"Complemento práctico en la enseñanza de lenguajes formales usando las expresiones regulares en el contexto de aplicaciones web"

Jose Miguel Blanco Arbe y Ana Sanchez Ortega

UPV/EHU / LSI / TR 05-2007

Complemento práctico en la enseñanza de lenguajes formales usando las expresiones regulares en el contexto de aplicaciones web

José Miguel Blanco Arbe, Ana Sánchez Ortega
Dpto. de Lenguajes y Sistemas Informáticos
Facultad de Informática - Universidad del País Vasco
Pº Manuel Lardizabal 1, 20018 San Sebastián
{josemiguel.blanco, ana.sanchez}@ehu.es

Resumen

Tradicionalmente la enseñanza de los autómatas y lenguajes formales basa su principal aplicación práctica en la construcción de compiladores. Sin embargo, las tareas de diseño y programación necesarias son excesivamente complejas como para que los estudiantes, que están cursando el tercer cuatrimestre de la Ingeniería, puedan abordarlas con el rigor necesario. Es posible incorporar otro enfoque práctico, real y más actual de las expresiones regulares en estas asignaturas, aprovechando su frecuente uso como herramienta de especificación de patrones a la hora de diseñar formularios de entrada de datos en diferentes contextos y, particularmente, en aplicaciones web de tres capas. El hecho de trabajar esta competencia junto con el desarrollo teórico de las expresiones regulares permite a los estudiantes ser conscientes de la importante utilidad práctica de este concepto, sin restringirlo a otros usos más clásicos relacionados con el diseño de procesadores de textos o analizadores léxicos.

Durante el curso 2006-07 se ha propuesto a los estudiantes de Ingeniería Técnica en Informática de Sistemas de la Universidad del País Vasco *desarrollar fragmentos de código basados en una notación formal para resolver problemas de reconocimiento de patrones*. La experiencia se ha llevado a cabo utilizando concretamente la notación, inspirada en las expresiones regulares, de JavaScript, resultando viable, efectiva y bien valorada por parte de los estudiantes.

1. Introducción

Autómatas y Lenguajes Formales (ALF) es una asignatura troncal en la Ingeniería Técnica de Informática de Sistemas. Sus contenidos teóricos están bien determinados y se incluyen dentro de las ramas de la Informática Teórica. Los temas que se tratan en ella están relacionados, por un lado, con la Jerarquía de lenguajes de Chomsky y, por otro, con los límites de los sistemas informáticos. Se estudian los reconocedores (autómatas) y generadores (gramáticas y expresiones regulares) de cada clase de lenguajes, junto con los algoritmos de equivalencia entre distintos modelos, y también la existencia de problemas no computables. La mayor parte de la bibliografía, incluso la más actual [2], sigue un enfoque teórico lo que, en el contexto de los estudios de Ingeniería en Informática, no facilita la motivación del alumno. Desde nuestro punto de vista, el hacer compatible una sólida aproximación teórico con una orientación hacia el uso de los conceptos en la resolución de problemas prácticos y reales (buscando un cierto equilibrio entre teoría y aplicabilidad), resulta en un enfoque más adecuado, sobre todo en una Ingeniería Técnica.

Con mucha frecuencia las aplicaciones prácticas de los conceptos teóricos de ALF suelen relacionarse con la construcción de compiladores. Sin embargo, las tareas de diseño y programación necesarias son suficientemente complejas como para ser difícilmente planteables a estudiantes que aún están cursando el tercer cuatrimestre de la Ingeniería Técnica, y por tanto no han tenido tiempo de asimilar muchos conceptos necesarios de estructuras de datos y algoritmos. Además, tanto por su disponibilidad como software de base, como por la magnitud y complejidad del desarrollo de la ingeniería de los compiladores, la construcción de los mismos queda fuera del ámbito de actuación (y, por lo tanto de motivación) de la inmensa mayoría de los estudiantes de ALF. Estas razones han limitado tradicionalmente la función motivadora del

enfoque práctico basado en la construcción o especificación de analizadores léxicos. Como limitación añadida, dada la complejidad de esta tarea, acaba simplificándose de una manera difícilmente compatible con el rigor necesario en la asignatura.

Tras el análisis anterior decidimos explorar otros enfoques prácticos y sencillos de implantar para algunos de los conceptos que se trabajan en ALF, atendiendo a que las expresiones regulares, como descriptores de lenguajes regulares, son enormemente útiles y los estudiantes no siempre son conscientes de sus posibilidades de uso. Con este objetivo, nuestra propuesta se basa en que los estudiantes utilicen las expresiones regulares en un contexto real de aplicación, como son los formularios web, a través de los cuales las aplicaciones interactúan con el usuario, solicitando datos y procesando o comprobando la información solicitada.

De forma más genérica y enunciado en forma de competencia académica, proponemos que los estudiantes de ALF sean capaces de *desarrollar fragmentos de código basados en una notación formal para resolver problemas de reconocimiento de patrones*. La introducción de esta competencia se ha llevado a cabo durante el curso 2006-07 en la Ingeniería Técnica de Informática de Sistemas de la UPV-EHU y el resultado ha sido muy satisfactorio.

Este artículo expone cómo ha sido la propuesta y un valoración de sus resultados. En el segundo apartado se presentan más detalles sobre los antecedentes, motivaciones y objetivos de la experiencia. En el apartado tres se incluyen los pasos y diseño del complemento práctico en la asignatura ALF, revisándose en el apartado cuatro algunas de las dificultades que pueden encontrarse los estudiantes en su realización. El apartado cinco presenta los datos concretos sobre el desarrollo de la experiencia en el curso 2006-07, para terminar con las conclusiones y valoración en el apartado seis.

2. Motivación y objetivos

La asignatura Autómatas y Lenguajes Formales (ALF) aparece, con este u otros nombres, en todas las titulaciones de Ingeniería Informática e Ingeniería Informática de Sistemas. Incluye materias troncales y obligatorias en los planes de estudios. Son contenidos básicos, que permanecen en todas las recomendaciones curriculares sujetas a revisiones y cambios periódicos [1]. Los contenidos concretos en las distintas universidades apenas varían. Las diferencias existentes pueden darse, como para cualquier otra materia, en el grado de profundidad de determinados temas, el orden de impartición o la diferenciación entre básico y avanzado. Los contenidos comunes siempre incluyen los tipos de lenguajes formales más importantes, particularmente los lenguajes regulares y los lenguajes independientes de contexto. En este entorno los autómatas finitos y las expresiones regulares son modelos abstractos indispensables para la definición de los lenguajes regulares.

Tradicionalmente estos contenidos han estado ligados a las ramas más teóricas de la Informática y su enfoque práctico se ha solido centrar en los principios, técnicas y herramientas relacionados con la compilación. Como tanto las expresiones regulares como los autómatas finitos son elementos imprescindibles para el diseño y construcción de analizadores léxicos, uno de los trabajos prácticos habituales a desarrollar por los alumnos suele estar en esta línea (en algunos casos utilizando herramientas de generación automática de analizadores y en otros casos sin recurrir a ellas). El uso y potencia expresiva de las expresiones regulares queda así reducido a los casos de unidades léxicas presentes habitualmente en los lenguajes de programación: identificadores, palabras reservadas, números enteros y reales, operadores, comentarios y separadores.

Otro tipo de actividades prácticas relacionadas con esta materia están dirigidas al uso de herramientas gráficas de visualización, como JFLAP [6], JCT [5] u otras. Estas utilidades permiten practicar con el diseño o aplicar algoritmos de transformación, principalmente con autómatas. Si bien es clara la intención motivadora del uso de estas herramientas los problemas y ejercicios planteables son similares a lo que se pueden realizar con lápiz y papel. Además, el paradójico fin del uso de la herramienta "práctica" es la comprensión del concepto teórico, y fuera de este "uso" las herramientas mencionadas no tienen ninguna otra utilidad.

A lo largo de los años de docencia en esta asignatura hemos ido planteando estas aproximaciones u otras menos habituales, como la realización de una práctica sobre autómatas

celulares [3]. Si bien todas estas experiencias tratan de buscar el enfoque práctico hemos ido constatando sus limitaciones.

Por otro lado, con la continua evolución de los lenguajes de programación han surgido los lenguajes informáticos especializados en tareas de descripción y estructuración de información, los lenguajes de marcas, y los lenguajes de programación más utilizados en el entorno del desarrollo web (Perl, Java, JavaScript, PHP) han ido incorporando las expresiones regulares como objetos primitivos o librerías. Así pues, el uso real de expresiones regulares lejos de restringirse está cada vez más extendido. Si los estudiantes manipulan realmente algún lenguaje que permita el uso de expresiones regulares intuirán su utilidad práctica. No se trata en ningún caso de introducir un nuevo lenguaje de programación, sino de buscar la manera más sencilla y rápida de "programar" con expresiones regulares.

Analizando la arquitectura de las aplicaciones web, encontramos que existe una capa de interacción con el usuario por medio de la cual se acepta la información introducida por éste. Para evitar el tráfico innecesario, parte de las tareas de validación de la corrección de los datos de entrada se realiza directamente en el lado cliente, asociando las operaciones de validación a la propia presentación de los formularios de entrada. Así, por ejemplo, cuando se rellena un formulario donde es necesario insertar el DNI, el número de cuenta bancaria, una dirección de correo o una fecha, se comprueba si se ha introducido una cadena de caracteres de acuerdo al formato correspondiente. Esta tarea de comprobación se ve facilitada por el uso de expresiones regulares. El diseño de formularios (o documentos html sencillos) que incluyan como tarea de programación el comprobar si un texto o secuencia de caracteres se ajusta a un patrón determinado es una práctica de programación abordable, sin ninguna duda, por estudiantes de Ingeniería Técnica Informática de segundo curso. En ese momento están en condiciones de entender y valorar la mejora que supone utilizar un constructor que permite resolver directamente el problema frente a la programación *adhoc* de reconocedores léxicos. Se trata, por tanto, de utilizar un lenguaje de programación que incluya las expresiones regulares como objeto. Los estudiantes serán así conscientes de que la dificultad radica en desarrollar correctamente la especificación del lenguaje o patrón que se desea reconocer y no en los problemas de programación.

El planteamiento de un complemento formativo de estas características persigue favorecer un aprendizaje significativo al buscar la aplicabilidad de las expresiones regulares y vincularlas con un contexto real. Con esta estrategia la intención es también mejorar la motivación para el aprendizaje y el estudio de la asignatura, y también, aunque en menor medida, promover el aprendizaje autónomo.

3. Planificación, estructuración y diseño

El primer paso para el planteamiento completo de la actividad práctica ha sido la selección del entorno y lenguaje más apropiado para alcanzar los objetivos previstos. Cuando decidimos explorar la idea presentada en el apartado anterior, consideramos la necesidad de que la tarea a realizar se pudiera integrar con facilidad en la experiencia de programación de los estudiantes y fuera sostenible en el tiempo, independientemente de que tuvieran una cultura de programación asociada a lenguajes diferentes, como Java, C, Ada o Visual Basic (en los últimos tiempos no son pocos los estudiantes que provienen de ciclos formativos de FP). Estas razones nos llevaron a optar por JavaScript. Al margen de los lenguajes de programación con los que los estudiantes trabajan en otras asignaturas es factible, con muy poco coste, trabajar con las expresiones regulares de JavaScript. La independencia de plataforma que aporta esta opción permite, además, intercambiar soluciones entre estudiantes y profesores sin necesidad de ninguna inversión de tiempo ni preparación de infraestructura hardware o software de soporte. Los programas en JavaScript generalmente se ejecutan en el propio navegador y van incrustados dentro de una página HTML. Por esta razón el entorno de trabajo necesario está disponible para estudiantes y profesores prácticamente en cualquier ordenador, sin necesidad de recurrir a software o instalaciones que requieran de formación y apoyo externo adicional.

La tarea concreta que el alumno debe abordar consiste en la creación de un formulario (página html) que tenga asociadas expresiones regulares de JavaScript para reconocer patrones de datos correctamente introducidos. Pueden proponerse numerosos patrones interesantes y de diferentes

grados de dificultad (código postal, dirección de correo electrónico, dirección URL, etc.)

El diseño de cualquier actividad de aprendizaje ha de completarse con la planificación, la dinámica y la estructuración más adecuadas para el tipo de aprendizaje que deseamos proponer. En el anexo se incluye el enunciado completo con que se ha presentado la actividad a los estudiantes, la cual ha sido concebida para que los alumnos puedan abordarla casi al inicio del curso. Si se pretende aumentar la motivación, conviene comenzar a trabajar pronto en aplicaciones prácticas de la misma, sin restar tiempo a la presentación de la teoría.

La actividad se plantea como un trabajo a realizar en pequeños grupos (de dos personas), debido tanto al momento de la realización como a los objetivos y habilidades que requiere. Se plantea como un trabajo guiado y por ello se divide en siete subtareas. Unas, previas a la actividad propiamente dicha, consisten en la búsqueda y lectura de información sobre las expresiones regulares de JavaScript y sus particularidades sintácticas (tarea 1), así como la realización de unos primeros ejemplos simples (tareas 2 y 3). Con estas actividades previas los estudiantes podrán realizar con menos dificultades la tarea central (tarea 4).

Como trabajo posterior, y con el fin de que los estudiantes realicen una labor de reflexión y elaboración, se les solicita que establezcan una relación entre el patrón utilizado y la correspondiente expresión regular "teórica" (tarea 6), y también que analicen la dificultad de realizar esta misma tarea con un lenguaje de programación que no disponga de expresiones regulares (tarea 7).

4. Dificultades de la actividad

Nuestro objetivo es que los estudiantes realicen la actividad de forma autónoma sin recibir ninguna formación sobre JavaScript ni sobre las especificidades de sus expresiones regulares. Con la división en subtareas mencionada en el apartado tres pretendemos separar la verdadera dificultad que pueden encontrar los estudiantes para realizar correctamente este complemento formativo.

No presenta ninguna dificultad la realización de un primer programa sin expresiones regulares, ya que basta con investigar cómo incluir código JavaScript en un documento html.

Se ha considerado útil comenzar con un programa que utilice una expresión regular sencilla para introducir las "expresiones regulares" como "objeto" con su declaración y sus métodos. Es necesario comprender en primer lugar cómo se crean las expresiones regulares en JavaScript, pero sobre todo el funcionamiento del método "test" de las expresiones regulares, que permite comprobar si una palabra es generada por la expresión regular (o se ajusta al patrón proporcionado). Aunque se pretenda desarrollar una expresión regular tan simple como la palabra ALF, hacer el primer programa supone empezar a trabajar con la sintaxis de las expresiones regulares. Esto es debido a que es necesario descartar las apariciones de ALF como subpalabra y para ello hay que utilizar símbolos especiales, concretamente dos que indican el principio y final. El patrón a usar es ^ALF$ y no simplemente ALF.

Este ejemplo nos sitúa frente al mayor obstáculo de la actividad: la comprensión y uso de la sintaxis particular de las expresiones regulares de JavaScript. Para la correcta construcción de un patrón determinado resultan necesarias numerosas consultas a su definición, lo mismo que para documentar algunas diferencias entre el metalenguaje que se utiliza para describir un patrón y su correspondiente expresión formal como expresión regular teórica.

La dificultad del lenguaje de especificación o definición de patrones que utiliza JavaScript no es exclusivo de este lenguaje, pues lo mismo sucede con cualquier otro entorno concreto (Perl, Lex, ...) en que es necesario combinar los mismos símbolos como elementos del lenguaje y metalenguaje.

La definición formal de expresión regular es una definición inductiva de tres elementos básicos (vacío, palabra vacía, símbolo del alfabeto) y tres operaciones constructoras (unión, concatenación y clausura). Sin embargo, en la definición de patrones que permite JavaScript [4] existe, además de algunos cambios de notación respecto a esta definición formal, un incremento en los operadores permitidos, ya que para construir el patrón se utilizan una serie de caracteres especiales o metacaracteres, como pueden ser el interrogante, el punto o el asterisco, que a modo de operadores nos permiten construir el patrón. Estos metacaracteres pueden ser de posicionamiento, cuantificadores, de modificación, pueden indicar un rango, etc. Por

ejemplo, ^\d{1,2}\ / \d{1,2}\ / \d{2,4}$, es un patrón para definir fechas con el formato día/mes/año. El subpatrón \d{1,2} define el formato del día y el mes. En este caso el literal d modifica su significado al ir precedido del metacaracter \ y equivale a cualquier dígito; al ir seguido de {1,2} indica que puede aparecer una o dos veces. De modo similar para el año se indica que han de aparecer bien dos, bien cuatro dígitos. Otra forma equivalente de hacerlo sería utilizando la notación de rango, [0-9]{2,4}. Para asegurar que el día, mes y año aparezcan separados con /, el patrón incluye \ / indicando que el carácter que sigue al metacaracter \, ha de interpretarse literalmente. Además se usan también metacarateres delimitadores (^ y $) para indicar el principio y final.

La utilización correcta de los metacaracteres en la definición de los patrones es la única dificultad real a la hora de construir expresiones regulares en JavaScript. Al hecho de ser una sintaxis suficientemente compleja se une la inexistencia de mensajes de error en una fase de preproceso (por ejemplo sobre el mal uso de metacaracteres), por lo que no es fácil hacer un diagnóstico de dónde o por qué el programa no funciona como se esperaba. En algunos casos, sin pruebas suficientemente significativas podría suponerse válida una expresión regular errónea.

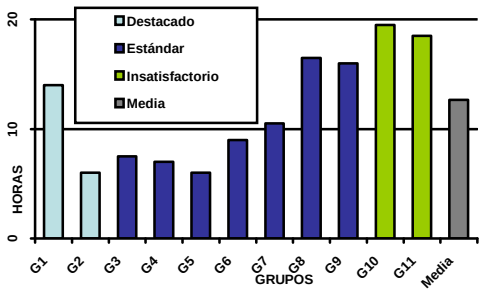
5. Desarrollo de la experiencia

Durante el curso 06/07 se ha implantado por primera vez la actividad concebida, siendo planteada a estudiantes que han seguido ALF en la Facultad de Informática de San Sebastián en un grupo en el que, a la par, se ha experimentado la adaptación de la asignatura al crédito europeo [7]. Ha sido realizada por veinte estudiantes repartidos en grupos de dos. Estos grupos no se formaron únicamente para esta actividad, pues desarrollaron otras dos dentro de esta misma asignatura. Todos los estudiantes pudieron realizarla de manera totalmente autónoma, obteniendo resultados satisfactorios ocho de los diez grupos.

Los trabajos se calificaron con tres notas: destacado, estándar o insatisfactorio. En la evaluación se tuvo en cuenta la validez de la solución desarrollada (concretamente la definición correcta de la expresión regular utilizada), pero también la calidad de la documentación. Las calificaciones superiores se asignaron a aquellos trabajos que mejor detallaron la comparación de los metalenguajes de definición de expresiones regulares.

Con el fin de disponer de datos con que valorar el tiempo que necesitaron los estudiantes para realizar una actividad como esta se les solicitó información sobre la dedicación a cada subtarea. También se consideró interesante tener en cuenta su opinión sobre lo aprendido y se les solicitó una valoración relacionada con los dos objetivos planteados (entender mejor las expresiones regulares y comprender su utilidad práctica), y una autoevaluación de la competencia que habían trabajado. Esta información la entregaron simultáneamente con la actividad.

Figura 1. Coste actividad



Respecto al coste en tiempo, como puede verse en la figura 1, la media fue de doce horas por grupo, aunque en general tendieron a trabajar conjuntamente toda la actividad y sumar la dedicación de los dos estudiantes del grupo. Quienes más tiempo invirtieron fueron los que no llegaron a realizar correctamente la actividad, por no usar adecuadamente las expresiones regulares de JavaScript y dedicar demasiado tiempo a búsqueda de información.

Respecto a los objetivos previstos consideraron con carácter general que con la realización de la actividad entendieron mejor las expresiones regulares y comprendieron su utilidad real. También hubo un grupo que, a pesar de realizar la actividad, reconoció que no era capaz de resolver problemas de reconocimiento de patrones.

6. Conclusión

Además de los objetivos de motivación y comprensión de la utilidad práctica comentados

anteriormente, la propuesta de una actividad como esta apunta hacia otros objetivos no menos importantes.

Con esta actividad los estudiantes pueden reforzar los nuevos conocimientos adquiridos con este complemento práctico. Como hemos justificado, les permite aplicar aprendizajes, ya que experimentan con el uso de las expresiones regulares en un entorno real. El desarrollo de esta actividad exige también la búsqueda y transmisión de información. Han de manejar una sintaxis concreta para las expresiones regulares que los estudiantes han de preocuparse de buscar, practicar y entender. En algunos casos la especificación puede alejarse bastante de la que se correspondería con la definición formal, por lo que es importante la reflexión y comparación entre dos o más especificaciones posibles y también la argumentación de la validez de la expresión regular elegida como patrón. Por último, como cualquier problema que requiera una solución algorítmica, la realización de la actividadconlleva cierta dosis de creatividad.

Existen dos subtareas aparentemente extras para un estudiante de ALF: el uso de un lenguaje de programación probablemente nuevo y el aprendizaje de una sintaxis de expresiones regulares más compleja que la de la definición teórica. Sin embargo, hemos recalcado que consideramos importante conocer el uso real que hay tras una definición teórica y además, obtener una solución conlleva una mayor comprensión del uso de operadores de las expresiones regulares y su orden de precedencia.

La obligación de adquirir habilidades que les permitan incluir dentro de un documento html código JavaScript no es más que avanzar algunos conocimientos mínimos que terminarán completando realmente en otras asignaturas. El uso de este lenguaje tiene sobre todo ventajas desde el punto de vista de instrumentación informática, ya que todos los usuarios informáticos disponen de un navegador que permite trabajar con JavaScript.

Una vez realizada esta actividad, completada con el conocimiento y estudio de los autómatas finitos y su equivalencia, el siguiente paso es plantear la reflexión sobre el alcance de especificación de las expresiones regulares para ser conscientes de las limitaciones del uso de este modelo como patrón para cualquier lenguaje, lo que justificará el estudio de otros modelos de descripción. En todo caso, a partir de un conocimiento real es más fácil justificar otros usos sin llegar a experimentarlos, como puede ser el uso de expresiones regulares en otros contextos.

Una última característica reseñable de esta experiencia es la facilidad de exportación e implantación dada su independencia del entorno software y hardware, disponible para profesores y estudiantes. Esto permite que se pueda aprovechar en contextos académicos similares.

Agradecimientos

Nuestro agradecimiento a los estudiantes de ALF de la Facultad de Informática de San Sebastián.

Referencias

[1] ACM/AIS/IEE-CS Task Force. Computing Curriculum 2005.

[2] Hopcroft J. E., Motwani R., Ullman J. D. *Introducción a la Teoría de Autómatas, lenguajes y Computación.* Pearson Educación. S.A., Madrid 2002

[3] Ibáñez, J., Irastorza, A., Sánchez, A., Ferrero, B. Implantación de una práctica de programación en una asignatura teórica de Ingeniería Informática: autómatas celulares autorreprodcutivos. UNIMAC. Madrid 1994

[4] Nieto Pérez I. JavaScript avanzado. [on line] http://www.elcodigo.net/tutoriales/jsavanzado/jsavanzado5.html

[5] Robinson, M.B., Hamshar, J.A., Novillo, J.E., Duchowski, A.T. *A Java-based Tool for Reasoning About Models of Computation Through Simulating Finite Automata and Turing Machines.* 30th Annual ACM SIGCSE Symposium, New Orleans, 1999

[6] Rodger S. H., Finley T. W. *JFLAP: An Interactive Formal Languages and Automata Package.* Jones and Bartlet Publishers. 2006

[7] Sánchez, A. Hacia la adaptación al crédito europeo en Autómatas y Lenguajes Formales. Informe interno. UPV/EHU/LSI/TR-01-2007

Autómatas y Lenguajes Formales-Curso 2006-07 - Grupo 16
Evaluación continua modelo ECTS-**AE3 -** Actividad en **grupo**
Plazo máximo de entrega 13 de Noviembre

USO DE EXPRESIONES REGULARES EN JAVASCRIPT

Las **expresiones regulares** son descriptores de lenguajes regulares. Una de sus utilidades más conocidas es en la realización de búsquedas de las cadenas o palabras que describen o que se ajustan a dicho "patrón", como lo hacen, en mayor o menor medida, los editores de textos. Otra utilidad es la comprobación de que una cadena de texto cumple ciertos requisitos, por ejemplo a la hora de rellenar formularios. Los patrones o descripciones que utilizan estas aplicaciones están relacionados con las expresiones regulares.

Ya hemos visto la definición y ejemplos de expresiones regulares. Ahora se trata de utilizarlas con una finalidad y en un contexto concreto.

El objetivo específico de esta actividad es conocer el manejo de expresiones regulares que permite un lenguaje concreto: **JavaScript**.

Para la realización de esta actividad los pasos a seguir son:

T1. Búsqueda de información sobre JavaScript y las expresiones regulares en JavaScript. Algunos ejemplos de direcciones útiles para esta tarea:

- Página web que describe el lenguaje JavaScript
  http://www.unav.es/cti/manuales/TutorialJavaScript/indices/
- Página web que describe las expresiones regulares y sus aplicaciones
  http://es.wikipedia.org/wiki/Expresi%C3%B3n_regular
- Página web que describe la sintaxis de las expresiones regulares y sus métodos
  http://developer.mozilla.org/es/docs/Gu%C3%ADa_JavaScript_1.5:Expresiones_Regulares
- Capítulo 5 del libro de varios autores "Programación de Aplicaciones Web" que incluye creación de páginas Web interactivas: JavaScript

T2. Desarrollo de al menos un programa con JavaScript

T3. Desarrollo de al menos un programa con JavaScript que incluya el uso de expresiones regulares.

T4. Creación de un documento HTML directamente procesable en un navegador, que solicite un correo electrónico y detecte si responde al patrón de correo utilizado por la UPV-EHU para asignar direcciones de e-mail a sus estudiantes.

T5. Documentar el programa de la tarea T4 y las expresiones regulares utilizadas

T6. Reflexionar sobre las diferencias entre nuestra definición de expresión regular y las expresiones regulares utilizadas en JavaScript

T7. Estimar justificadamente el coste (en tiempo) que supondría realizar la tarea T4 con un lenguaje de programación que no disponga de expresiones regulares.

ENTREGA

Se entregará el documento HTML de la tarea T4 y otro documento que contendrá un apartado por cada una de las tareas especificadas y el coste y valoración de la actividad en el modo descrito en la siguiente página. Se enviará por correo electrónico a través del correo institucional. El mensaje llevará por título AE3 y los apellidos de las dos personas del grupo.

EVALUACIÓN

Se tendrán en cuenta tanto las tareas realizadas como la documentación presentada.

P1: Apellidos, Nombre

P2: Apellidos, Nombre

COSTE DE LA ACTIVIDAD AE3

Horas totales para la realización de esta actividad:

	más de tres		**entre dos y tres**		**entre una y dos**		**una o menos**		**Total**
	P1	**P2**	**P1**	**P2**	**P1**	**P2**	**P1**	**P2**	
T1:	☐	☐	☐	☐	☐	☐	☐	☐	____
T2:	☐	☐	☐	☐	☐	☐	☐	☐	____
T3:	☐	☐	☐	☐	☐	☐	☐	☐	____
T4:	☐	☐	☐	☐	☐	☐	☐	☐	____
T5:	☐	☐	☐	☐	☐	☐	☐	☐	____
T6:	☐	☐	☐	☐	☐	☐	☐	☐	____
T7:	☐	☐	☐	☐	☐	☐	☐	☐	____

	más de tres		**entre dos y tres**		**entre una y dos**		**una o menos**	
	P1	**P2**	**P1**	**P2**	**P1**	**P2**	**P1**	**P2**
Elaborar la documentación: **Total** ___	☐	☐	☐	☐	☐	☐	☐	☐
Otros (especificar): **Total** ___	☐	☐	☐	☐	☐	☐	☐	☐

Total P1: __Total P2:___ Total Grupo**: ___**

VALORACIÓN DE LA ACTIVIDAD

La realización de la actividad me ha servido para	**P1**		**P2**	
Entender mejor la definición de expresiones regulares	**No**	**Si**	**No**	**Si**
Comprender la utilidad real de las expresiones regulares	**No**	**Si**	**No**	**Si**

AUTOEVALUACIÓN Valora tu capacidad para el desarrollo de la competencia trabajada

Soy capaz de desarrollar fragmentos de código basados en una notación formal para resolver problemas de reconocimiento de patrones

P1		**P2**	
Si	No	Si	No