Universidad Internacional de La Rioja
Facultad de Educación

Trabajo fin de máster

Enseñanza en Bachillerato de Geometría a través del aprendizaje de un lenguaje de programación orientado a objetos

Presentado por: Luis González Torquemada
Línea de investigación: 1.7.2. Recursos didácticos digitales
Director/a: Dra. Mª José Díaz González

Ciudad: Madrid
Fecha: 2012

Resumen / Abstract

Resumen

El diseño de software y los lenguajes de programación son cada vez más necesarios y demandados, pero no están abordados en el currículo de la ESO ni en el Bachillerato en España. El aprendizaje de los lenguajes de programación se puede plantear como un objetivo curricular en sí mismo o bien como un medio para alcanzar otro objetivo curricular. Ambos aspectos son importantes, aunque es en el segundo en el que se centra el presente trabajo, concretamente en la geometría y la programación en C#. Este enfoque puede permitir alcanzar objetivos como: introducir los lenguajes de programación en el currículum; estructurar conocimientos matemáticos de forma similar a como se estructura un programa de software o reforzar la competencia digital en aspectos no cubiertos actualmente. Para profundizar en estos elementos, en este trabajo se analizan los conceptos y metodologías comunes entre el lenguaje matemático y los lenguajes de programación, se estudian los procesos de enseñanza-aprendizaje interdisciplinares para adaptarlo al caso de geometría y programación en C# y se generara una propuesta educativa para el desarrollo de procesos de enseñanza-aprendizaje en Bachillerato. Entre las conclusiones, se muestra la viabilidad y soporte teórico de la propuesta.

Palabras clave: Geometría, Bachillerato, recurso educativo digital, lenguaje de programación C#, APOS.

Abstract

Software design and programming languages are increasingly necessary and demanded, but are not approaches in the Spanish ESO or Bachelor curriculum. Learning programming languages can be stated as a curricular goal by itself or as a means to reach another goal-curricular. Both aspects are important, although it is in the second one in which this work is focused, particularly in geometry and C# programming. This approach can help achieve goals such as introducing programming languages in the curriculum, structure mathematical knowledge in a similar way to the structure of a software program or digital enhancing competition in areas not currently covered. To do this, the present work analyzes the concepts and methodologies between the mathematical and programming languages; it examines the processes of teaching and learning interdisciplinary response to match the case of geometry and C# programming and generates an educational proposal for development process of teaching and learning in high school. Among the findings, they show the feasibility and theoretical support of the proposal.

Key words: Geometry, High school, digital educational resource, C# programming language, APOS.

Índice

1 Introducción

El diseño de software y los lenguajes de programación son cada vez más necesarios y demandados en entornos laborales y, sin embargo, no están abordados en el currículo de la Educación Secundaria Obligatoria ni en el Bachillerato en España (ni a nivel nacional en los Reales Decretos 1631/2006 y 1467/2007, ni a nivel de la Comunidad de Madrid en los Decretos 23/2007 y 67/2008), aunque sí están presentes en otros países. Incluso Google (2010-2012) tiene una iniciativa denominada CS4HS (Computer Science for High School) que se conforma como una iniciativa para promover las Ciencias de la Computación y el Pensamiento Computacional en las escuelas de enseñanza media y superior (nuestra ESO y Bachillerato) en EE.UU., Canadá, Europa, Oriente Medio, África, China, Nueva Zelanda y Australia. Otra iniciativa similar a nivel supranacional es la denominada DreamSpark realizada por Microsoft (2012). A nivel nacional tenemos ejemplos de México con su iniciativa EMAT (Enseñanza de las Matemáticas con Tecnología) y de Uruguay con los trabajos de la profesora Sylvia da Rosa (2001, 2012). Estudiar y subsanar este retraso español es uno de los objetivos del presente trabajo.

Ya la Unión Europea (UE) en 1990, de acuerdo con Yábar (1995) y ratificado por la conferencia *Models of ICT integration in Education* celebrada en Madrid en 2010 con motivo de la presidencia española de la UE, entendía que era un objetivo primordial, introducir las nuevas tecnologías en todas las fases del proceso educativo y formativo, aunque Yábar (1995), distingue entre introducir el uso del ordenador como un fin del aprendizaje curricular e introducirlo como un medio de aprendizaje curricular. Lo mismo se puede decir respecto al aprendizaje de los lenguajes de programación: se puede plantear como un objetivo curricular en sí mismo o bien como un medio para alcanzar otro objetivo curricular (en matemáticas, por ejemplo). Ambos aspectos son importantes, aunque es en el segundo (lenguaje de programación como herramienta o ayuda) en el que se centra el presente trabajo.

Efectivamente, siguiendo a Yábar (1995), se produce una interacción entre las disciplinas de informática y matemáticas de la que las dos salen beneficiadas. Como dice este autor, esta interacción nos lleva a ver la educación informática desde una perspectiva global, como un eje transversal del currículum que influye en todas las materias y que recibe información de todas ellas, si bien en este trabajo se focaliza la atención en la geometría y la programación orientada a objetos.

Cabría preguntarse por qué la legislación educativa española (la LOE de 2006) no incluye ni como contenido curricular ni como parte de la competencia ma-

temática o digital (el objetivo del presente trabajo) la programación informática (ciencias de la computación como la denominan muchos autores), a pesar de establecer la competencia digital como una de las competencias básicas a alcanzar en la ESO y el acento puesto en las TIC. Da Rosa (2012), menciona que, si bien en los albores de la informática (años 60, 70 y 80) la relación entre Matemáticas y Ciencias de la Computación era evidente, ésta se ha ido diluyendo por dos razones:

- El avance y la popularización de la tecnología, que lleva a confundir alfabetización informática (o digital) con informática.
- El estrecho vínculo entre matemática y computación no se ha volcado al sistema educativo, sino que ha quedado restringido a los ámbitos de investigación y a las carreras específicas en computación. Los programas, metodologías y objetivos de la educación en Matemáticas no han incorporado el desarrollo de la rama de la Matemática denominada ciencia de la computación o informática.

Además, muchos conceptos inherentes a los lenguajes de programación orientados a objetos (herencia, polimorfismo, especialización o reutilización) están basados en estructuras mentales relacionadas con la manera en que se estructura el bagaje matemático de los alumnos. En efecto: de las seis fases del pensar (Carrasco et al., 2008), la fase extensiva está muy relacionada con la herencia y reutilización de clases.

De acuerdo con Da Rosa y Cirigliano (2001), en los estudios universitarios de ciencias de la computación se observa una dificultad en su aprendizaje cuyo origen estiman está sobre todo en la inadecuada base matemática del alumnado. Para solucionarlo proponen, en la Educación Secundaria y Bachillerato, la enseñanza de contenidos curriculares de matemáticas relacionándolos con conceptos de ciencias de la computación. En esta línea se encuadran las propuestas estudiadas en este trabajo.

La utilización de un lenguaje moderno orientado a objetos como C# en el aprendizaje de Geometría puede permitir alcanzar los siguientes objetivos: introducir los lenguajes de programación en el currículum; estructurar conocimientos matemáticos de forma similar a como se estructura un programa de software; reforzar la competencia digital en aspectos no cubiertos por la extinta 'Escuela 2.0'; etcétera.

C# fue creado en el año 2000 por Microsoft para dar soporte a su plataforma .NET Framework y, en palabras de Schildt (2010) es uno de los lenguajes más importantes del siglo veintiuno desde cualquier punto de vista. A pesar de nacer como un lenguaje propietario, pronto se inscribió como estándar ECMA (ECMA-334) e

ISO (ISO/IEC 23270). Además, Microsoft dispone de una plataforma de desarrollo gratuito en la que se encuadra *Microsoft Visual C# Express 2010*, que se engloba dentro de un proyecto más amplio, Microsoft DreamSpark, que "pone una parte del software más poderoso del mundo en las manos de estudiantes y educadores".

Este trabajo se desarrolla en el marco de una estrategia más amplia, que explora una posible reestructuración del Bachillerato y la incorporación a su currículo de contenidos que se adapten a las necesidades actuales que puedan tener los estudiantes y el contexto que los espera especialmente en el ámbito del desarrollo de la competencia digital y su desarrollo desde el área de Matemáticas.

2 Planteamiento del Problema

2.1 Objetivos

Objetivo general

Analizar cómo se pueden desarrollar procesos de enseñanza-aprendizaje de las Matemáticas a través del uso de los Lenguajes de programación. Dentro de las Matemáticas el trabajo se centra en la rama de la Geometría a nivel de Bachillerato, y dentro de los lenguajes de programación, se centra en los lenguajes orientados a objetos (OOP: Object Oriented Programming), más concretamente en C#.

Objetivos específicos

Para alcanzar el objetivo general, se marcan los siguientes objetivos secundarios o específicos:

- Analizar los conceptos y metodologías comunes entre el lenguaje matemático y un lenguaje de programación OOP como el C#.
- Generar una propuesta educativa para el desarrollo de procesos de enseñanza-aprendizaje en Bachillerato.
- Estudiar los procesos de enseñanza-aprendizaje interdisciplinares para adaptarlo al caso de geometría (disciplina de matemáticas) y programación en C# (disciplina de Ciencias de la Computación).

2.2 Antecedentes

Los antecedentes más próximos al objetivo planteado en este trabajo (el uso del lenguaje de programación para la enseñanza-aprendizaje de la geometría) se engloban en los siguientes apartados. Su procedencia es muy variada, puesto que son muchos los ámbitos desde los que se han estudiado y desarrollado programas de innovación educativa. Algunas iniciativas nacen de un impulso institucional de las autoridades educativas, otras provienen de equipos de investigación universitarios y. finalmente, otras tienen su origen en la iniciativa privada. A continuación se realiza un recorrido por estas múltiples perspectivas.

- **La enseñanza de la geometría a través de otras herramientas TIC**

En este apartado se enmarca la variedad de recursos existentes vinculados al uso de las TIC en la enseñanza-aprendizaje de geometría, por ejemplo a través de plataformas y entornos gráficos como son Cabri[1] o GeoGebra[2].

Ambas plataformas poseen amplio material educativo (libros de texto, actividades, ejemplos…) relacionado con la enseñanza de la geometría. Además, algunas iniciativas institucionales se apoyan en estas plataformas.

- **El uso de lenguajes de programación básicos como recurso educativo en los procesos de enseñanza – aprendizaje**

En este apartado se encuadra el uso de lenguajes de programación básicos, generalmente en otros niveles educativos distintos del bachillerato, como el Logo[3], que suele utilizarse en la Enseñanza Primaria. Por ejemplo, Yábar (1995) lo menciona como herramienta a utilizar en primaria para que posibilite y potencie la construcción de procesos cognitivos. Sin embargo, en México, Sacristán (2005, pp. 15 y 16), justifica la incorporación de la programación en lenguaje Logo en el currículo de matemáticas de la Enseñanza Secundaria con parecidos objetivos:

> *[…] este tipo de actividad conlleva el aprendizaje implícito de muchos conceptos, ideas y razonamientos matemáticos. […] El estudiante usa y expande sus habilidades de razonamiento lógico, de resolución de problemas y de análisis y síntesis, además de utilizar nociones como secuencialidad, modularidad y repetición.*

Yábar (1995), insiste en que la interacción entre informática y matemáticas puede hacerse en todas las etapas educativas (desde Primaria a la Universidad), con objetivos adaptados a cada una de ellas. De esta forma, será posible en Bachillerato plantearse un objetivo como el que persigue este trabajo, introduciendo lenguajes de programación más complejos para la enseñanza-aprendizaje de la geometría.

- **El uso de lenguajes de programación específicos de tipo funcional**

Dentro de este apartado se encuentra el uso de lenguajes de programación de muy alto nivel específicos en otros apartados de la matemática, como los trabajos realizados por Dubinsky (1995) y Da Rosa (2001 y 2012) dentro de la matemática discreta con lenguaje funcional ISETL, un lenguaje de programación inspirado en el lenguaje matemático creado por Jack Schwartz y cuyos primeros usos en el ámbito

[1] http://cabri.com/es/

[2] http://www.geogebra.org/cms/

[3] http://el.media.mit.edu/logo-foundation/index.html

educativo de las matemáticas se remontan a 1981 en las universidad de Clarkson. Posteriormente se extendió tanto en ámbitos universitarios como de bachillerato y ya en 1995 varios miles de estudiantes usaban ISETL en el aprendizaje de las matemáticas. Aunque estos autores centran sus investigaciones en la matemática discreta, ellos mismos indican que su metodología es aplicable a otras ramas de las matemáticas o las ciencias, como por ejemplo a la geometría de bachillerato.

- **La iniciativa mexicana ECIT·EMAT**

Tras revisar diversas iniciativas donde se justifica el uso de los lenguajes de programación como recurso en la clase de Matemáticas, especialmente interesante es el programa ECIT·EMAT (Enseñanza de la Ciencia y las Matemáticas con Tecnología) de México[4] para este trabajo, ya que une la enseñanza-aprendizaje de la geometría con el uso de dicho recurso.

Dentro de este programa, Ursini y Orendain (2000) coordinaron un estudio experimental sobre enseñanza-aprendizaje de geometría en varias aulas de México mediante la plataforma gráfica Cabri, para lo que publicaron el libro de texto referenciado que es de distribución gratuita. En dicho texto (pp. 9 y 10), se mencionan los principales objetivos de EMAT, enmarcados en el Programa de Modernización Educativa:

- Elevar la calidad de la enseñanza de las matemáticas en la escuela secundaria.
- Impulsar la formación de profesores de matemáticas de este nivel escolar.
- Promover el uso de las nuevas tecnologías en la educación.
- Generar y actualizar métodos y contenidos educativos de la matemática escolar.

De igual manera, Sacristán (2005) ha publicado un libro de texto para este programa sobre enseñanza de las matemáticas a través de la programación en Logo, que fue puesto a prueba de varias escuelas del Distrito Federal de México con financiación del Conacit (Consejo Nacional de Ciencia y Tecnología) y la supervisión del

[4] http://www.efit-emat.dgme.sep.gob.mx/

Cinvestav (Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional).

- **Iniciativas patrocinadas por empresas informáticas**

Existen multitud de iniciativas, donde entidades privadas están realizando un importante esfuerzo por generar recursos de carácter educativo, también en los casos en los que se usan los lenguajes de programación como base del recurso. En este apartado podemos englobar dos actuaciones de empresas norteamericanas de informática:

- Google y su iniciativa CS4HS (Computer Science for High School).
- Microsoft y su iniciativa DreamSpark.

La iniciativa de Google comenzó como un esfuerzo conjunto de la Universidad Carnegie Mellon, UCLA, y la Universidad de Washington para ayudar a la introducción de las Ciencias de la Computación y el Pensamiento Computacional en las escuelas de enseñanza media y superior. Se organiza a través de talleres de verano dirigidos a los profesores y centralizados en las universidades a las que pertenecen las escuelas. Google aporta fondos por valor de hasta 15.000 USD por escuela para la realización de estos talleres.

Por otro lado, la iniciativa de Microsoft pretende poner en manos de estudiantes y profesores tanto de instituto como universitarios, todas las herramientas de desarrollo de la compañía de forma gratuita. Las instituciones educativas adscritas al programa también reciben servidores y soporte técnico.

- **Apoyos institucionales a la incorporación real de las TIC en los procesos de enseñanza-aprendizaje en las aulas: España y la Escuela 2.0**

El programa estatal Escuela 2.0 se inició en el año 2009, aunque para su implantación en el territorio nacional, es necesario el establecimiento de un convenio de colaboración con cada respectiva Comunidad Autónoma.

Pero no todas las Comunidades Autónomas se adhirieron a este programa, ni firmaron cuerdo de colaboración con el Estado. Es el caso de la Comunidad de Madrid o la Comunidad Valenciana, si bien ambas tienen sus propios planes de incorporación de las TIC en las aulas: por ejemplo, a través de la Orden 1275/2010 de su Consejería de Educación, Madrid estableció su propia versión de esta Escuela 2.0, con un carácter más experimental y restringido a los denominados *institutos de innovación tecnológica*. En todo caso, la propuesta educativa incluida en este trabajo

es lo suficientemente flexible como para tener cabida tanto en las Comunidades Autónomas adheridas a Escuela 2.0 como al resto.

Escuela 2.0 es un proyecto de integración de las Tecnologías de la Información y de la Comunicación (TIC) en los centros educativos, que contempla el uso personalizado de un ordenador portátil por parte de cada alumno (a lo que se suele denominar modelo 1:1 y que, de acuerdo con Valiente (2010), tuvo sus orígenes en el estado de Maine, EE.UU., en 2002).

Se debía desarrollar en varias fases a lo largo de varios cursos, pero en la actualidad hay una cierta incertidumbre sobre el abandono de este programa y su sustitución por un nuevo enfoque.

Dentro de este programa Escuela 2.0, el Ministerio de Educación se apoyó en el Instituto de Tecnologías Educativas, ITE, denominado en la actualidad Instituto Nacional de Tecnologías Educativas y de Formación del Profesorado, INTEF[5].

Instituto Nacional de Tecnologías Educativas y de Formación del Profesorado

Los objetivos del INTEF, tal como se recogen en su página web, son:

- *Elaboración y difusión de materiales curriculares y de apoyo y de formación del profesorado y el diseño y la realización de programas específicos, en colaboración con las Comunidades Autónomas, destinados a la actualización científica y didáctica del profesorado.*
- *Elaboración y difusión de materiales en soporte digital y audiovisual de todas las áreas de conocimiento.*
- *La realización de programas de formación específicos, en colaboración con las Comunidades Autónomas, en el ámbito de la aplicación en el aula de las Tecnologías de la Información y la Comunicación.*
- *El mantenimiento del Portal de recursos educativos del Departamento y la creación de redes sociales para facilitar el intercambio de experiencias y recursos entre el profesorado.*

En relación a este último objetivo y referido al ámbito de las matemáticas, en la página web del INTEF[6] se encuentran diversos recursos educativos. Entre los más

[5] http://www.ite.educacion.es/

[6] http://ntic.educacion.es/v5/web/jovenes/matematicas/

relacionados con este trabajo, existen cursos de geometría apoyados en Cabri o GeoGebra.

El futuro de programas como Escuela 2.0 a nivel estatal o autonómico está en entredicho tal como reconoce el Ministerio de Educación, Cultura y Deporte (ha transformado dicho programa hacia otros modelos como TIC 2012[7]).

2.3 Metodología a emplear

La metodología empleada en este trabajo se puede resumir en el siguiente esquema:

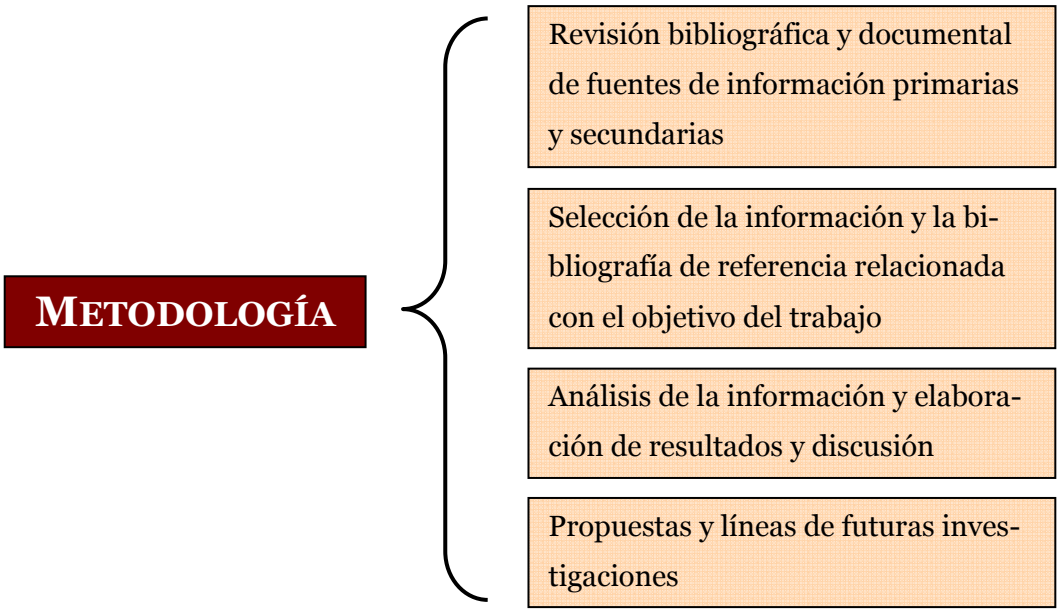

Figura 1. Metodología. Fuente: Elaboración propia

El objetivo de este trabajo, como ya se ha indicado, es analizar cómo se pueden desarrollar procesos de enseñanza-aprendizaje de las Matemáticas a través del uso de los Lenguajes de programación en Bachillerato. Este objetivo no está actualmente cubierto, por lo que las dos primeras fases (revisión bibliográfica y documental, y selección de información y bibliografía) se centrarán en realizaciones similares o equivalentes que puedan servir de base a este trabajo.

La tercera fase (análisis de la información y elaboración de resultados y discusión) conlleva una síntesis de los datos obtenidos sobre la situación de estudio, así como su discusión y crítica.

[7] http://www.educacion.gob.es/horizontales/prensa/notas/2012/04/20120404-presupuestos.html

La última fase (propuestas) nace de la consideración de que este trabajo se engloba dentro de una estrategia más amplia, que deberá desarrollarse y completarse en trabajos futuros a realizar por la comunidad educativa.

3 Desarrollo

En los siguientes apartados de este capítulo se irán analizando diversos aspectos necesarios para alcanzar el objetivo de este trabajo (analizar cómo se pueden desarrollar procesos de enseñanza-aprendizaje de la Geometría a través del uso del lenguaje de programación C#).

Todos ellos serán sintetizados en el capítulo de Propuestas, en donde se realizará un esbozo de realización práctica de la enseñanza en Bachillerato de Geometría a través del aprendizaje del lenguaje de programación C#.

3.1 Desarrollo gradual

La implantación de una metodología de enseñanza – aprendizaje de geometría a través de la programación en C# deberá realizarse de una forma gradual. Es el mismo camino seguido por ejemplo por ECIT·EMAT[8] (2012), que en su página web menciona que:

> *Este proceso [de implantación] debe ser gradual, por lo cual se optó en desarrollar una fase experimental para contar con evidencia sobre un modelo didáctico y pedagógico probado antes de valorar la pertinencia de una incorporación generalizada de las TIC.*

De manera similar, Valiente (2010), en su informe sobre la implantación internacional de iniciativas 1:1 en educación (un ordenador personal por cada alumno), manifiesta que muchos países han optado primero por experiencias piloto a pequeña escala.

De forma complementaria Ursini y Orendain (2000) constatan la importancia de la especialización de los usuarios de la tecnología (alumnos y maestros) en una herramienta, de tal forma que logren dominarla y, al mismo tiempo, la empleen en la enseñanza y aprendizaje de temas curriculares específicos. Eso implica que una implantación gradual debe ir acompañada de una continuidad en el tiempo, de forma que la experiencia se perfeccione y sea posible obtener un análisis coherente de la experiencia, que permita su generalización en fases posteriores. Por ello, para que la enseñanza de la geometría a través de lenguajes de programación en la etapa de bachillerato pueda mostrar su viabilidad y utilidad, es importante que su implantación sea gradual y que tenga una continuidad en el tiempo, para así poder analizar

[8] Programa de Enseñanza de la Ciencia y las Matemáticas con Tecnología desarrollado en México y explicado previamente.

mejor los resultados y realizar las modificaciones y ajustes necesarios que surjan de ese análisis.

3.2 Lenguaje matemático y lenguaje de programación

Cada ciencia o rama del conocimiento, tiene su propio lenguaje. Aunque en muchos casos, ese diferente lenguaje se traduce exclusivamente en una terminología y vocabulario propios, en el caso de las matemáticas y los lenguajes de programación, esa diferencia es mucho más profunda, incluyendo una sintaxis y simbología propias.

En este apartado se analizan las similitudes entre el lenguaje matemático y el de programación, de forma que es factible su aprendizaje conjunto en un ámbito de apoyo mutuo, como apunta Yábar (1995).

Dubinsky (1995, p. 1), menciona la siguiente cita de Jack Schwartz, creador de ISETL:

> *El conocimiento [matemático] adquirido [por la programación en SETL] es equivalente al que se adquiriría en un curso de primer nivel en Matemática Discreta.*

La idea es que mucha más gente consigue aprender programación que ciertos conceptos de la matemática discreta (conjuntos, secuencias, funciones algebraicas, proposiciones...) con los que trabaja ISETL y que, por tanto, los alumnos aprenden mejor con este lenguaje. El autor da el siguiente ejemplo de cómo indicar el conjunto de los números primos menores de 100:

- En lenguaje matemático:

$$\{x : x \in \{2,3,\ldots,100\} \mid (\nexists y \in \{2,3,\ldots,x-1\} \ni x \bmod y = 0)\}$$

- En lenguaje ISETL:

```
{x : x in {2..100}|(not exists y in {2,3..x-1}|x .mod y=0)}
```

Tal como insiste Dubinsky (1995), la similitud de estos dos lenguajes hace que al escribir y ejecutar este código, los estudiantes aprendan mejor el concepto matemático abstracto al facilitar su concreción y manipulación. De hecho, el estudiante puede imaginar qué ocurre en el ordenador al procesar esta instrucción, de manera similar a cómo el docente desearía que ocurriera en la mente del estudiante.

Para Dubinsky (1995), al estudiar los procesos mentales que se producen en el proceso de enseñanza – aprendizaje de conceptos matemáticas, se observa frecuentemente que éstos se fomentan con la escritura de programas informáticos.

Lenguajes de programación matemáticos

Tanto Dubinsky (1995) como Da Rosa y Cirigliano (2001) conciben ISTEL como un lenguaje de programación matemático o funcional, especialmente concebido para transcribir el lenguaje matemático. Realmente ISETL es una evolución de SETL (ISETL proviene de Interactive SETL) enfocado específicamente para el ámbito educativo cuya versión 1.0 data de 1988. Dubinsky (1995) indica que dentro de este grupo de lenguajes de programación podrían englobarse otros lenguajes con tal que cumplan una serie de características:

- **Su sintaxis es razonablemente similar a uno de los estándares de notación matemática**

  Debería ser posible escribir el código de forma que los elementos extraños al concepto matemático que transcriben sean muy escasos. Por ejemplo, para obtener un vector V como suma dos vectores V_1 y V_2, en lenguaje matemático se escribe:

  $$V = V_1 + V_2$$

  En un lenguaje de programación que cumpla este requisito, por ejemplo C#, se puede transcribir, de forma muy similar al lenguaje matemático, como[9]:

  ```
  V = V1 + V2;
  ```

  Por el contrario, en un lenguaje de programación que no cumpla este requisito, como por ejemplo C, la transcripción se aleja bastante del lenguaje matemático:

  ```
  for (i=0; i<NDIM; i++)
     V[i] = V1[i] + V2[i];
  ```

- **Algunas características matemáticas son soportadas junto con sus propiedades**

  Esto incluye conjuntos y listas de número finito de elementos con elementos de cualquier tipo (incluyéndose a sí mismos), sentencias lógicas, funciones como procedimientos, pares ordenados y relaciones definidas como pares

[9] Diseñando una clase 'Vector' y sobrecargando el operador '+' para acepte dos vectores como argumentos y el vector suma como retorno.

ordenados. Por ejemplo, en lenguaje matemático se puede definir el conjunto Z_{12} como el subconjunto de los números enteros de módulo 12, es decir, los números entre el 0 y el 11. En un lenguaje de programación que respete este principio, como ISETL, se define como:

```
Z12 := {0..11}
```

En un lenguaje de programación que no respete este criterio, como C, su definición no es tan directa (aparecen muchos más símbolos y palabras que no provienen del lenguaje matemático) ni tan precisa (es una lista ordenada, no un conjunto de elementos):

```
int Z12[12] = {0,1,2,3,4,5,6,7,8,9,10,11};
```

- **Todos los tipos de datos son objetos de primera clase, en el sentido que pueden usarse directamente en cualquier expresión que matemáticamente tenga sentido**

  En particular, los conjuntos y listas pueden tener elementos de cualquier tipo y las funciones pueden tener como entrada y salida otras funciones. Es decir, por ejemplo, si matemáticamente tiene sentido sumar números, fracciones, vectores o polígonos, en el lenguaje de programación debe poder hacerse como una simple operación con el símbolo '+', como se ha mostrado anteriormente con la suma de dos vectores.

Es decir, para estos autores, en el ámbito de la enseñanza de Matemáticas a través de lenguajes de programación, se busca un lenguaje de programación que permita escribir (o describir) operaciones y propiedades matemáticas con una sintaxis lo más similar posible o como se escribiría en lenguaje matemático.

ISETL cumple estas especificaciones, siendo el lenguaje que mejor lo hace (no en vano ISETL fue creado específicamente para ello). Dubinsky (1995) comenta que MAPLE se aproxima razonablemente a estas especificaciones, y no duda que surgirán otros lenguajes que también evolucionen en este sentido.

Respecto a los lenguajes que Da Rosa et al. (2001) denomina *imperativos* (Basic, Logo, Pascal, Fortran, Cobol, C...) para estos autores, los lenguajes funcionales tienen las siguientes ventajas y desventajas:

- Ventajas:
  - El uso de un lenguaje diferente al usado en entornos profesionales o a los lenguajes conocidos por los alumnos de etapas educativas anteriores, permite abordar los temas a un nivel conceptual mayor, a la vez

que reduce la heterogeneidad de los alumnos (todos parten de la misma base).

- Su sencilla sintaxis permite centrarse en los conceptos: si la sintaxis es similar a la de las matemáticas que se están estudiando, no se pierde tiempo y esfuerzo en su comprensión, focalizando la atención en las propias matemáticas.
- Algunos conceptos matemáticos importantes (recursión, listas…) poseen una transcripción en el lenguaje similar al lenguaje matemático, evitando la necesidad (como ocurriría con un lenguaje de propósito general) de aprender simultáneamente dos lenguajes (matemático y de programación) que sean demasiado diferentes entre sí.
- Refuerzan la idea de que la interacción entre matemáticas y ciencias de la computación es bidireccional, ayudando a los alumnos a entender la utilidad de las matemáticas en los lenguajes de programación y, por tanto, en la vida cotidiana.

- Desventajas:
  - Existe un rechazo inicial de los alumnos ante lenguajes no usados en el mercado o los que no conocen ni siquiera por el nombre. Por ejemplo, los alumnos de hoy en día tienen cada vez más dispositivos electrónicos (teléfonos inteligentes, tabletas…) basados en lenguajes como Java, hacia los que pueden tener una cierta motivación, que no existe hacia lenguajes más desconocidos.
  - También hay un rechazo inicial en el profesorado a formarse en estos lenguajes, algo imprescindible si han de utilizarlos como herramienta en la enseñanza – aprendizaje de las matemáticas.
  - El trabajo con ellos requiere una mayor abstracción que con lenguajes imperativos. Si bien esto es una desventaja desde el punto de vista que dificulta su aprendizaje, se puede también como una ventaja desde el punto de vista que facilita el aprendizaje del lenguaje, también abstracto, de las matemáticas.

Una vez establecida la cercanía existente entre el lenguaje matemático y ciertos lenguajes de programación, es necesario proponer un modelo de enseñanza-aprendizaje que permita aprovechar dicha proximidad en el desarrollo curricular de las matemáticas.

3.3 Modelo de enseñanza-aprendizaje de las Matemáticas a través del lenguaje de programación y su desarrollo curricular

Dubinsky (1995), establece la siguiente pauta para el diseño curricular de un determinado concepto matemático que se desee realizar apoyándose en la programación en ISETL, aunque es válido para cualquier lenguaje de programación y rama de la matemática.

- **Análisis teórico** basado en una teoría general de enseñanza aprendizaje (Dubinsky se apoya en teorías de tipo constructivista, como se indica más adelante), el propio conocimiento del educador sobre el concepto matemático en cuestión y en su experiencia en la enseñanza-aprendizaje por medios 'convencionales'. De esta manera se realiza una aproximación preliminar a cómo plantear el aprendizaje y el desarrollo mental que deberá realizar el alumno.
- **Diseño e implementación** de las actividades (basadas en el desarrollo de programas de ordenador en el lenguaje elegido) encaminadas a que los estudiantes consigan el desarrollo mental propuesto. Al experimentar este proceso, los alumnos adquieren el conocimiento por diferentes canales, tanto cuantitativos como cualitativos.
- **Evaluación**. Finalmente, debe haber una coordinación entre el conocimiento empírico conseguido y el conocimiento que en el análisis teórico se ha presupuesto. Cuando este proceso de evaluación no es satisfactorio deberán analizarse las causas y realizar las correcciones necesarias.

Dubinsky (1995, p. 7) lo resume en la figura 2 (que no difiere mucho del proceso seguido en la elaboración de cualquier unidad didáctica) y en la que se basa la propuesta de enseñanza-aprendizaje desarrollada por este autor.

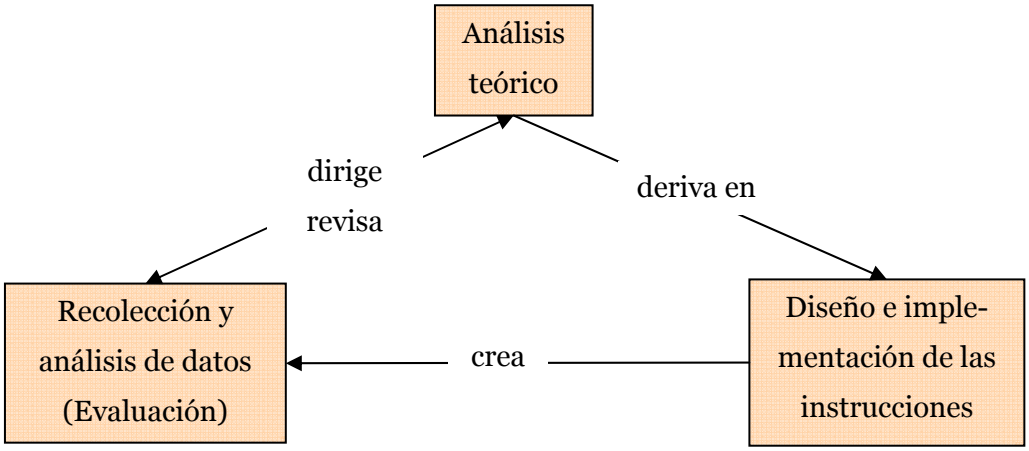

Figura 2. Investigación y desarrollo curricular.
Fuente: Elaboración propia a partir de Dubinsky (1995)

El conocimiento matemático y su adquisición

Para Dubinsky (1995, pp. 8 y 9), la adquisición del conocimiento matemático es diferente al de otras materias. Basándose en las teorías constructivistas y en las ideas de Piaget (1992), resumiéndola en la siguiente frase y en el modelo conceptual de la figura 3:

> *"Un conocimiento matemático de un individuo es su tendencia a responder a situaciones o problemas matemáticos mediante la reflexión sobre ellos en un contexto social y construir o reconstruir acciones matemáticas, procesos y objetos, organizándolos en esquemas para utilizarlos en el tratamiento de dichas situaciones o problemas" (Dubinsky, 1995, pp. 8 y 9).*

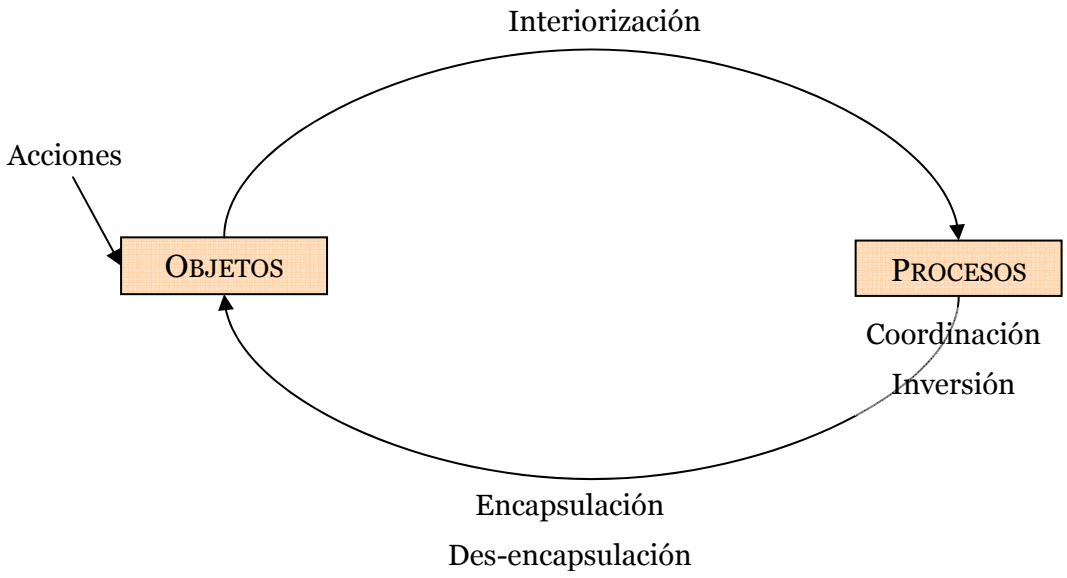

Figura 3. Construcciones para el conocimiento matemático.
Fuente: Elaboración propia a partir de Dubinsky (1995)

Enlazando a Dubinsky con las fases del pensar expuestas por Carrasco et al. (2008), se puede decir que para el conocimiento de un concepto, tan importante es aprenderlo (etapa adquisitiva del pensar) como acceder a él (etapa reactiva del pensar), siendo la reflexión fundamental en ambas tareas (fase reflexiva de la etapa adquisitiva y fase extensiva de la etapa reactiva). Dubinsky relaciona este concepto con la dicotomía asimilación / acomodación establecida por Piaget en su obra *Los Principios de la Epistemología Genética* de 1972.

Es decir: conocer un concepto matemático no es solo saber calcular o saber aplicar algoritmos matemáticos, sino que es fundamental saber variar o adaptar esas capacidades en la resolución de problemas nuevos. Por tanto, poseer un conocimiento es tener una tendencia hacia construir procesos mentales para resolver un problema, lo que implica reconstruir y adaptar lo ya adquirido y por tanto avanzar en el conocimiento.

Dubinsky (1995), apoyándose en al figura anterior, indica que el entender un concepto matemático comienza con la manipulación de construcciones mentales anteriores u objetos físicos para formar acciones; las acciones son entonces interiorizadas para formar procesos que son entonces encapsulados para formar objetos. Los objetos pueden ser des-encapsulados de nuevo en los procesos que los formaron. Finalmente, procesos y objetos se organizan en esquemas.

Es la denominada teoría APOS (Action, Process, Object, Schema, es decir, Acción, Proceso, Objeto y Esquema), teoría constructivista desarrollada fundamentalmente por este autor, que, de cuerdo con Tziritas (2011) y Dubinsky et al. (2001), engloba tanto una teoría de enseñanza – aprendizaje como una metodología de investigación de la enseñanza matemática. Concretando más los conceptos de acción, proceso, objeto y esquema, Dubinsky et al. (2001) indican:

Acción y proceso

La diferencia entre acciones y procesos es que una acción actúa sobre un objeto específico, mientras que un proceso es genérico y puede actuar sobre una determinada clase de objetos. Dubinsky (1995) lo explica a través de ejemplos realizados en lenguaje ISETL aplicados al álgebra. Aplicado al objetivo de este trabajo (geometría y C#) se puede realizar el siguiente ejemplo:

- **Acción**: obtener la longitud de la hipotenusa de un triángulo rectángulo, de catetos 4 y 3 cm.

$$L = \sqrt{4^2 + 3^2} = \sqrt{25} = 5\,\text{cm}$$

```
Double L = Math.Sqrt (4*4 + 3*3);
```

- **Proceso**: obtener la longitud de un segmento *AB* cualquiera.

$$L_{AB} = \sqrt{(x_B - x_A)^2 + (y_B - y_B)^2}$$

```
Double Lsegmento (Punto A, Punto B)
{
    Double dx = B.x - A.x;
    Double dy = B.y - A.y;
    return Math.Sqrt (dx*dx + dy*dy);
}
```

De esta forma, una vez entendida y asimilada la acción (que en términos de programación se identificaría como una *sentencia*), es posible que el alumno la generalice y describa el proceso a través del diseño de una *función*. Además, al diseñar esta función (este proceso) se puede mostrar al alumno el proceso llevado a cabo por el programa para obtener la longitud del segmento: se 'des-encapsula' de forma que se ejecutan las acciones necesarias sobre los datos concretos (los parámetros de la función), devolviéndose el valor requerido, evidenciándose su semejanza con el proceso mental de adquisición del conocimiento matemático.

Objeto

Los objetos se obtienen por encapsulado de procesos. Los estudiantes necesitan realizar esta encapsulación cuando reflexionan en una situación en la que una acción debe aplicarse sobre un proceso dinámico. Dubinsky (1995) indica que éste es un concepto difícil de entender por los alumnos y para el que hay pocas herramientas pedagógicas. Apoyándose en varios ejemplos desarrollados en ISETL sobre matemática discreta, propone algunas vías de actuación.

En el caso de un lenguaje de programación orientada a objetos como C#, es bastante más directo abordar este concepto, puesto que es muy similar al concepto de *clase*, que engloba y encapsula tanto los datos como las acciones y procesos que actúan sobre ellos. El siguiente ejemplo muestra una clase que encapsula los datos y operadores de un vector en el plano (extracto):

```
public class VectorLibre
{
    public Punto fin;
    public double modulo { get; set; } //Tamaño del vector
    public double angulo { get; set; } //Ángulo en radianes

    //Constructores
    public VectorLibre()
    {
        fin = new Punto();
        modulo = angulo = 0;
    }
```

```
    public VectorLibre(Punto i, Punto f);
    public VectorLibre(double angulo, double modulo);

    //Obiene las coordenadas polares a partir de las cartesianas
    void aPolares ();

    //-------------------------------------------------------------
    // Operadores sobrecargados
    //-------------------------------------------------------------
    // Comparación de igualdad
    public static bool operator ==(VectorLibre a, VectorLibre b);
    // Comparación de desigualdad
    public static bool operator !=(VectorLibre a, VectorLibre b);
    // Suma de vectores
    public static VectorLibre operator +(VectorLibre a, VectorLibre b);
    // Resta de vectores
    public static VectorLibre operator -(VectorLibre a, VectorLibre b):
    // Vector por un escalar: v * k
    public static VectorLibre operator *(VectorLibre v, double k);
    // Escalar por un vector: k * v
    public static VectorLibre operator *(double k, VectorLibre v);
    // Producto escalar de dos vectores
    public static double operator *(VectorLibre a, VectorLibre b);
    // ...
}
```

Esquema

Se puede entender un esquema como un programa organizado a través de acciones, procedimientos, objetos y otros esquemas para la resolución de un determinado tipo de problema. Para adquirir este concepto, Dubinsky (1995) pone como tareas a los alumnos la realización de un programa informático (en ISETL) que implemente un determinado concepto matemático y luego lo aplique a determinadas situaciones concretas. Dubinsky et al. (2001) pone el acento en que un esquema es una suerte de 'marco de trabajo' que la mente del alumno pone en acción para resolver un problema o situación para la que dicho esquema es adecuado.

En ese sentido, es obligado recordar que la competencia matemática implica, entre otras cosas, la capacidad del alumno para resolver problemas vinculados con la realidad. Esa resolución podrá llevarla a cabo, precisamente, mediante los esquemas mentales que previamente ha asimilado o que ha utilizado en la elaboración de un programa informático que le ayude a resolver dicho problema.

El ciclo ACE de enseñanza

Dubinsky (1995) ha utilizado con éxito, en la enseñanza de las matemáticas a través de los lenguajes de programación, el ciclo ACE (Activities, Class, Excerceses en inglés ó Actividades, Clase y Ejercicios en castellano). Para ello dividió el curso en temas de trabajo de una semana de duración, compatibilizando la enseñanza con

ordenador como sin él. Las tareas se completan fuera de clase. Todas las etapas se plantearon como trabajo en grupo.

Actividades

Las actividades se plantean en la sala de ordenadores, en actividades de computación diseñadas para adquirir una determinada construcción mental. En general, el tiempo asignado es insuficiente, por lo que debería completarse fuera del horario normal o con el ordenador personal del alumno.

Clase

En la clase ordinaria, se continúa con la labor anterior en la sala de ordenadores, pero con lápiz y papel. Se completa con puestas en común entre los diferentes grupos sobre los cálculos realizados. El profesor, a veces, da definiciones y realiza explicaciones, con objeto de afianzar y fijar los conocimientos adquiridos.

Ejercicios

Los ejercicios, para trabajar en equipo, son más o menos tradicionales y deben completarse en casa. El objetivo de estos ejercicios es reforzar las ideas que los alumnos han construido, para que hagan uso de las matemáticas que han aprendido y, en ocasiones, para que comiencen a pensar en los próximos temas que se abordarán en clase.

3.4 Aspectos a considerar en el planteamiento de la propuesta educativa

Aprendizaje colaborativo

Como se ha mencionado en el apartado anterior, Dubinsky defiende, para el ciclo ACE de enseñanza, un aprendizaje colaborativo. La razón es que Dubinsky (1995) sostiene, apoyándose entre otros en Vidakovic, D. (1993) *Differences between Group and individual processes of construction of the concept of inverse function,* Purdue University, que la reflexión necesaria para adquirir y utilizar los conocimientos matemáticos se consigue mejor en entornos con interacción entre estudiantes (es decir, en grupo) tanto antes como durante y después de realizar el trabajo matemático.

Ursini y Orendain (2000, p. 10), también indican como una de las características de la iniciativa mexicana ECIT·EMAT, la puesta en práctica de un modelo de

cooperación para el aprendizaje de las matemáticas con el ordenador. De esta forma se promueve la discusión y el intercambio de ideas. La labor del profesor es para estos autores la de promover el intercambio de ideas y la discusión en grupo, y al mismo tiempo actuar como mediador entre el estudiante y la herramienta, asistiendo a los estudiantes en su trabajo con las actividades de clase y compartiendo con ellos el mismo medio de expresión.

Trabajo multidisciplinar

Da Rosa et al. (2001) reconoce que si bien los esfuerzos realizados en la enseñanza de las matemáticas con lenguajes de programación suponen un avance y una mejora, no solucionan el problema de la dificultad en el aprendizaje de muchos alumnos ni de las matemáticas ni de las ciencias de la computación. Estos autores sugieren, por el contrario, que la solución está en el trabajo conjunto de docentes de ambas disciplinas para crear y diseñar nuevos cursos en los que la relación entre las matemáticas y las ciencias de la computación "debe ser enfatizada explícitamente", en palabras de estos autores.

3.5 Ciencias de la Computación sin ordenador

Dentro de la iniciativa de Google (2010-2012) CS4HS, Stalvey (2012), del departamento de ciencias de la computación del College of Charleston ha desarrollado un trabajo denominado *CS Unplugged*, cuyo objetivo es desarrollar un conjunto de actividades sobre ciencias de la computación para introducir a los estudiantes en el pensamiento computacional sin necesidad de utilizar ordenadores. De esta forma, Stalvey quiere enfatizar que las ciencias de la computación no estudian el hardware, sino el pensamiento computacional. Fruto de este trabajo ha surgido un portal con actividades de uso gratuito (http://csunplugged.org/) y un libro de texto de apoyo tanto para alumnos como para profesores en varios idiomas (incluyendo el español) que puede descargarse de dicho portal.

Este enfoque puede adoptarse perfectamente para los objetivos de este trabajo (razón por la que se ha situado en este apartado y no en el de antecedentes), de forma que se reduce la brecha digital y se reducen también las necesidades de financiación para implantarlo. De esa forma, se aumenta la audiencia potencial de este trabajo, mediante su adaptación al entorno social y económico de cada caso.

4 Propuestas educativa

4.1 Por qué C#

En este apartado se analizan las características de C# (en castellano se lee 'ce sharp') que pueden ser importantes para la enseñanza – aprendizaje conjunto del propio lenguaje y de conceptos matemáticos relacionados con la geometría, recordando que es este último aspecto el principal objetivo de este trabajo.

En todo caso, tal como indica Schildt (2010), hay que tener en cuenta que los elementos y características de cualquier lenguaje de programación no se dan de forma aislada, sino que están interrelacionados entre sí y tienen sentido en su conjunto (lo cual es también aplicable a la enseñanza de la geometría o de otros contenidos de Matemáticas).

Programación orientada a objetos

La programación orientada a objetos (*Object Oriented Programming, OOP*) está en el núcleo de C#. A lo largo de la historia de los lenguajes de programación, éstos han ido cambiando su metodología al ir avanzando la complejidad de los proyectos a realizar. Para Schildt (2010), las diferentes etapas fueron:

- **Lenguaje máquina**. Los primeros programas informáticos se desarrollaron en *lenguaje máquina*: unas pocas decenas de instrucciones escritas en el 'lenguaje' interno del procesador del ordenador, a base de ceros y unos. Era totalmente críptico e imposible de interpretar para una persona.
- **Ensamblador**. El siguiente paso fue el lenguaje ensamblador. A partir de unas pocas 'palabras' se podía sustituir el lenguaje máquina a algo más 'legible'. De esta forma los programas podían tener unos cuantos centenares de líneas, aumentando por tanto su potencia y versatilidad.
- **Lenguajes de alto nivel**. Cuando el lenguaje ensamblador llegó a su límite de aplicación práctica aparecieron los lenguajes de alto nivel tipo FORTRAN (utilizado sobre todo en el campo científico) y COBOL (utilizado sobre todo en al campo de la gestión). Poseían nuevas herramientas que permitían ya realizar programas de una cierta envergadura.
- **Lenguajes estructurados**. Al alcanzar su límite, fueron sustituidos por lenguajes más estructurados como C. Éstos permitían subdividir los progra-

mas en partes (procedimientos o funciones) interdependientes entre sí, lo que permitía alcanzar una mayor complejidad.

- **Lenguajes orientados a objetos**. Finalmente, cuando esta metodología llegó a su límite, es decir, era complejo realizar y sobre todo mantener programas cada vez más grandes y complejos, surgieron los lenguajes orientados a objetos.

En cada una de estas etapas de la historia de los lenguajes de programación, se partía de lo mejor que tuvieran las etapas anteriores, incorporando nuevas herramientas y maneras de aproximarse al problema de la programación. Así, los lenguajes OOP, parten de lo mejor de los lenguajes estructurados, combinándolo con nuevos conceptos, dando como resultado una nueva y mejor forma de organizar los programas[10].

Partiendo del título de un texto clásico de programación, "Algoritmos más estructuras de datos igual a programas"[11], se puede decir que los lenguajes estructurados ponen su énfasis en el código (los algoritmos), es decir, en el qué pasa (Schildt lo resume en "el código actúa sobre los datos"), mientras que los lenguajes orientados a objetos ponen el énfasis en los datos, es decir, en el cómo se afectan los datos (Schildt lo resume en "los datos controlan el acceso al código"). Dicho de otra forma: al diseñar un programa en lenguaje estructurado, se piensa en el algoritmo que se desea desarrollar, añadiéndole después las variables y parámetros (los datos) necesarios, mientras que en un lenguaje de programación orientado a objetos, se empieza diseñando los datos de cada objeto y posteriormente se añaden los métodos (los algoritmos) que actúan sobre esos objetos.

Para el objetivo de este trabajo, un lenguaje OOP es esencial, pues permite trabajar con objetos matemáticos o geométricos a través de su representación como objetos del lenguaje (podemos crear y manipular objetos 'punto', 'línea', 'vector', 'polígono', 'triángulo', etc.). Ésta es una de las características que Dubinsky (1995) define como básicas para poder decir que un lenguaje de programación es idóneo para manejar el lenguaje matemático.

---

[10] No en vano, el nombre C# proviene de la nota musical do sostenido (para los anglosajones, la A es el la, la B es el sí, la C es el do y así sucesivamente) dando a entender que C# está [un semitono] por encima del lenguaje C.

[11] Wirth, N., (1986?). *Algoritmos más estructuras de datos igual a programas*. Madrid: Ediciones del Castillo

Todo lenguaje de programación orientado a objetos poseen tres características comunes: encapsulación, polimorfismo y herencia.

Encapsulación

La *encapsulación* es un mecanismo de programación que permite unir en un único objeto los datos y el código que los manipula, de forma que quedan aislados del exterior y previenen su uso inadecuado, al exponer solo funciones (*métodos*, en la terminología del lenguaje) que controlan su manipulación. La unidad de encapsulación de C# es la *clase*.

En el ámbito de este trabajo, por ejemplo, es posible crear una clase *vector* que encapsule sus datos (las coordenadas de su origen y fin) junto con las operaciones que son legales en un vector (suma y resta de vectores, multiplicación por un escalar, producto escalar, traslación, giro, etc.).

Polimorfismo

El *polimorfismo* es la cualidad que permite a un *método* de poder actuar de una misma forma sobre muchos tipos de objetos. Como ejemplos en el ámbito de interés de este trabajo, un método llamado *Área* podría obtener el área tanto de un objeto *triángulo* como de un objeto *rectángulo* o *círculo*. Del mismo modo, el operador '+' puede actuar sobre números u objetos tipo *vector* o *polígono*.

Ésta es otra de las características que Dubinsky (1995) define como básicas para poder decir que un lenguaje de programación es idóneo para manejar el lenguaje matemático, ya que permite crear y manipular del mismo modo listas de cualquier tipo de objetos (que es el ejemplo mencionado por este autor).

De esta forma, se consigue que el lenguaje de programación se aproxime al lenguaje matemático, facilitando el principal objetivo de este trabajo.

Herencia

La herencia es el proceso por el cual un objeto puede heredarías características de otro objeto. Esto es importante porque supone utilizar el concepto de clasificación jerarquizada. Este concepto también es común en matemáticas: ciñéndonos a la geometría, del concepto genérico *polígono* se pueden derivar los conceptos *triángulo* y *cuadrilátero*. A su vez, el *triángulo* se puede subdividir en *equilátero*, *isósceles* y *escaleno*. Es decir, se puede entender esta herencia de clases como subconjuntos sucesivos que suponen una especialización progresiva.

Las características y funcionalidades comunes estarán en la parte superior de la clasificación (por ejemplo, todos los polígonos tienen área) y las específicas, solo en las clases heredadas (siguiendo con el mismo ejemplo, solo los polígonos de más de tres lados tienen diagonales).

4.2 Enseñanza en Bachillerato de Geometría a través del aprendizaje de C#: Un ejemplo

Este apartado sirve de síntesis del capítulo de desarrollo y pretende mostrar cómo aplicar este trabajo a nivel práctico en el aula. Como ya se ha indicado, es necesario que esta propuesta práctica se englobe dentro de una estrategia interdisciplinar entre ciencias de la computación y matemáticas, de forma que aquí no se entrará en los detalles de la enseñanza-aprendizaje del lenguaje de programación C#, sino que se hace hincapié en su utilización como recurso para la enseñanza-aprendizaje de las matemáticas. Para seguir este apartado del presente trabajo se ha intentado que no sea imprescindible conocer el lenguaje de programación C# o los conceptos de geometría que pretende enseñar, aunque ciertamente sí es muy aconsejable.

Para esta propuesta, se adoptará el ciclo de enseñanza ACE (Actividades, Clase y Ejercicios) descrito en el capítulo de desarrollo, aunque con algunas variaciones respecto a la propuesta de Dubinsky (1995):

- La etapa de actividades, para Dubinsky, tenían lugar en la sala de ordenadores. La propuesta que plantea este trabajo pretende ser versátil, por lo que se plantea la posibilidad de que se lleve a cabo en la sala de informática o en la propia clase, dependiendo de la dotación de herramientas informáticas de que esté dotado el centro. No es necesario que cada alumno cuente con un ordenador (sistema 1:1) pero sí que exista un ordenador por cada 2 o 3 alumnos. El profesor, sin embargo, podría realizar su labor con pizarra tradicional, aunque es aconsejable una pizarra electrónica o un sistema de proyección que le permita mostrar el contenido de su ordenador.
- Además, de acuerdo a la propuesta de Stalvey (2012), parte de las tareas de programación en C# podrían realizarse en el aula, con lápiz y papel (tal como se programaba en los albores de la informática, cuando los ordenadores no tenían ni pantalla ni teclado, los programadores realizaban los programas en papel, posteriormente eran transcritos en tarjetas perforadas por un equipo de 'perforistas' y finalmente el ordenador leía las tarjetas, compilaba el pro-

grama, lo ejecutaba e imprimía los resultados en papel continuo). De esta forma, la necesidad de disponibilidad de ordenador para los alumnos puede reducirse a momentos concretos del desarrollo de la unidad docente.

- De manera similar, los ejercicios para casa se pretende que sean lo suficientemente sencillos para que puedan realizarse con lápiz y papel, aunque la tarea de autoevaluación de ducho trabajo, a realizar ya en clase, convendría que se hiciese con el ordenador.

Como ejemplo de desarrollo, se plantea esbozar el desarrollo de la Unidad Didáctica sobre **vectores en el plano**, correspondiente al segundo epígrafe del Bloque 2 (Geometría) de los contenidos mínimos correspondientes a la asignatura de Matemáticas I de 1º de Bachillerato en su modalidad de Ciencias y Tecnología (RD 1467/2007). Véase también Santillana, (2012).

Las tareas a realizar en C#, gracias a las posibilidades de encapsulación y modularidad que presenta este lenguaje, se pretende que sean de extensión pequeña (muchas tareas, cada una de ellas de pocas líneas de extensión), por lo que las explicaciones del profesor pueden llevarse a cabo tanto en una pizarra convencional como en una pizarra digital o incluso transmitiéndolas a los ordenadores de los alumnos. De esta manera se pueden adaptar a los recursos disponibles y aumenta la disponibilidad y viabilidad de esta propuesta.

No obstante, dentro de un marco multidisciplinar de colaboración con el Departamento de Tecnología, sería recomendable realizarlas con ordenador de forma que también sirvan para la enseñanza – aprendizaje del propio lenguaje como parte del currículo de dicho departamento.

Contenidos

Los contenidos de esta unidad docente de Vectores en el Plano, relacionándolos con los contenidos del lenguaje de programación C# implicados, son:

VECTORES DEL PLANO	PROGRAMACIÓN EN C#
El conjunto R × R. Operaciones	Creación de una clase: La clase Punto • Asignación y salida de resultados
Vectores libres. • Módulo, dirección y sentido • Operaciones: suma y resta • Escalado y división	Clases como datos de otras clases: La clase VectorLibre • Métodos de las clases Sobrecarga de operadores • Unitarios: igualdad • Binarios: suma, resta…
Producto escalar. Propiedades y aplicaciones	Ampliación de la clase VectorLibre
Dependencia lineal de vectores. Bases. Coordenadas	Parámetros de funciones por valor y por referencia
Vectores deslizantes y vectores fijos	Herencia: la clase VectorFijo derivada de la clase VectorLibre

Tabla 1. Contenidos de la Unidad Didáctica

Objetivos educativos

Cognitivos

- Reconocer el conjunto R × R y sus elementos, utilizar su relación con los puntos del plano. Aprender a diseñar una clase Punto con los datos y características de un punto del plano.
- Utilizar los conceptos de vector libre, módulo, dirección y sentido, distinguir si dos vectores son iguales y calcular las componentes de un vector dados sus extremos. Aprender a diseñar la clase VectorLibre a partir de la clase Punto de forma que tenga los mismos datos y propiedades, y que se pueden hacer las mismas operaciones y obtener los mismos resultados que con un vector libre.
- Realizar operaciones de suma de vectores y producto por un número real, así como combinaciones lineales de vectores. Aprender a definir esas operaciones dentro de la clase VectorLibre a través de la sobrecarga de operadores.
- Distinguir si dos vectores en el plano son linealmente dependientes o independientes y si forman base, y obtener las coordenadas de un vector cualquiera en una base dada. Aprender a diseñar métodos que operen con ele-

mentos de la clase VectorLibre para realizar esas comprobaciones y obtener esos datos.

- Calcular el producto escalar de dos vectores, y utilizar su interpretación geométrica y sus propiedades para resolver problemas. Sobrecargar el operador '*' para definir el producto escalar.
- Aplicar el producto escalar al cálculo del módulo de un vector, del ángulo de dos vectores y a demostrar el teorema del coseno. Realizar esas mismas operaciones con la clase VectorLibre.

Procedimentales

- Reconocer el conjunto R × R y su relación con los puntos del plano.
- Utilizar los conceptos de vector libre, módulo, dirección y sentido en distintos contextos y determinar la existencia o no de equipolencia entre dos vectores.
- Realizar sumas de vectores libres, producto de un número por un vector y obtener de combinaciones lineales de vectores, todos de forma gráfica.
- Determinar la relación de linealidad entre dos vectores dados y calcular las coordenadas de un vector en una base cualquiera y en la base canónica.
- Obtener el producto escalar de dos vectores de forma gráfica, analítica y programática y utilizar sus propiedades para resolver distintos problemas: cálculo del módulo de un vector, del ángulo de dos vectores...
- Dividir un segmento en partes iguales.
- Manejar la herramienta *Microsoft Visual C# 2010 Express* y sus conceptos de solución, proyecto y archivo.
- Crear programas para resolver problemas y obtener resultados.

Actitudinales

- Ser participativo y colaborar en la realización de tareas grupales.
- Aprender a respetar las opiniones de los demás en un ambiente de trabajo colaborativo.
- Fomentar la actitud investigadora e imaginativa para diseñar algoritmos que resuelvan los problemas planteados.

Actividades y temporalización[12]

El desarrollo de la unidad se plantea en cuatro fases: inicial, de desarrollo, de síntesis y evaluación, repartidas en 6 sesiones. A modo de ejemplo se expone en profundidad cómo se podría desarrollar una sesión de la Unidad Didáctica:

SESIÓN	TIEMPO	CONCEPTOS	DESARROLLO
1	15 min	El conjunto R × R	Introducción al plano El primer programa
1	45 min	El punto	La clase Punto

Tabla 2. Esquema de la sesión 1

El conjunto R×R

Esta primera parte de la primera sesión corresponde a la **fase inicial**, en la que se introduce el tema de la unidad: el conjunto R × R y sus elementos el punto y el vector. Al mismo tiempo, se creará el primer programa.

- **Introducción del profesor**

En un plano R × R (en el que las abscisas y ordenadas son miembros del conjunto de los números reales, R) podemos definir dos puntos A y B como los de la figura 4, y obtener por ejemplo la distancia entre ambos.

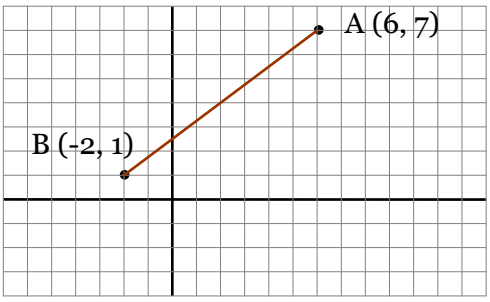

Figura 4. Puntos en el plano

La distancia será, aplicando el teorema de Pitágoras:

$$d = \sqrt{(6-(-2))^2 + (7-1)^2} = \sqrt{64+36} = \sqrt{100} = 10$$

[12] Todas las figuras y programas de este apartado son de elaboración propia.

✦ Actividad 1

A continuación se realizará el programa 1 para realizar este mismo ejercicio (y verificando que se obtiene la misma solución). Dependiendo del tiempo y los equipos disponibles, los alumnos se agruparán, preferiblemente por parejas, e introducirán el siguiente programa, que el profesor habrá escrito en la pizarra o les habrá entregado en papel.

```
using System;

namespace ProgramaGeom1
{
    class Programa1
    {
        static void Main(string[] args)
        {
            double[] A = new double[2];
            double[] B = new double[2];
            double dx, dy, d;

            A[0] = 6;
            A[1] = 7;
            B[0] = -2;
            B[1] = 1;

            Console.WriteLine("El punto A es: (" + A[0] + ", " + A[1] + ")");
            Console.WriteLine("El punto B es: (" + B[0] + ", " + B[1] + ")");

            dx = B[0] - A[0];
            dy = B[1] - A[1];

            d = Math.Sqrt(dx * dx + dy * dy);

            Console.WriteLine("La distancia entre A y B es " + d);
        }
    }
}
```

✦ Actividad 2

Los alumnos, con la agrupación ya establecida, deberán modificar el programa anterior para definir los puntos C (-3, -2) y D (4, 2), y obtener las distancias entre B y C y entre C y D.

El objetivo de esta actividad es múltiple: por un lado se trata de que los alumnos practiquen las tareas de programación; por otro, siguiendo la teoría APOS mencionada en el capítulo de desarrollo, se pretende que programen acciones sobre objetos concretos (en este caso, obtener la distancia entre puntos específicos) de forma que las interioricen y de esta forma prepararles para en una fase posterior encapsularlas en procedimientos y objetos.

El Punto

Como continuación de la actividad 2, el profesor puede suscitar una reflexión y debate sobre cómo han definido los nuevos puntos y cómo han obtenido las nuevas distancias. Haciéndoles ver que han tenido que repetir las mismas acciones (las mismas sentencias) varias veces, se les indica lo beneficioso que sería agrupar esas acciones en un único procedimiento. La solución será crear la clase Punto y el método Distancia (el objeto Punto y el procedimiento Distancia en la terminología de la teoría APOS).

- **Actividad 3**

En esta actividad, se plantea la creación de la clase Punto. Se plantea como una actividad a realizar con lápiz y papel por fases, con el siguiente planteamiento: Cada fase incorporará un elemento nuevo a la clase Punto (propuesto por el profesor) que intentará resolver cada alumno por separado. Al final de cada fase, un alumno (voluntario o designado por el profesor) escribirá su aportación en la pizarra, de forma que en conjunto se decida si es correcta o no.

Las diferentes fases podrían ser:

- Definir los datos de la clase (las coordenadas x e y del punto). Esta fracción de código podría ser la siguiente:

```
public class Punto
{
    public double x { get; set; }
    public double y { get; set; }
}
```

- Definir sus constructores (la manera en que se crea un objeto Punto a partir de sus coordenadas). La fracción de código, a añadir dentro de la clase Punto, podría ser la siguiente:

```
//Constructores
public Punto()
{
    x = y = 0;
}
public Punto(double x, double y)
{
    this.x = x;
    this.y = y;
}
public Punto(Punto pto)
{
    x = pto.x;
    y = pto.y;
}
```

Se guiará a los alumnos para que creen tres constructores distintos: uno por defecto (es decir, sin parámetros) que creara un punto de coordenadas (0, 0), otro en el que se darán las coordenadas x e y del punto y un tercero que creará un punto copiando el contenido de otro punto.

- Definir las operaciones de igualdad y desigualdad (es decir, comparar si dos objetos Punto son iguales o no). El código a añadir dentro de la clase Punto, podría ser:

```
//Operadores sobrecargados: igualdad y desigualdad
public static bool operator ==(Punto a, Punto b)
{
    return Math.Abs(a.x - b.x) < 1e-10 &&
           Math.Abs(a.y - b.y) < 1e-10;
}
public static bool operator !=(Punto a, Punto b)
{
    return !(a==b);
}
```

De esta forma, dados dos puntos A y B, para comprobar si son iguales basta utilizar la operación:

```
if (A == B) ...
```

- Definir un método para que un punto muestre sus coordenadas en pantalla. El código de este método, podría ser el siguiente:

```
//Descripción del punto
public void escribe()
{
    Console.WriteLine("(" + x + ", " + y + ")");
}
```

- Definir un método que calcule la distancia entre dos puntos. Al definir este método, es cuando entra en juego la interiorización realizada en la actividad 1, creando el procedimiento siguiente:

```
//Distancia entre dos puntos
public static double distancia(Punto a, Punto b)
{
    double dx = b.x - a.x;
    double dy = b.y - a.y;
    return Math.Sqrt(dx * dx + dy * dy);
}
```

Al final de la actividad, la definición de esta clase sería similar a la siguiente:

```
using System;

namespace Geom1
{
    public class Punto
    {
```

```
        public double x { get; set; }
        public double y { get; set; }

        //Constructores
        public Punto()
        {
            x = y = 0;
        }
        public Punto(double x, double y)
        {
            this.x = x;
            this.y = y;
        }
        public Punto(Punto pto)
        {
            x = pto.x;
            y = pto.y;
        }

        //Descripción del punto
        public void escribe()
        {
            Console.WriteLine("(" + x + ", " + y + ")");
        }

        //Operadores sobrecargados: igualdad y desigualdad
        public static bool operator ==(Punto a, Punto b)
        {
            return Math.Abs(a.x - b.x) < 1e-10 && Math.Abs(a.y - b.y) < 1e-10;
        }
        public static bool operator !=(Punto a, Punto b)
        {
            return !(a==b);
        }

        //Distancia entre dos puntos
        public static double distancia(Punto a, Punto b)
        {
            double dx = b.x - a.x;
            double dy = b.y - a.y;
            return Math.Sqrt(dx * dx + dy * dy);
        }

        //Funciones auxiliares
        public override bool Equals(object o)
        {
            Punto p = o as Punto;
            return this == p;
        }
        public override int GetHashCode()
        {
            return (int)(x + y);
        }

    }
}
```

✦ Ejercicio 1

Como ejercicio para casa, los alumnos deberán reescribir el programa 1 de la actividad 1 de forma que utilicen la clase Punto. Para la realización de esta tarea no

es imprescindible el uso del ordenador: es perfectamente realizable con lápiz y papel. Desde el punto de vista de la teoría APOS, la finalidad de esta tarea es fijar los conocimientos adquiridos y crear un esquema mental que englobe tanto el objeto punto (tanto a nivel de concepto matemático como a nivel del lenguaje de programación) como las acciones que se pueden realizar sobre él.

Este ejercicio se corrige al principio de la siguiente sesión, como una tarea grupal, corrigiendo entre todos las dificultades que los alumnos hayan podido tener en su realización. Una posible solución de este ejercicio sería:

```
using System;
using Geom1;

namespace ProgramaGeom1
{
    class Programa2
    {
        static void Main(string[] args)
        {
            Punto A = new Punto(6, 7);
            Punto B = new Punto(-2, 1);

            Console.Write("El punto A es: "); A.escribe();
            Console.Write("El punto B es: "); B.escribe();

            Console.WriteLine("La distancia entre A y B es "
                 + Punto.distancia(A, B));
        }
    }
}
```

La salida de este programa es:

```
El punto A es: (6, 7)
El punto B es: (-2, 1)
La distancia entre A y B es 10
```

Criterios de evaluación

- Distinguir si dos elementos de R × R son iguales y utilizar su relación con los puntos del plano.
- Determinar el módulo, dirección y sentido de un vector libre, su equivalencia o no con otro dado y calcular sus componentes.
- Utilizar el concepto de vector deslizante y vector fijo.
- Sumar vectores libres, multiplicarlos por un número real y obtener combinaciones lineales de vectores, todo ello de forma gráfica.
- Determinar la relación de linealidad entre dos vectores dados.

- Obtener las coordenadas de un vector en una base cualquiera y en la canónica.
- Hallar el producto escalar de dos vectores de forma gráfica y analítica y utilizar sus propiedades para resolver distintos problemas.
- Calcular el módulo de un vector y el ángulo de dos vectores.
- Dividir un segmento en partes iguales.
- Crear clases y utilizarlas en programas para superar los criterios de evaluación anteriores.
- Valorar la actitud participativa en el grupo y el trabajo personal.
- Comprobar la capacidad del alumno en proponer alternativas a los problemas y tareas planteadas.

5 Conclusiones

En primer lugar, utilizando la técnica DAFO (o SWOT en inglés), se pueden resumir en el siguiente cuadro las ventajas y desventajas sobre el objetivo general del presente trabajo.

DEBILIDADES	FORTALEZAS
• Brecha digital del profesorado: necesidad de formación • Exige interrelación y coordinación entre profesores de matemáticas e informática • Exige cambios legislativos • No se ha testado	• Estrategia multidisciplinar • Antecedentes • No implica una implantación 1 : 1. Es más barato que otras herramientas TIC
AMENAZAS	**OPORTUNIDADES**
• Falta de financiación pública en un entorno de crisis económica • Brecha digital del entorno familiar • Escepticismo sobre la utilidad de la informática en educación	• Empresas (Google, Microsoft ...) y organismos (Unión Europea) que pueden aportar financiación y apoyo • Se puede aprovechar el actual proceso de reforma educativa

Tabla 3. DAFO del presente trabajo

En el análisis de los datos de esta tabla, que se realiza a continuación, se puede comprobar que gran parte de las debilidades y amenazas se compensan con las fortalezas y oportunidades señaladas en la misma.

✦ **Brecha digital del profesorado: necesidad de formación**

El desarrollo de una propuesta de enseñanza de matemáticas a través de los lenguajes de programación exige, lógicamente, un conocimiento por parte del profesor de estos lenguajes de programación. Aunque esto no suponga un problema para los titulados en los últimos años (los lenguajes de programación entran dentro del currículo de casi cualquier estudio universitario de ciencias), sí puede resultar una barrera infranqueable para muchos profesores que carecen de este conocimiento, de

ahí el nombre de 'brecha digital'. Incluso para los que posean este conocimiento, es necesario re-elaborar los contenidos y metodologías para adaptarlos al enfoque propuesto, por lo que podemos hablar que siempre será necesaria una mayor formación del profesorado de matemáticas.

Parte de esta debilidad puede solventarse con la fortaleza del *trabajo multidisciplinar*, que se tratará más adelante. Una adopción gradual de la propuesta contenida en este trabajo también puede ayudar a reducir esta brecha.

- **Exige interrelación y coordinación entre profesores de matemáticas e informática**

Para que una propuesta de enseñanza de las matemáticas a través del aprendizaje de un lenguaje de programación sea efectiva, sobre todo en cuanto a tiempo de desarrollo, debería englobarse dentro de una estrategia más amplia, que implique que, cuando los alumnos lleguen al bachillerato tengan algún conocimiento básico de programación. Sería por tanto necesario que la asignatura de matemáticas y la de informática estuvieran coordinadas, de forma que, como indica Yábar (1995) haya una relación de mutuo apoyo.

Parte de esta debilidad también puede solventarse con la fortaleza del *trabajo multidisciplinar*, que se tratará más adelante. En todo caso, si pretendemos que los alumnos perciban la relación entre las matemáticas y los lenguajes de programación, sería deseable que también percibieran esa relación entre los diferentes departamentos implicados.

- **Exige cambios legislativos**

Si se apuesta por introducir los lenguajes de programación (las ciencias de la computación) como un recurso para la enseñanza de las matemáticas (aunque también sería útil en el campo de la física, por ejemplo), sería importante que formara también parte del currículo, al menos en la ESO y el Bachillerato. Para ello sería bueno que se reflejara en los contenidos mínimos de dichas etapas, como una materia transversal. Es cierto que podría encajarse dentro del actual marco de las asignaturas optativas, pero esto no dejaría de ser algo forzado.

Aunque esto pueda entenderse como una debilidad de la propuesta de este trabajo, también es una oportunidad para abordar el retraso español en la competencia matemática respecto de la media de la OCDE reflejado en el informe PISA

2009[13]. Precisamente las matemáticas suelen percibirse como una de las asignaturas que más fracaso escolar provocan, por lo que merece la pena considerar cualquier innovación en esta asignatura que pudiera reducir este problema.

- **No se ha testado**

La propuesta propugnada por este trabajo no se ha llevado a cabo todavía. Por tanto, se carece de un análisis realista sobre sus posibles beneficios en la enseñanza-aprendizaje de la geometría. Será por tanto una de las futuras líneas de investigación que deberían surgir como consecuencia de este trabajo.

Sin embargo, gran parte de esta debilidad puede subsanarse en la fortaleza de los antecedentes similares (que se analizará más adelante) y en la base teórica que sustenta este trabajo, mostrada en el capítulo de desarrollo.

- **Estrategia multidisciplinar**

Tal como defienden algunos teóricos de la escuela, la educación es una tarea de equipo: deberíamos huir de la división en asignaturas o departamentos estancos que no se relacionan entre sí. Desde ese punto de vista, es un valor positivo una propuesta como la que analiza este trabajo, que aprovecha y provoca una relación multidisciplinar (en este caso, matemáticas y ciencias de la computación o tecnología) de la que puede salir beneficiado el conjunto y, por ende, el alumno.

Además, es un hecho que tanto las matemáticas como los lenguajes de programación son materias transversales, que sirven de herramienta y soporte a gran parte de las ciencias. Si esa estrategia multidisciplinar ya es real en la vida cotidiana y laboral, más motivo para que también exista a nivel organizativo de los centros de enseñanza.

- **Antecedentes**

Hay algunos antecedentes y propuestas similares a las que propone este trabajo (particularmente las de Dubinsky de Da Rosa) y otras iniciativas de las que es posible aprovechar su experiencia (por ejemplo la iniciativa ECIT·EMAT de México) que permiten vislumbrar las posibilidades y beneficios de una enseñanza-aprendizaje de la geometría a través del lenguaje de programación C#.

Es cierto que muchas de las iniciativas de utilizar los lenguajes de programación en la educación (tanto como materia curricular en sí misma como siendo un

[13] OCDE (2009). *PISA 2009. Programa para la Evaluación Internacional de los Alumnos*. Informe Español. P. 147.

recurso educativo en otras materias) proceden de los albores de las ciencias de la computación en los años 60 y no han tenido la continuidad y desarrollo que sus iniciadores presagiaron. Como ya se ha indicado en la introducción de este trabajo, Da Rosa (2012) comenta como una de las razones la confusión entre alfabetización digital e informática. Es cierto: los avances tecnológicos han permitido el desarrollo de aplicaciones informáticas de carácter educativo muy sofisticadas y visualmente muy atractivas (GeoGebra, por ejemplo) que han propiciado el abandono de técnicas más 'espartanas' como los lenguajes de programación. Pero los lenguajes de programación permiten abordar técnicas (como las indicadas en la propuesta de este trabajo: la elaboración del pensamiento matemático y computacional) que no permiten esas aplicaciones más elaboradas.

- **No implica una implantación 1 : 1. Es más barato que otras herramientas TIC**

Desde la perspectiva de que es muy importante maximizar los recursos (sobre todo tecnológicos) con los que se cuenta, esta propuesta es muy flexible y adaptable. Se basa en la idea de Stalvey (2012) de que las ciencias de la computación no estudian el hardware, sino el pensamiento computacional y, para pensar, basta papel y lápiz. Además, como se ha indicado en el apartado 'Aprendizaje colaborativo', se consiguen buenos resultados con estrategias colaborativas en las que el ordenador es un recurso compartido.

En este sentido, es importante la adaptación que se haga de una propuesta de este tipo en función de las posibilidades y características del entorno en el que se implemente. La calidad de esta adaptación puede suponer la diferencia entre unos resultados positivos o negativos. La fase de análisis tiene por tanto capital importancia, para poder corregir los problemas detectados.

- **Falta de financiación pública en un entorno de crisis económica**

Todo proyecto de innovación, sea del ámbito que sea, e independientemente de la situación económica, debe contemplar sus necesidades financieras. Estas necesidades no sólo deben contemplar las necesidades materiales (ordenadores e instalaciones de banda ancha, por ejemplo) sino también los servicios (mantenimiento por ejemplo) y los recursos humanos (formación y profesorado especializado, por ejemplo). En situaciones de crisis económica como la actual, estas necesidades de financiación pueden significar la viabilidad o no de una propuesta, independientemente de sus valores educativos.

Parte de esta amenaza puede soslayarse con la oportunidad definida por la financiación privada y parte por la menor necesidad de financiación necesaria frente a otras propuestas que necesiten un ordenador por alumno.

- **Brecha digital del entorno familiar**

En muchos entornos, los alumnos no cuentan en sus hogares con ordenador o con Internet o, simplemente, no tienen a quién preguntar sus dudas porque nadie de su familia tiene conocimientos informáticos y menos aún sobre lenguajes de programación.

Para limitar esta amenaza, en esas circunstancias, se debe evitar que las tareas para casa requieran ordenador. Los trabajos en grupo también son una posibilidad, o incluso, plantear que parte de las tareas a realizar fuera de clase puedan realizarse en la sala de ordenadores fuera del horario normal de clases.

- **Escepticismo sobre la utilidad de la informática en educación**

No hay unanimidad en la comunidad educativa sobre si existe una transferencia real de conocimientos entre las ciencias de la computación y las matemáticas o no. Yábar (1995) indica que hay estudios en ambos sentidos, pero que se puede decir que esta transferencia (este aprendizaje conjunto) es mayor cuando se enseña de forma explícita e intencionada para lograr esta transferencia y, además, esta transferencia es cercana (es decir, cuando el conocimiento adquirido en una de las disciplinas es equivalente a un conocimiento de la otra). En todo caso, Yábar (1995) arguye que es muy difícil cuantificar esta transferencia, pero que para que dicha transferencia se produzca es imprescindible un trabajo continuo y por un tiempo suficientemente largo.

En este sentido es interesante la respuesta que Dubinsky y Noss (1996) dio a Koblitz (1996), publicándose ambos escritos en el mismo número de la revista *Mathematical Intelligencer*. Koblitz, apoyándose entre otros autores en investigaciones de Pea y Kurlans[14], es bastante escéptico sobre el uso los ordenadores en general y los lenguajes de programación en particular en el ámbito de la enseñanza, con argumentos como que la inclusión de ordenadores en las escueles suponen un derroche de dinero (sobre todo para países subdesarrollados, para los que postula que la tecnología de bajo coste es mejor), que los ordenadores son poco pedagógicos, o qué se puede esperar de una herramienta (la informática) que es incapaz de hacer tra-

[14] Pea, R., Kurland, M. (1984). On the cognitive effects of learning computer programming, *New Ideas in Psychology*, 2, pp. 137-168

ducciones lingüísticas. También indica que Pea et al. (1984) ha 'demostrado' que un grupo de alumnos a los que se les enseñó Logo no presentaban mejores resultados que a los que no se les enseñó. A todos estos argumentos, Dubinsky et al. (1996) responden, entre otras cosas que las autoridades educativas de cada país son las que deben decidir dónde invertir recursos, o que los ordenadores no son los que deben ser pedagógicos, sino las aplicaciones y sobre todo el uso que se haga de ellas. Respecto a que los ordenadores son incapaces de realizar traducciones indican que es como pedir a un piano que cante (aquí se diría que es como pedir peras al olmo). Salvando el hecho que desde que se escribieron esos artículos las traducciones automáticas han avanzado mucho, lo importante es destacar la idea de que cada herramienta sirve para lo que sirve y se trata de aprovechar sus potencialidades en la dirección perseguida. Finalmente, Dubinsky y Noss descalifican los estudios de Pea y Kurland, destacando su antigüedad (una década respecto al artículo de Koblitz), arguyendo que su metodología es inadecuada, que la información que aporta es escasa y, sobre todo, que las pruebas que se utilizaron para 'demostrar' los logros alcanzados no estaban relacionadas con las enseñanzas que se les impartió mediante la programación en Logo.

Pero esta discusión entre Koblitz por un lado y Dubinsky y Noss por otro, es ya antigua. Tal vez el acento hoy en día está en el polo opuesto: algunos pretenden dotar a las aplicaciones informáticas, incluidas las específicamente educativas, de un carácter de 'autoridad en la materia' o incluso que pueden sustituir eficazmente a un especialista o a un profesor. Este pensamiento no es único del ámbito educativo: por ejemplo, en el ámbito del cálculo de estructuras de edificación, algunos piensan que utilizando programas informáticos adecuados no es necesario saber calcular estructuras. Nada más lejos de la realidad. Un ejemplo que suele utilizar para mostrar esta falacia es que un procesador de textos no hace a una persona un buen novelista, tan solo mejora la productividad de una persona que ya es un buen novelista.

Es decir, la informática es una herramienta que *puede* ayudar a un profesor en su tarea, pero nunca debe sustituirlo.

- **Empresas (Google, Microsoft …) y organismos (Unión Europea) que pueden aportar financiación y apoyo**

Como ya se ha indicado, la iniciativa privada presenta una oportunidad de financiación, suministro de equipos y software y apoyo técnico que se debe aprovechar. Para la propuesta de este trabajo (enseñanza-aprendizaje de geometría con apoyo de la programación en C#) es particularmente interesante el programa DreamSpark de Microsoft ya comentado en los antecedentes, porque permite disponer

de forma gratuita, tanto para profesores como para alumnos, del software necesario para realizar la programación en C#.

- **Se puede aprovechar el actual proceso de reforma educativa**

Toda crisis es una época de cambio y, de acuerdo con un principio positivo de las crisis, éstas pueden ser una oportunidad de cambio y mejora. Es cierto que muchos de los males que se achacan a nuestro sistema educativo provienen de la poca estabilidad en el tiempo de los diferentes modelos y que no debería producirse una reforma educativa cada vez que haya un cambio político. En todo caso, deberíamos acostumbrarnos a pensar a largo plazo y con criterios adecuados.

- **Los autores referenciados**

La mayoría de los autores referenciados son docentes de matemáticas y/o ciencias de la computación en el ámbito universitario (Cirigliano, G.A., Da Rosa, S., Dubinsky, E., Koblitz, N., McDonald, Noss, R. ó Yábar Madinaveitia, J.M.), aunque también centran su atención en la docencia preuniversitaria. Si bien la mayoría de ellos hace referencia a autores de otras ramas de la educación (psicólogos o pedagogos como Pea, R., Piaget, J. o Kurland, M.), en general consideran que la enseñanza – aprendizaje de las matemáticas necesitan metodologías específicas, siendo a veces muy críticos con las investigaciones realizadas por autores 'desconocedores' de las matemáticas (por ejemplo, Dubinsky, E., Noss, R. (1996)).

6 Referencias bibliográficas

Carrasco, J. B., Javaloyes Soto, J. J., Calderero Hernández, J.F. (2008). *Cómo personalizar la educación. Una solución de futuro*. Madrid: Narcea.

Colera, J., Oliveira, M.J., García, R., Santaella, E., (2008). *Bachillerato 1 – Matemáticas I*. Madrid: Anaya

Da Rosa, S.R., Cirigliano, G.A. (2001). *Formación en Matemática Discreta usando Lenguajes de Programación*. Montevideo: Universidad de la República.

Da Rosa, S.R. (2012). *Matemática y Programación*. Montevideo: Universidad de la República. Extraído el 17 de junio de 2012 de http://www.fing.edu.uy/~darosa/MatProg.pdf

DECRETO 23/2007, de 10 de mayo, del Consejo de Gobierno, por el que se establece para la Comunidad de Madrid el currículo de la Educación Secundaria Obligatoria. B.O.C.M. núm. 126 de 29 de mayo de 2007.

DECRETO 67/2008, de 19 de junio, del Consejo de Gobierno, por el que se establece para la Comunidad de Madrid el currículo del Bachillerato. B.O.C.M. núm. 152, de 27 de junio de 2008.

Dubinsky, E. (1995). ISETL: A Programming Language for Learning Mathenatics. *Communications on Pure and Applied Mathematics*. Volume 48 (9) pp 1027-1051. West Lafayette: Purdue University

Dubinsky, E., McDonald, M. (2001). APOS: A Constructivist Theory of Learning in Undergraduate Mathematics Education Research. *The teaching and Learning of Mathematics at University Level: An ICMI Study*, Kluwer Academic Publishers, pp. 273-280. Extraído el 17 de junio de 2012 de http://www.math.kent.edu/~edd/ICMIPaper.pdf

Dubinsky, E., Noss, R. (1996) Some Kinds of Computers for Some Kinds of Math-Learning. *Mathematical Intelligencer*, Vol. 18, 1, pp. 17-20. Extraído el 17 de junio de 2012 de http://www.math.kent.edu/~edd/Output.pdf

ECIT·EMAT, (1997-2012). *ECIT·EMAT Enseñanza de la Ciencia y las Matemáticas con Tecnología*. México: Secretaría de Educación Pública – Educación Secundaria. Extraído el 17 de junio de 2012 de http://www.efit-emat.dgme.sep.gob.mx/

Google, (2010-2012). *Computer Science for High School*. Google Educational Group. Extraído el 17 de junio de 2012 de http://cs4hs.com/

Koblitz, N. (1996). The Case Against Computers in K-13 Math Education (Kindergarten through Calculus). *The Mathematical Intelligencer*, Vol. 18, No. 1, pp. 9-16. Extraído el 17 de junio de 2012 de http://www.math.washington.edu/~koblitz/mi.html

Microsoft, (2012). *Microsoft DreamSpark*. Microsoft Corporation. Extraído el 17 de junio de 2012 de https://www.dreamspark.com/default.aspx#

ORDEN 1275/2010, de 8 de marzo, por la que se implanta el proyecto de institutos de innovación tecnológica en la Comunidad de Madrid. B.O.C.M. núm. 65, de 18 de marzo de 2010.

R. D. 1631/2006, de 29 de diciembre, por el que se establecen las enseñanzas mínimas correspondientes a la Educación Secundaria Obligatoria. BOE núm. 5, de 5 de enero de 2007.

R. D. 1467/2007, de 2 de noviembre, por el que se establece la estructura del bachillerato y se fijan sus enseñanzas mínimas. BOE núm. 266, de 6 de noviembre de 2007.

Resolución de 3 de agosto de 2009, de la Secretaría General Técnica, por la que se publica el Acuerdo del Consejo de Ministros de 31 de julio de 2009, por el que se formalizan los criterios de distribución, así como la distribución resultante, para el año 2009, de los créditos presupuestarios para la aplicación del Programa Escuela 2.0, aprobados por la Conferencia Sectorial de Educación. BOE núm. 188, de 5 de agosto de 2009.

Sacristán Rock, A.I. (2005). *Programación computacional para matemáticas de secundaria*. México: Secretaría de Educación Pública.

Santillana, (2012). *Programación de Aula. Matemáticas. 1º de Bachillerato (CCNN)*. Santillana Servicios Educativos. Extraído el 17 de junio de 2012 de http://www.deciencias.net/ambito/disenoud/proyectos/santillana_aula/mates1bach(ccnn).doc

Schildt, H. (2010). *C# 4.0: The Complete Reference*. New York: The McGraw-Hill Companies.

Stalvey, R.H. (2012). *CS Unplugged*. College of Charleston. Extraído el 17 de junio de 2012 de https://docs.google.com/document/d/1IL1L3BYxb_WqAD64L49PU7VAFdoDiTwdVPTlBkDmQfY/edit?hl=en&pli=1#

Tziritas, M. (2011). *APOS Theory as a Framework to Study the Conceptual Stages of Related Rates Problems*. Quebec: Concordia University.

Ursini Legovich, S., Orendain Tremear, M. (2000). *Geometría Dinámica*. México: SEP-ILCE.

Valiente, O. (2010). 1-1 in Education: Current Practice, International Comparative Research Evidence and Policy Implications, *OECD Education Working Papers, No. 44*, OECD Publishing. http://dx.doi.org/10.1787/5kmjzwfl9vr2-en

Yábar Madinaveitia, J.M. (1995). Informática y matemáticas. ¿Quién apoya a quién? *Uno – Revista de Didáctica de las Matemáticas*. (Núm.006)