NNODOCT 2015
Universitat Politècnica de València. València, 2015
DOI: http://dx.doi.org/10.4995/INNODOCT.2015.846

EL CICLO DEL APRENDIZAJE PREFERENCIAL COMPLEMENTARIO Y LAS TAREAS DE APRENDIZAJE. EL CASO DE LA RECURSIVIDAD.

Margarita Díaz-Roca [a], Francisco J. Gil-Cordeiro [b] y Daniel J. Ojeda-Loisel [c]

[a] Universidad de Las Palmas de Gran Canaria (España, mdiaz@dis.ulpgc.es), [b] Instituto de Enseñanza Secundaria Alonso Quesada (España, fgilcor@gobiernodecanarias.org), [c] Alumno Grado de Ingeniería Informática Univesidad de Las Palmas de Gran Canaria (España, daniel.ojeda107@alu.ulpgc.es)

Resumen

El modelo de estilos denominado Aprendizaje Preferencial Complementario (APC) se viene desarrollando desde 2008 en la práctica del aula con los alumnos universitarios de Ingeniería Informática. La metodología utilizada se basa en el trabajo colaborativo con equipos compuestos por estudiantes de diferentes estilos de aprendizaje, identificados con el APC, evaluados de forma continua y con coevaluación de la participación grupal. En consecuencia, se plantea la pregunta de cómo deben de estar organizados los contenidos y actividades formativas, de tal manera que haya consonancia con la presencia de estos estilos en el aula. El APC establece la existencia de un ciclo de aprendizaje colectivo y otro individual para los alumnos. Es necesario, en primer lugar, partir del ciclo colectivo, ajustando la elaboración de los materiales para que estos den oportunidad a todos los estilos de tener protagonismo, durante el trabajo colaborativo, en el aprendizaje grupal. En este artículo se estructura el aprendizaje de la recursividad según el Ciclo del APC, realizado con la experiencia de alumnos voluntarios que colaboraron desde sus propios estilos.

Palabras clave: *aprendizaje preferencial complementario; ciclo del APC; estilos de aprendizaje; recursividad.*

Introducción

Durante estos últimos años la investigación en la enseñanza y el aprendizaje ha pasado de la evaluación de la eficacia a través de las encuestas de satisfacción e impresiones de los profesores, a los diseños experimentales con grupos de control y, más recientemente, a una investigación más rigurosa utilizando métodos y filosofías de las ciencias sociales (Felder & Hadgraft, 2013). Proyectos recientes como el RREE (Streveler, Borrego & Smith, 2007) proponen niveles de rigor en la enseñanza y el aprendizaje, que suponen ir aumentando la posibilidad de que la investigación en esta materia sea presentada en foros y reproducible por otros investigadores. Más aún, se requiere cada vez más que se dé respuesta a cómo se produce el aprendizaje, se interpreten los resultados a la luz de las teorías y se preste atención al diseño del estudio y los métodos, añadiendo validez y fiabilidad a los hallazgos (Felder & Hadgraft, 2013).

Desde el año 2008 se ha llevado a cabo una investigación acerca de la eficiencia en el proceso de enseñanza-aprendizaje de la aplicación de la Metodología basada en los estilos del APC (MAPC) (Díaz, Gil y Alonso, 2010) (Díaz et al., 2011, 2012, 2013, 2014). Esta investigación se ha desarrollado en el marco de Proyectos de Innovación Educativa (el código del último es CPIE2013-05) con la participación de estudiantes de la Escuela de Ingeniería Informática de la Universidad de Las Palmas de Gran Canaria.

Llevar a la práctica educativa los estilos de aprendizaje implica el poder reconocerlos en los alumnos pero también en los profesores, ya que están directamente relacionados con sus estilos de enseñanza. Además, hay que conocer a fondo las relaciones entre estilos para organizar en equipos a los estudiantes. Hay que dinamizar los equipos para que cada miembro aporte colaborando desde su especificidad y así enriquezca el aprendizaje de los compañeros. Asimismo, hay que seleccionar y secuenciar los materiales de aprendizaje de forma que contengan específicamente secciones propias de los estilos para dar oportunidad de protagonismo a todos los estudiantes ante los demás compañeros.

En este trabajo se expone la organización del tema de la recursividad que ha sido objeto de estudio de diversos investigadores desde distintos ángulos (Lacave, Molina y del Castillo, 2014). En este caso, el objetivo es mejorar la dificultad de aprendizaje de este concepto desde la presencia detallada de la aportación de cada estilo de aprendizaje. Se propone una unidad didáctica inicial en torno a este eje vertebrador siguiendo el hilo del Ciclo del APC (CAPC) (Díaz et al., 2013), secuenciando los momentos específicos de actuación de los diferentes estilos de aprendizaje, dejando abierta al final la posibilidad de ampliar la espiral de aprendizaje en sucesivas vueltas. En concreto, se estructura el aprendizaje de la recursividad lineal según el CAPC, y se establecen actividades, empezando por la motivación para aprender a programar desde la recursividad, asociándolo al conocimiento previo de algoritmos iterativos, estructurando este nuevo concepto con sus casos base y general, detallando su funcionamiento, haciendo el seguimiento de la ejecución del código, estableciendo los pasos a seguir para hacer un programa recursivo y, finalmente, indicando criterios que faciliten la creación de tales programas.

El Aprendizaje Preferencial Complementario

El APC se puede explicar en función de los posibles roles que suele desempeñar una persona en un ámbito profesional. Para distinguir los estilos de aprendizaje se puede utilizar la demanda de necesidades que el individuo expresa en el sentido de su discurso, tal como se entiende éste según la "teoría del sentido" (Holzapfel, 2005). Al igual que Gardner, en la teoría de las inteligencias múltiples, el APC considera dos principios básicos del aprendizaje: una preferencia de la persona por un estilo de aprendizaje concreto, que se mantiene toda la vida, y una complementariedad entre los diferentes estilos de aprendizaje de las personas. Se puede decir que una persona cuando aprende actúa desarrollando alguno de los posibles roles (Díaz et al., 2011), cuya sinopsis es (Díaz et al., 2012):

- Orientador: Aprende planificando el camino a seguir para llegar a una nueva realidad.
- Racionalista: Aprende asociando lo nuevo con una experiencia recordada buscando la evolución.
- Creativo: Aprende buscando parecidos, es decir, establece paralelismos y confirma que hay una estructura común.
- Perfeccionista: Aprende buscando diferencias, cambios, y obtiene una temporalización (calendario) del trabajo para repartir el esfuerzo y llegar a un objetivo.
- Constructivista: Aprende buscando la clasificación basada en las definiciones de las formas.
- Metódico: Aprende buscando rarezas, intentando completar la colección, enumerada según una escala en la que cada elemento aporta algo que no existía anteriormente.
- Estratega: Aprende buscando el juego y estudiando la conducta de los jugadores para entender su estrategia.

El Ciclo del APC

Los alumnos pueden trabajar individualmente, en parejas, tríos,.., la situación ideal es que trabajen en equipos formados con los siete estilos de aprendizaje. Para ello hay que tener en cuenta que los distintos estilos están relacionados por complementariedad, así, el Racionalista se complementa con el Perfeccionista, el Creativo con el Metódico y el Constructivista con el Estratega. El Orientador se complementa con todos. El Creativo, el Perfeccionista y el Estratega desempeñan el papel activo en este emparejamiento, mientras que el Racionalista, el Constructivista y el Métódico hacen el papel pasivo.

Una persona cuando aprende sigue un ciclo desde el comienzo hasta conseguir el aprendizaje, de la misma manera que lo puede hacer un equipo, pero, evidentemente con una resultante de distinto grado que la de aquel. Hay una similitud a distinta escala entre una persona haciendo consecutivamente todos los roles del APC, complementándolo, y un equipo de expertos haciendo cada uno un rol cooperando de forma complementaria. Cuando lo hace una sola persona se circunscribe a su ámbito preferencial, completa su realidad, y cuando lo hace el equipo completa toda la realidad. El aprendizaje en equipo ocurre secuencialmente siguiendo el CAPC (Díaz et al., 2013): primero se aprende desde la motivación, después a partir de la experimentación, la conceptualización, el procesamiento, la mecanización, la consolidación y, por último, la evaluación.

El Ciclo del APC en el aprendizaje de la recursividad lineal

Motivación

El Ciclo comienza con la Motivación papel que corresponde en el aprendizaje en equipo al perfil del Orientador. La pregunta que hay que responder es: *¿para qué sirve la programación recursiva?* Desde la perspectiva de la realidad hay que buscar una *orientación* para seguir un camino en busca de otras posibilidades de programar. Se inicia el tema señalando que la programación recursiva es otra forma de programar que puede ayudar a resolver problemas que son complejos de implementar de forma iterativa.

Un programa iterativo resuelve un problema ejecutando las sentencias de principio a fin obteniendo resultados a medida que se avanza. A veces el problema es difícil de resolver por su tamaño, aunque sería fácil si tuviese un tamaño pequeño. Un programa recursivo parte del hecho de que *el problema se puede resolver si es posible resolver otro similar al original pero menos complejo*, y que este a su vez se puede resolver si es posible resolver otro similar pero menos complejo, y así sucesivamente, acercándose cada vez más a uno suficientemente simple que ya se puede resolver directamente. Volviendo hacia atrás, deshaciendo el camino, se obtiene la solución final.

Ahora hay que concretar esta orientación a partir de lo que ya se conoce.

Experimentación

La siguiente fase en el Ciclo es la Experimentación que corresponde al perfil del Racionalista. La pregunta que hay que responder es: *¿qué experiencia tenemos con la programación recursiva?* Desde esta perspectiva de la significación hay que desarrollar el *razonamiento* que lleva, poco a poco, a hacer programas recursivos.

En realidad las personas han vivido situaciones desde pequeños en las que han aprendido a resolver problemas por medio de la recursividad. Por ejemplo, si hay que contar 1000 euros en monedas de un

euro, se puede ir contando de uno en uno hasta completar las 1000 monedas de un euro. Esta sería una solución iterativa que puede hacer una máquina contadora de monedas, pero si hay que hacerlo a mano cabe la posibilidad de equivocarse en el recuento secuencial. La solución recursiva sería contar las monedas formando montones de 10 monedas, luego se agruparían 10 montones de 10 euros y se tendrían 10·10=100 euros. Formando 10 montones de 100 euros se tendrían 10·100=1000 euros. Se obtendría al final 10^3. En matemáticas una definición recursiva de la potencia (en base 10) es: $10^n=10\cdot10^{n-1}$, si $n\neq0$, siendo $10^0=1$.

El razonamiento es este, *se va reduciendo el problema hasta que alcanza un tamaño pequeño de datos para el que se puede resolver fácilmente* (10^0), *y a partir de ahí se va resolviendo para tamaños mayores* (10^1, 10^2), *hasta llegar al problema inicial* (calcular 10^3), *resolviendolo de la misma manera* (multiplicando por 10 la potencia anterior), *acumulando así las soluciones obtenidas hasta que se obtiene la solución final*.

A partir de aquí hay que hacer la síntesis de cualquier programa recursivo.

Conceptualización

Es el momento de la Conceptualización que corresponde al perfil del Creativo. La pregunta que hay que responder es: *¿cómo se estructura un programa recursivo?* Desde la perspectiva del sentido hay que establecer los *conceptos* que permitan generar de forma estructurada cualquier programa recursivo.

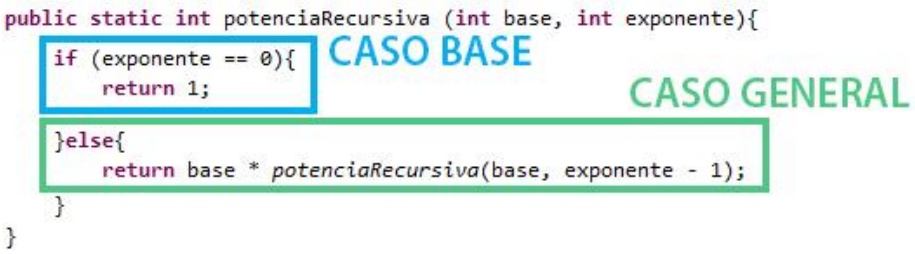
El sentido del programa está en el principio de inducción: *el programa resuelve el problema si lo hace para un caso sencillo (caso base) y si se supone resuelto para el caso n-1, se puede afirmar que lo resuelve también para el caso n (caso general).*

Por ejemplo, en el cálculo de la potencia de un número entero a^n ($a\neq0$ y $n\geq0$), el caso base es $a^0=1$ y el caso genral es $a^n=a\cdot a^{n-1}$. En particular, el siguiente programa en Java calcula 5^4:

```
public static void main (String args[]){
    int base = 5;
    int exponente = 4;
    int resultado = potenciaRecursiva(base, exponente);
    System.out.println(resultado);
}

public static int potenciaRecursiva (int base, int exponente){
    if (exponente == 0){
        return 1;
    }else{
        return base * potenciaRecursiva(base, exponente - 1);
    }
}
```

¿Cómo se sabe que realmente calcula esa potencia? Calcula el caso más sencillo 5^0, devolviendo 1, es el caso base que está representado por la primera alternativa de la estructura de selección. También calcula 5^4 asumiendo que el programa sabe calcular 5^3, multiplicando $5\cdot5^3$, lo cual es correcto, es el caso general representado por la segunda alternativa. La estructura general de un programa recursivo lineal es, por tanto, una estructura de selección como en el ejemplo:



El CASO BASE: Se expresa mediante una condición que se tiene que cumplir en algún punto. Marcará el "final" de la recursividad. Cuando se invoca a una función recursiva, hay que saber cuándo debe dejar de seguir realizando llamadas sucesivas. ¿Llamadas sucesivas? Sí, la función recursiva se llama a sí misma una y otra vez hasta que se cumple el caso base y devuelve un resultado concreto.

El CASO GENERAL: Cuando no se cumpla el caso base, entra en acción el caso general. Es en el caso general donde se realizan la mayoría de los cálculos de la recursividad, y donde se hacen las llamadas sucesivas. Se observa que en el caso general se llama a *potenciaRecursiva* que a su vez devuelve un resultado. Y que se invoca a la función sustrayéndole 1 al exponente.

A partir de aquí hay que comprobar que se logra obtener el resultado con esta estructura.

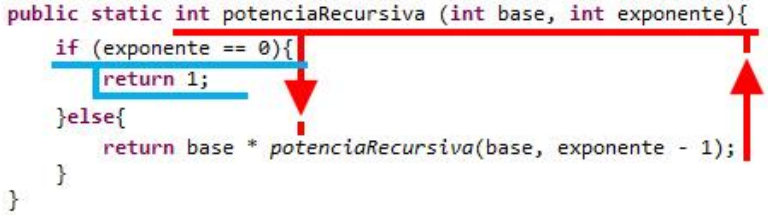
Procesamiento

Ahora es el momento del Procesamiento que corresponde al perfil del Perfeccionista. La pregunta que hay que responder es: *¿cómo trabaja un programa recursivo?* Se tiene que ver cómo funciona al detalle un programa recursivo *supervisando* todas las tareas.

La funcionalidad del programa del ejemplo anterior es la siguiente: *sigue un recorrido descendente en el que la función se llama a sí misma, disminuyendo la dificultad del problema*, decrementando el exponente, *hasta que llega al caso base y finaliza el descenso*, con exponente cero, *ahí devuelve el valor que tiene estipulado y deshace el recorrido ascendiendo, cada llamada devuelve el último resultado al cálculo del caso general hasta que se para*, logrando así calcular la potencia original.

¿Cómo está funcionando la recursividad para el cálculo de 5^4? Se llama a la función *potenciaRecursiva* una y otra vez, sustrayéndole 1 al exponente. ¿Cuándo parará? Cuando el exponente sea igual a 0, tal y como se especifica en el caso base.

¿Qué pasa entonces con la variable *base* que multiplica a la función *potenciaRecursiva* en la setencia return? *base* * *potenciaRecursiva*(base, exponente - 1) no puede ser calculado hasta que la función *potenciaRecursiva*(base, exponente - 1) devuelva un resultado para multiplicarlo por *base*. Así, cuando se llega al caso base (exponente == 0), se devuelve por fin un número entero, 1, por primera vez se devuelve y finaliza la ejecución de una de las funciones que se han ido llamando una y otra vez.



Ahora, en la función cuyo exponente valía 1, al ejecutar la sentencia return, se había llamado a *potenciaRecursiva*(base, exponente - 1). Esto hizo que el exponente valiera 0 y se cumpliera el caso base, devolviendo 1. Por lo que se reanuda la función cuyo exponente vale 1, que ahora ya sabe que el resultado de *potenciaRecursiva*(base, exponente - 1) = 1. Así, esta función devolverá *base* * 1.

Volviendo a la llamada previa: La función cuyo exponente valía 2, ahora sabe que *potenciaRecursiva*(base, exponente - 1) devolvió *base* * 1. Por tanto, devolverá *base* * (*base* * 1). La función cuyo exponente valía 3 devolverá: *base* * (*base* * (*base* * 1)). Y la función cuyo exponente valía 4 (la primera invocada) devolverá: *base* * (*base* * (*base* * (*base* * 1))). O lo que es lo mismo: $5*5*5*5*1 = 5^4 = 625$.

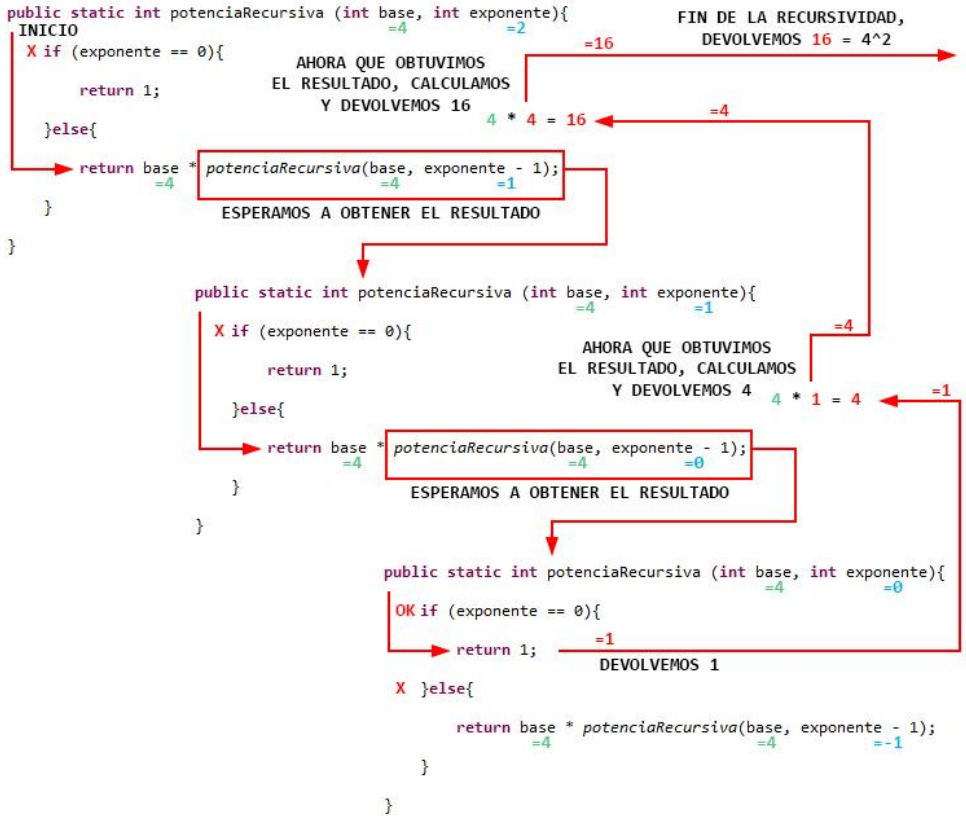
La funcionalidad del programa requiere ahora herramientas para poder comprobar de forma simple cómo se produce el resultado.

Mecanización

La fase siguiente es la de Mecanización que corresponde al Constructivista. La pregunta que hay que responder es: *¿cuál es la mecánica de un programa recursivo?* Así se lleva a la práctica el aprendizaje, para ello se cuenta con ejemplos que permiten seguir la lógica de la ejecución y comprender en la práctica de cada situación esta *forma* de programar.

A partir de los datos de entrada, base y exponente, *cada vez que se llama a la función recursiva se le pasan por parámetro los nuevos valores decrementados*, el exponente, *creando un nuevo registro de activación en la pila. Al llegar al caso base se devuelve el valor obtenido*, en este caso 1, *y se retorna a la dirección indicada en el último registro de activación para obtener el valor de la función en ese nivel. Al devolver el valor de cada llamada a la función se va vaciando la pila hasta llegar a calcular el resultado final*.

En el ejemplo de la potencia se puede ver cómo se ejecuta el código al llamar a la función recursiva para calcular 4^2 y seguir la traza de la ejecución con las flechas para construir la solución:



Es fundamental tener un protocolo para elaborar un programa recursivo que guie en la obtención del programa.

Consolidación

En la Consolidación interviene el Metódico. La pregunta que hay que responder es: *¿qué pasos hay que seguir para hacer un programa recursivo?* Así se normaliza el uso de esta forma de programar, se desarrolla un *método*, haciendo reproducible lo aprendido en cualquier instante.

El método para programar un problema de forma recursiva consta de los siguientes pasos: *buscar el dato que hace que el problema sea difícil de resolver y al que es posible disminuir su valor o reducir su tamaño;* d*eterminar el tamaño y la forma de obtener la solución del caso base; deducir la expresión adecuada para lograr tamaños del problema descendentes hacia el caso base; establecer el tamaño y la solución del caso general, las operaciones a realizar con el valor devuelto por la función para el tamaño del problema siguiente al actual. Normalmente, con una estructura condicional simple se completa el programa.*

Ahora que se tiene una idea más clara de la recursividad, ¿cómo se debería empezar la resolución del cálculo de la potencia de forma recursiva? En primer lugar, el exponente llama la atención, y es posible darse cuenta de que si se disminuye de uno en uno, se tiene una forma de ir acortando el problema. Esto no puede repetirse hasta el infinito, hay que encontrar el caso base que es cuando el exponente vale 0. Por último, hay que establecer los cálculos necesarios para que todo funcione y situar la llamada recursiva y el caso base en el lugar adecuado en el código. El caso base se sitúa al principio como condición del *if*. Hay que darse cuenta de que la solución del caso general se obtiene multiplicando la base por el

resultado de la llamada recursiva para el exponente decrementado en uno, sólo queda reflejar en el código este cálculo con un *else* que corresponde al caso general.

A partir de aquí hay que tener criterios para sopesar cómo hacer para resolver un problema de manera recursiva.

Evaluación

Por último viene el aprendizaje desde la Evaluación que corresponde al perfil del Estratega. La pregunta que hay que responder es: *¿cuál es la clave para hacerlo fácil?* Esto permite hacer bien un programa recursivo, tener una *estrategia* ganadora conduce fácilmente a la obtención de un buen programa.

La clave es considerar que el programa calcula una sucesión de valores y por lo tanto hay un término general que formular en función de los términos anteriores y un primer término, que hay que relacionar entre sí.

La dificultad de la recursividad radica en el caso general, básicamente en saber qué cálculos hay que realizar para obtener el resultado mediante las llamadas recursivas. Y cómo no, la práctica y la experiencia siempre son la mejor ayuda que se puede tener, el dominio de la recursividad se comprueba resolviendo problemas. Como por ejemplo, calcular el producto de dos números naturales mediante sumas sucesivas, o invertir una ristra de caracteres, o cualquier otro problema.

Esta unidad de programación recursiva lineal supone el ciclo inicial, además, hay que considerar recursividad múltiple (divide y vencerás) e incorporar análisis de la eficiencia, lo cual implica ampliar con nuevas vueltas el CAPC formando una espiral del aprendizaje.

Conclusiones

Los problemas de enseñanza-aprendizaje se pueden abordar desde los estilos del APC. Los estilos están presentes en los alumnos y en los profesores, pero también, en los documentos y materiales. En principio, hay que intentar coordinar estos tres elementos para mejorar el aprendizaje. Una estrategia para la identificación de estilos en los documentos y materiales puede ser pedir a los estudiantes, con estilos previamente identificados a través de test, que expliquen su comprensión de los temas tratados en los mismos. Las características de los estilos están en los razonamientos, conceptos, métodos, etc, que los alumnos transmitan. La clave está en establecer la correspondencia unívoca entre estilos de alumnos y documentos.

En este trabajo se ha hecho esta tarea con la colaboración de los alumnos en el tema de la recursividad lineal. Con su ayuda se ha estructurado este tema en forma de unidad didáctica siguiendo el ciclo del APC. El resultado ayudará a docentes e investigadores en el proceso de estructuración y adaptación a los estilos del APC de otros temas de la materia de programación.

Referencias

Díaz, M., Gil, F. J. y Alonso, J. (2010). Un nuevo modelo de estilos de aprendizaje: el Aprendizaje Preferencial Complementario. Actas de XVI Jornadas de Enseñanza Universitaria de la Informática, JENUI, 283-290.

Díaz, M. y Gil, F. J. (2011). Aplicación del Aprendizaje Preferencial Complementario ajustada a la disponibilidad de estilos de la clase. Proceedings of VII International Conference on Engineering and Computer Education, ICECE, 534-538.

Díaz, M., Gil, F. J. & Afonso, M. D. (2012). Preferential Complementary Learning. Practical Experiences. Proceedings of 4th annual International Conference on Education and New Learning Technologies, EDULEARN, 6711-6719.

Díaz, M., Gil, F. J. & Afonso, M. D. (2013). Creation of activities oriented to preferential complementary learning styles. Examples of computer science. Proceedings of 5th annual International Conference on Education and New Learning Technologies, EDULEARN, 3615-3623.

Díaz, M. y Gil, F. J. (2014). ¿La composición y la dinamización de equipos son factores importantes para el éxito del trabajo colaborativo? Experiencia con equipos organizados según los estilos del Aprendizaje Preferencial Complementario. Actas de la Conferencia Internacional en Innovación, Documentación y Tecnologías de la Enseñanza, INNODOCT, 527-536.

Felder, R. M. & Hadgraft, R. G. (2013). Educational Practice and Educational Research in Engineering: Partners, Antagonists, or ships passing in the night? Journal Engineering Education, 102(3), 339–345.

Holzapfel, C. (2005). A la búsqueda del sentido, Santiago de Chile, ed. Sudamericana.

Lacave, C., Molina, A. I. y del Castillo, E. (2014). Evaluación de una innovación docente a través de un diseño estadístico cuasi-experimental: aplicación al aprendizaje de la recursividad. Actas de XX Jornadas de Enseñanza Universitaria de la Informática, JENUI, 159-166.

Streveler, R.A., Borrego, M., & Smith, K.A. (2007). Moving from the scholarship of teaching and learning to educational research: An example from engineering. To Improve the Academy, 25, 139–149.