentación a algunas indicaciones breves.

Una manera de estudiar el lugar de las raíces y otros temas de Control Automático es mediante las herramientas interactivas presentadas en [4]. Concretamente, se empleó la herramienta 6.1 *Lugar de las raíces,* presentada en la figura 7. Esta herramienta permite trabajar sobre el gráfico del lugar de las raíces (abajo a la izquierda) de uno de los sistemas disponibles en el menú. Además, permite elegir la ganancia estática K del sistema en bucle abierto. El efecto de variar la ganancia del controlador (*k* minúscula) se observa mediante unos marcadores que nos indican en qué punto de las ramas del gráfico se encuentran los polos de bucle cerrado para cada valor concreto de *k*. Esto permite analizar y preguntar a los alumnos los intervalos de *k* para los que se cumplen las siguientes condiciones:

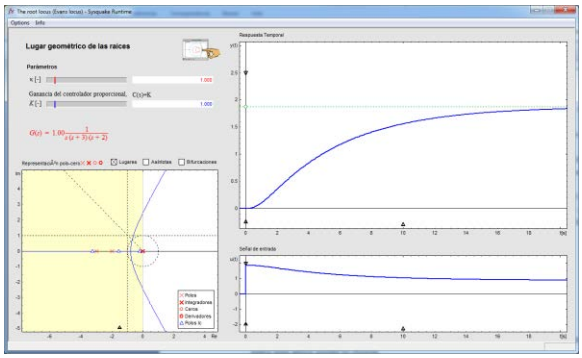


Figura 7: Herramienta interactiva *Lugar de las raíces.*

- El sistema es estable (o inestable).
- El sistema es sobreamortiguado (o subamortiguado).

- El sistema tiene un tiempo de pico inferior a un valor determinado.
- El sistema tiene una sobreoscilación inferior a un valor determinado.
- El sistema tiene una sobreoscilación y un tiempo de subida inferiores a unos valores determinado.

Todas estas cuestiones se responden proporcionando un intervalo de *k* (o el intervalo vacío). La oportunidad surge del hecho de que variando la ganancia estática del sistema en bucle abierto, los intervalos que responden al problema varían, por lo que el ejercicio se puede personalizar con sólo cambiar el parámetro K. Además, el evaluador automático es de sencilla implmentación: conociendo los intervalos solución a las preguntas para un valor de K (ganancia estática) se puede inferir los intervalos correspondientes a cualquier otra K.

Para analizar la efectividad del sistema, se creó un gráfico de dispersión entre la media de las calificaciones obtenidas en promedio a lo largo de los distintos ejercicios Doctus planteados en el curso, y el examen parcial. La figura 8 ilustra los resultados.

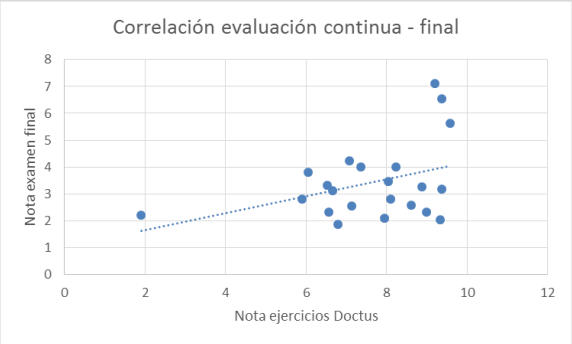


Figura 8: Correlación entre las calificaciones obtenidas con Doctus y el examen final.

5 CONCLUSIONES

En este trabajo se ha presentado una nueva herramienta de e-learning con interesantes aplicaciones en el área del Control Automático. Se han detallado los principios de funcionamiento, las novedades fundamentales respecto a sus antecesoras, y se ha ilustrado un ejemplo de uso en clase. En un trabajo futuro se ilustrará la función de Learning Analytics incorporada al nuevo Doctus, mediante la cual se pretende hacer un seguimiento y predicción del proceso de aprendizaje por competencias.

Agradecimientos

Los autores agradecen a David Becerra Alonso, de la Universidad Loyola Andalucía, pos sus útiles aportaciones a la arquitectura de enunciados personalizados aquí descrita.

Referencias

[1] Dorf, R.C. and Bishop, R.H., (2005) Sistemas de Control Moderno 10ED, Pearson, Inglaterra.

[2] G. Farias, F. Gomez-Estern, L. De la Torre, D. Muñoz de la Peña, C. Sánchez, S. Dormido. Enhancing Virtual and Remote Labs to Perform Automatic Evaluation. 9th IFAC Symposium Advances in Control Education. 2012.

[3] F. Gómez-Estern, M. López-Martínez, D. Muñoz de la Peña. Sistemas de Evaluación Automática Vía Web en Asignaturas Prácticas de Ingeniería. Revista Iberoamericana de Automática e Informática Industrial. Vol. 7. Núm. 3. 2010. Pag. 111-119.

[4] Guzmán, J.L, Costa, R., Berenguel, M., Dormido, S., (2012) Control automático con herramientas interactivas, Pearson, Inglaterra.

[5] M. López-Martínez, F. Gómez-Estern, D. Muñoz de la Peña. Automatic Web-Based Evaluation of C-Programming Exercises in Engineering Education. International Journal for Knowledge, Science and Technology. Vol. 1. Núm. 2. 2010. Pag. 1-6.

[6] A. Mu oz de la P a, D. Gon lez- mez, D. Mu oz de la P a, F. mez-Estern, and M. chez. Automatic Web-Based Grading System: Application in an Advanced Instrumental Analysis Chemistry Laboratory, Journal of Chemical Education Vol. 90, Pag. 308-314, 2012.

[7] D. Muñoz de la Peña, F. Gómez-Estern, S. Dormido. A new Internet tool for automatic evaluation in Control, Systems and Programming, IEEE Computers & Education Vol. 59, Pag. 535-550, 2012.

[8] Ogata, K., (2011) Ingeniería de Control Moderna 5ED, Pearson, Inglaterra.

[9] C. Sánchez, F. Gómez-Estern, D. Muñoz de la Peña. A virtual lab with automatic assessment for nonlinear controller design exercises. 9th IFAC Symposium Advances in Control Education. 2012.