CODDII
CONFERENCIA DE DIRECTORES Y DECANOS DE INGENIERÍA INFORMÁTICA

SCIE
Sociedad Científica Informática de España

INFORME CODDII/SCIE

SOBRE

FORMACIÓN DEL PROFESORADO Y DIDÁCTICA DE LA INFORMÁTICA EN ETAPAS PREUNIVERSITARIAS

–

ENERO 2023

Autores:

Velázquez Iturbide, J. Ángel – Universidad Rey Juan Carlos (coordinador)

Llorens Largo, Faraón – Universidad de Alicante

López Álvarez, David – Universitat Politècnica de Catalunya

Marqués Andrés, Mercedes – Universitat Jaume I

RESUMEN EJECUTIVO

La sociedad de la información requiere nuevas habilidades y conocimientos. Saber desenvolverse en un entorno informatizado es hoy tan importante como saber calcular, leer o escribir y, aunque está totalmente asumido en el discurso político y social, es un tema descuidado en la educación reglada hasta el punto de que el siglo XXI ha visto resurgir un nuevo analfabetismo, el digital. La competencia digital no se adquiere simplemente por haber crecido en un mundo digital, sino que es necesario un esfuerzo para introducir esta competencia en la escuela. Sin embargo, la competencia digital no es suficiente para la sociedad futura, ya que hacen falta además conocimientos de informática que incluyan las bases conceptuales y metodológicas de la misma y habilidades de resolución de problemas con los computadores. Cualquier profesional del futuro debe dominar tanto la competencia digital como la informática, ya que interaccionará con los sistemas informáticos y deberá imaginar nuevas maneras de hacer las cosas en su profesión, en lo que se denomina transformación digital.

La informática es una disciplina que recibe la influencia de otras, pero ha evolucionado hasta ser una disciplina en sí misma. Hay ciertos términos que resultan muy atractivos, como inteligencia artificial, *big data* o robótica, siendo todos ellos términos que caen bajo el paraguas de la informática. Por ello reclamamos una formación en informática, adaptable a la evolución de esta y no simplemente guiada por términos atractivos que pueden cambiar rápidamente.

Los distintos programas de grado relacionados con la informática se inspiran en los *Computing Curricula* desarrollados por las dos asociaciones mundiales más importantes en este ámbito, la *Association for Computing Machinery* (ACM) y la *Institute of Electrical and Electronics Engineers Computer Society* (IEEE-CS). Esto ha dado una uniformidad internacional a esta disciplina y la utilización de un vocabulario compartido, que no existe a nivel preuniversitario. La Ley Orgánica 3/2020 (LOMLOE) ha establecido un renovado ordenamiento legal que conlleva la implantación de una nueva definición del currículo y sus elementos básicos, y una redistribución de las competencias entre gobierno central y comunidades autónomas en esta materia. En la Educación Primaria, los contenidos relacionados con la Informática aparecen en las enseñanzas mínimas en forma de desarrollo de la competencia digital, como apartado específico, pero también intercalados en las áreas curriculares clásicas, como el área de conocimiento del medio natural, social y cultural y el área de matemáticas, fundamentalmente. Las enseñanzas mínimas para la Educación Secundaria Obligatoria incluyen dos materias con contenidos de informática: Tecnología y Digitalización (de 1º a 3er curso) y Digitalización (materia de opción de 4º). Además, se establece que, en el conjunto de los tres primeros cursos, deberá cursarse alguna materia optativa, cuya oferta debe incluir una materia para el desarrollo de la competencia digital. Llama la atención el hecho de que prácticamente no se usa el término informática en la oferta de asignaturas, ni en las obligatorias, ni en las optativas, ni en las propuestas por el Estado, ni en las de las distintas comunidades autónomas.

Como destacan distintos informes internacionales, un aspecto crucial para el éxito de la implantación de una materia en un sistema educativo es la preparación de su profesorado. Esto aún es más relevante en nuevas materias, como es el caso de la informática. El estudio realizado en Europa por *Informatics for All* señala que es necesario que el profesorado de informática de la enseñanza obligatoria reciba formación en este campo, además de poder contar con apoyo pedagógico y metodológico sobre la enseñanza de esta disciplina.

Hay tres tipos de conocimientos relacionados con la educación de cualquier disciplina, y en concreto de la informática. El primero es el conocimiento de la propia materia, que es necesario pero no suficiente para estar en disposición de enseñar dicha materia. Un segundo conocimiento se refiere a las tecnologías que pueden usarse para la enseñanza de la materia, tanto tecnologías digitales como de otros tipos. El tercer conocimiento se refiere a las dificultades inherentes al aprendizaje de la informática y las mejores estrategias pedagógicas para promover un aprendizaje motivado y profundo. Estos conocimientos pueden formularse por separado, pero su verdadero valor para la labor docente reside en su uso combinado: el profesorado que logra situarse en esta intersección es capaz de desarrollar una acción educativa más eficaz.

La enseñanza de la informática debe sustentarse en los marcos teóricos generales de educación. Uno de estos marcos, de especial interés para la enseñanza de la informática, es el constructivismo, entendido como que cada uno debe construir su conocimiento durante su aprendizaje creando los esquemas mentales adecuados, de forma que no es suficiente con que el profesorado transmita sus conocimientos en clase para que sus estudiantes los aprendan. Lo que nos lleva a proponer para la enseñanza de la informática la utilización de métodos activos, como pueden ser el aprendizaje basado en problemas, el aprendizaje basado en proyectos, el aprendizaje colaborativo y el aprendizaje por descubrimiento. Además, esta construcción del conocimiento es un proceso incremental, que se realiza de forma gradual. Se construye sobre nuestros conocimientos previos, si existen, de forma que la detección por parte del profesorado de los conocimientos previos o preconcepciones ayuda a construir el nuevo conocimiento. Es habitual en aprendizajes realizados en entornos no formales que se produzcan malas concepciones que pueden confundir. Detectar estas malas concepciones sobre la informática puede ayudarle a planificar su enseñanza para eliminarlas o reducirlas.

Está en manos de las administraciones educativas velar por que se disponga de buenos profesores de informática. El acceso a los cuerpos de docentes se realiza a través de procesos selectivos que evalúan la idoneidad de las personas aspirantes para el ejercicio de la docencia. El sistema de ingreso en la función pública docente es el de concurso-oposición, convocado por las respectivas administraciones educativas. Y como el mismo Ministerio de Educación y FP reconoce en su propuesta de reforma para la mejora de la profesión docente, los temarios tienen una antigüedad excesiva. Es necesaria su actualización para incorporar los avances científicos y los nuevos enfoques adoptados en cada una de las ramas del conocimiento. Esto último es especialmente grave en informática por la rápida evolución de la propia disciplina, ya que los temarios de 1996 no incluyen los avances destacados en esta materia de las últimas décadas.

Todo ello nos conduce a la propuesta de las siguientes acciones para la mejora de la implantación de la informática en las etapas educativas preuniversitarias:

1. Diferenciar entre competencia digital e informática, aspectos ambos de enorme interés y que se complementan, siendo necesarios en la actualidad ciudadanos y profesionales que sepan desenvolverse en un entorno informatizado, pero que también tengan un conocimiento de las bases de la informática y hayan desarrollado habilidades de resolución de problemas con la misma.
2. Reivindicar el uso del término Informática, evitando caer en la tentación de utilizar términos atractivos que pueden pasar rápidamente de moda en el cambiante mundo tecnológico.
3. Enseñar informática desde edades tempranas en la educación básica y obligatoria, adaptada a cada edad, como forma de garantizar una educación informática para todos.
4. Avanzar hacia una informática más inclusiva, justa y social, promoviendo iniciativas que despierten vocaciones tecnológicas con especial énfasis en mujeres, reduciendo los estereotipos existentes.

5. Formar al profesorado en informática en los tres tipos de conocimientos implicados en la acción educativa: conocimiento del contenido (qué es la informática y sus principales campos), conocimiento tecnológico (cómo usar la informática en su propia docencia) y conocimiento pedagógico (cómo enseñar informática), así como sus intersecciones.
6. Promover el uso de métodos activos de aprendizaje para la enseñanza de la informática, como el aprendizaje basado en problemas, el aprendizaje basado en proyectos, el aprendizaje colaborativo y el aprendizaje por descubrimiento, ya que son las mejores estrategias de aprendizaje para crear los esquemas mentales adecuados en informática.
7. Incorporar la didáctica de la informática en los Grados en Educación Infantil y Primaria, al igual que ocurre con la didáctica de otras materias.
8. Promover la incorporación de asignaturas optativas de enseñanza de la informática en los Grados en Ingeniería en Informática, para así fomentar vocaciones docentes y dar a conocer y formar en este perfil profesional a sus titulados.
9. Actualizar los temarios de las oposiciones a los cuerpos de profesores de Educación Secundaria para incorporar los avances científicos y los nuevos enfoques surgidos en las últimas décadas, muchos de ellos relacionados con la evolución de la informática.

ÍNDICE

1. LA EDUCACIÓN PREUNIVERSITARIA DE LA INFORMÁTICA EN ESPAÑA

En este apartado realizamos una breve aclaración de qué es la Informática, sobre todo en relación con otros términos usados con frecuencia, así como del tratamiento dado a la misma en la LOMLOE.

a. Qué es Informática

La informática (cuyo término en inglés es *computing* en el entorno anglosajón e *informatics* en el entorno europeo) es una disciplina que bebe de muchos campos, entre ellos la matemática, la ingeniería electrónica o las ciencias experimentales, pero teniendo influencias de todas estas disciplinas no forma parte de ninguna, sino que es una disciplina en sí misma que lleva años evolucionando y que trata problemas más allá de los propios de dichas disciplinas.

Puede encontrarse una definición más detallada de la informática y cómo ha ido evolucionando en las últimas décadas en las sucesivas recomendaciones curriculares *Computing Curricula* desarrolladas por las dos asociaciones internacionales más importantes de informática, la *Association for Computing Machinery* (ACM) y la *Institute of Electrical and Electronics Engineers Computer Society* (IEEE-CS). Dicha serie se remonta a los *Computing Curricula 1991* (Tucker, 1991) y su previo *informe Denning* (Denning et al., 1989). Estas recomendaciones curriculares se toman como referencia en muchas partes del mundo para la actualización de los planes de estudios en ingeniería informática.

La última recomendación curricular desarrollada ha sido los *Computing Curricula 2020* - CC2020 (ACM-IEEE, 2020). De acuerdo con los CC2020, la informática la forman las actividades que requieren, se benefician o están relacionadas con la creación y uso de computadores. Esto incluye una gran variedad de temas, basados en el diseño y construcción de hardware y software para un amplio rango de propósitos, como procesar, estructurar y gestionar diversos tipos de información, resolver problemas o demostrar que son irresolubles, construir sistemas computacionales que actúen de manera inteligente, crear sistemas de comunicación, o recolectar, procesar y explotar información para cualquier propósito particular.

Los CC2020 definen los campos de especialización en informática, aunque indica que estos campos no están limitados y que previsiblemente irán creciendo con el tiempo:

- Ciberseguridad
- Ciencia de la Computación
- Ciencia de Datos
- Ingeniería de Computadores
- Ingeniería del Software
- Inteligencia Artificial.
- Tecnologías de la Información
- Sistemas de Información

En este informe buscamos aclarar qué es la informática y qué forma parte de la misma, ya que actualmente, quizá porque hay muchos términos que están de moda o resultan atractivos, muchas ramas de ingeniería y ciencias están reclamando como propios algunos conceptos y procesos que, en realidad, son propios de la informática. Así, en algunos foros se reclama que el estudiantado preuniversitario debería aprender ciertos temas como “inteligencia artificial”, “robótica”, “internet de

las cosas", "*machine learning*", "*cloud computing*", "*deep learning*", "realidad virtual", "computación cuántica", "*big data*", "industria 4.0", "5G", "*blockchain*", "criptomonedas". Sin embargo, no hemos de olvidar que todos y cada uno de estos términos caen bajo el paraguas de "informática" y por ello reclamamos una formación *en informática*, adaptable a la rápida evolución de esta y no guiada por términos atractivos que pueden pasar rápidamente de moda en el cambiante mundo tecnológico.

Hace tiempo que la informática dejó de ser el futuro para ser el presente. La sociedad del conocimiento requiere de nuevas habilidades. Hoy es tan importante saber desenvolverse en un entorno informatizado como saber leer y escribir, y sin embargo está descuidado en la educación, al punto que el siglo XXI está viendo nacer un nuevo fenómeno: el analfabetismo digital.

En primer lugar, se necesitan ciertos conocimientos prácticos (como por ejemplo manejarse con un lector de correo electrónico, un navegador, una hoja de cálculo o un editor de textos), que suelen denominarse "competencia digital" (*digital literacy*). Ya en su informe "*Europe cannot afford to miss the boat*" de 2013 (https://www.informaticsforall.org/the-informatics-boat-reports/), las asociaciones *Informatics Europe* y *ACM Europe* indicaban la imperiosa necesidad de incluirla en la educación desde edad temprana.

La competencia digital se va alcanzando poco a poco a nivel europeo. En su informe sobre el índice de economía y sociedad digital, la Comisión Europea (European Commission, 2021) indica que sólo el 58% de la población europea tiene las competencias digitales básicas. Sin embargo, por franjas de edad, las personas denominadas "jóvenes adultas" (16-24 años) alcanzan un 80% de población con competencias digitales básicas. Esto no se debe a que hayan crecido en un mundo digital, sino al esfuerzo por ir introduciendo estas competencias en la escuela, como indica el *International Computer and Information Literacy Study* (https://www.iea.nl/studies/iea/icils/2018).

Sin embargo, la competencia digital no es suficiente para la sociedad futura: hace falta un conocimiento de la informática que incluya dos cosas. Por un lado, un conocimiento de las bases de la informática (programación, hardware, redes, bases de datos, etc.) de forma que todos comprendan cómo funciona el "mundo virtual". Dicho conocimiento es imprescindible porque no solamente usarán los ordenadores los profesionales de la informática, sino también cualquier otro tipo de profesional. Por otro lado, la resolución de problemas mediante la informática ayuda a la familiarización y al desarrollo del pensamiento informático.

Las sociedades española y europea necesitan más personas especializadas en informática. En 2020, el 55% de las empresas que quisieron contratar especialistas en informática tuvieron problemas para cubrir esas vacantes (European Commission, 2021), con mayores problemas en temas como ciberseguridad o análisis de datos. Especialmente significativo es que sólo el 19% de las personas especialistas en informática son mujeres, sin que se produzca una mejora de estos números en los últimos años, cuando el objetivo comunitario es que en 2030 haya en Europa 20 millones de especialistas con paridad de género (European Commission, 2022).

Despertar vocaciones en el entorno tecnológico es importante, pero no es la razón para incluir los conocimientos de informática en la formación preuniversitaria: cualquier profesional del futuro debe dominar no sólo la competencia digital sino también las bases de la informática. Una parte importante de la tecnología emergente (inteligencia artificial, internet de las cosas, robótica, *big data*, entre otras) es básicamente informática, pero orientada a resolver problemas reales. Aunque hagan falta personas expertas en informática, hoy en día casi cualquier profesión requiere dominar lo que puede ofrecer la

informática a esa profesión. Por ello profesionales en medicina, arquitectura, filología, derecho, biología, ingeniería, etc. deben dominar las bases de la informática, para *imaginar* nuevas maneras de hacer las cosas en su profesión. Porque no se trata de hacer lo de siempre con un ordenador (eso sería digitalizar las cosas) sino de repensar lo que hacemos para aprovechar lo que la informática puede aportar, en lo que se denomina *transformación digital*. Un ejemplo de la necesidad de estos conocimientos de informática en personas no especialistas en informática es la colaboración entre el Instituto Tecnológico de Massachusetts (MIT) y las fuerzas aéreas de los EEUU (USAF) para realizar un curso de inteligencia artificial (IA) para la totalidad de miembros de la USAF. Como profesionales, trabajarán y alimentarán de datos las diversas IA que cada vez tienen un peso mayor en el día a día de las fuerzas de defensa, por lo que deben conocer cómo funcionan para mejorar su interacción (Salazar-Gómez et al., 2022).

b. La Informática en la LOMLOE

La Ley Orgánica 3/2020, de 29 de diciembre (LOMLOE), por la que se modifica la Ley Orgánica 2/2006, de 3 de mayo, de Educación (LOE), ha renovado el ordenamiento legal que conlleva, entre otras importantes modificaciones, la implantación de una nueva definición del currículo y sus elementos básicos y una redistribución de las competencias educativas entre Gobierno y comunidades autónomas. Se han dictado las normativas que establecen las enseñanzas mínimas de cada etapa (las enseñanzas mínimas cuentan con el 50% de los horarios escolares en las comunidades autónomas con lengua cooficial y el 60% en aquellas que no la tengan):

- Real Decreto 157/2022, de 1 de marzo, por el que se establece la ordenación y las enseñanzas mínimas de la Educación Primaria.
- Real Decreto 217/2022, de 29 de marzo, por el que se establece la ordenación y las enseñanzas mínimas de la Educación Secundaria Obligatoria.
- Real Decreto 243/2022, de 5 de abril, por el que se establecen la ordenación y las enseñanzas mínimas del Bachillerato.

En la Educación Primaria, son escasos los contenidos relacionados con la Informática que aparecen en las enseñanzas mínimas:

- La Competencia Digital, que implica el uso seguro, saludable, sostenible, crítico y responsable de las tecnologías digitales para el aprendizaje, para el trabajo y para la participación en la sociedad, así como la interacción con estas. Incluye la alfabetización en información y datos, la comunicación y la colaboración, la educación mediática, la creación de contenidos digitales (incluida la programación), la seguridad (incluido el bienestar digital y las competencias relacionadas con la ciberseguridad), asuntos relacionados con la ciudadanía digital, la privacidad, la propiedad intelectual, la resolución de problemas y el pensamiento computacional y crítico.
- En el área de Conocimiento del Medio Natural, Social y Cultural se incluye el bloque de *Tecnología y digitalización* que se orienta, por un lado, a la aplicación de las estrategias propias del desarrollo de proyectos de diseño y del "pensamiento computacional", para la creación de productos de forma cooperativa, que resuelvan y den solución a problemas o necesidades concretas. Por otra parte, este bloque busca también el aprendizaje, por parte del alumnado, del manejo básico de una variedad de herramientas y recursos digitales como medio para satisfacer sus necesidades de aprendizaje, de buscar y comprender información, de reelaborar y crear contenido, de comunicarse de forma efectiva y de desenvolverse en un ambiente digital de forma responsable y segura.

- En el área de Matemáticas se incluye también el “pensamiento computacional” mediante una competencia específica: “Utilizar el pensamiento computacional, organizando datos, descomponiendo en partes, reconociendo patrones, generalizando e interpretando, modificando y creando algoritmos de forma guiada, para modelizar y automatizar situaciones de la vida cotidiana.”

A continuación, se enumeran las materias con contenidos de Informática que se fijan en las enseñanzas mínimas para la Educación Secundaria Obligatoria:

- Tecnología y Digitalización (de 1º a 3º curso). Los saberes básicos de esta materia se organizan en cinco bloques: Proceso de resolución de problemas; Comunicación y difusión de ideas; Pensamiento computacional, programación y robótica; Digitalización del entorno personal de aprendizaje; y Tecnología sostenible.
- Digitalización (materia de opción de 4º). La materia se organiza en cuatro bloques interrelacionados de saberes básicos: Dispositivos digitales, sistemas operativos y de comunicación; Digitalización del entorno personal de aprendizaje; Seguridad y bienestar digital; y Ciudadanía digital crítica.

Además, se establece que, en el conjunto de los tres primeros cursos, deberá cursarse alguna materia optativa, cuya oferta debe incluir, al menos, una segunda lengua extranjera en todos los cursos, cultura clásica y una materia para el desarrollo de la competencia digital. La oferta de estas materias optativas será realizada por las administraciones educativas y puede consultarse desde el portal del sistema educativo español del Ministerio de Educación y FP (https://educagob.educacionyfp.gob.es sección Currículo). Llama la atención el hecho de que son pocas las comunidades autónomas que usan el término "informática" en su oferta de asignaturas optativas.

Ya que el desarrollo de esta nueva ordenación ha comportado la desaparición o el cambio de denominación de algunas materias del currículo de la Educación Secundaria Obligatoria, y la incorporación de otras nuevas, se debe regular la asignación de las nuevas materias a las especialidades de los cuerpos de profesorado. La siguiente tabla muestra la asignación de materias de la Educación Secundaria Obligatoria según aparece en el borrador del Real Decreto que regulará dicha asignación (publicado en julio de 2022):

Especialidades	**Materias ESO**
Informática	Digitalización (materia de opción de 4º)
Tecnología	Digitalización (materia de opción de 4º) (*) Tecnología (materia de opción de 4º) Tecnología y Digitalización (1º - 3º) Ámbito de Ciencias Aplicadas

(*) En los centros en los que exista profesorado de la especialidad de Informática, este tendrá preferencia sobre el de la especialidad de Tecnología para impartir la materia Digitalización.

2. FORMACIÓN DEL PROFESORADO PREUNIVERSITARIO EN INFORMÁTICA

En esta sección se revisa en primer lugar la formación del profesorado que debe asumir la enseñanza de la informática en las etapas de la educación básica. Después, se revisan algunas experiencias de formación de profesorado en informática en otros países. Finalmente, se identifican los distintos saberes que debe adquirir el profesorado de cualquier materia; proporcionan una base para una adecuada formación del profesorado, especialmente si trabaja en un entorno con amplio soporte tecnológico.

a. Formación, perfiles y acceso del profesorado preuniversitario

El acceso a los cuerpos docentes para ejercer la docencia en las diferentes enseñanzas reguladas en la Ley Orgánica 2/2006, de 3 de mayo, de Educación (LOE), modificada por la Ley Orgánica 2/2006, de 3 de mayo, de Educación (LOMLOE), está regulado por el Real Decreto 276/2007, de 23 de febrero, por el que se aprueba el reglamento de ingreso, accesos y adquisición de nuevas especialidades. Quienes aspiren a participar en los procedimientos selectivos, deberán reunir los siguientes requisitos específicos:

1. Para el ingreso en el cuerpo de maestros: estar en posesión del título de Maestro o el título de Grado correspondiente.
2. Para el ingreso en el cuerpo de profesorado de Educación Secundaria:
    a. Estar en posesión de un título de Doctorado, Licenciatura, Ingeniería, Arquitectura o el título de Grado correspondiente, u otros títulos equivalentes a efectos de docencia.
    b. Estar en posesión de la formación pedagógica y didáctica a la que se refiere el artículo 100.2 de la LOE.

Las especialidades docentes se establecen según el cuerpo de pertenencia:

- Maestros (Real Decreto 1594/2011): Educación Infantil, Educación Primaria, Inglés, Francés, Alemán, Educación Física, Música, Pedagogía Terapéutica y Audición y Lenguaje, además de las propias de la lengua cooficial en aquellas Comunidades Autónomas que así lo tuvieran regulado.
- Profesorado de secundaria (Real Decreto 1834/2008): las especialidades se detallan en el Anexo I del decreto, encontrándose entre ellas la especialidad de Informática y la especialidad de Tecnología.

Para el ingreso en el cuerpo de profesores de secundaria de las especialidades de Informática y de Tecnología, podrán ser admitidos quienes, aun careciendo de la titulación exigida con carácter general, estén en posesión de alguna de las titulaciones que se relacionan a continuación (Anexo V del RD 276/2007):

Especialidad	**Titulaciones**
Informática	Diplomatura en Estadística
	Ingeniería Técnica en Informática de Gestión
	Ingeniería Técnica en Informática de Sistemas
	Ingeniería Técnica de Telecomunicación, especialidad en Telemática

Especialidad	Titulaciones
Tecnología	Ingeniería Técnica
	Arquitectura Técnica
	Diplomatura en Máquinas Navales
	Diplomatura en Navegación Marítima
	Diplomatura en Radioelectrónica Naval

En cuanto a la formación pedagógica y didáctica para el acceso al cuerpo de Educación Secundaria, se debe cursar un Máster Oficial de 60 créditos que haya sido verificado de acuerdo con lo dispuesto en la Orden ECI/3858/2007, de 27 de diciembre, por la que se establecen los requisitos de los títulos universitarios oficiales que habiliten para el ejercicio de las profesión de Profesorado de Educación Secundaria Obligatoria y Bachillerato, Formación Profesional y Enseñanzas de Idiomas (modificada por la Orden EDU/3498/2011, de 16 de diciembre).

El Máster tiene un módulo genérico, un módulo específico y el prácticum. El módulo específico y el prácticum están ligados a una especialidad concreta del cuerpo de docentes de secundaria. Si bien la orden que regula el plan de estudios del máster señala que "para el ingreso en el máster se establece como requisito de acceso la acreditación del dominio de las competencias relativas a la especialización que se desee cursar, mediante la realización de una prueba diseñada al efecto por las universidades, de la que quedarán exentos quienes estén en posesión de alguna de las titulaciones universitarias que se correspondan con la especialización elegida", la convocatoria de dicha prueba no se contempla en la mayor parte de las ocasiones, accediendo estudiantes con titulaciones notablemente alejadas de las especialidades correspondientes (Ministerio de Educación y FP, 2022).

Para el acceso a los cuerpos de docentes se realizan procesos selectivos que evalúan la idoneidad de las personas aspirantes para el ejercicio de la docencia. El sistema de ingreso en la función pública docente es el de concurso-oposición, convocado por las respectivas administraciones educativas. Además, existe una fase de prácticas, que incluye también cursos de formación y que es parte del proceso selectivo.

La fase de oposición consta de dos pruebas: una prueba que comprueba los conocimientos específicos, científicos y técnicos de la especialidad docente a la que se opta (prueba práctica y desarrollo por escrito de un tema) y otra prueba que tiene por objeto la comprobación de la aptitud pedagógica de la persona aspirante y su dominio de las técnicas necesarias para el ejercicio docente (presentación de una programación didáctica de una materia de la especialidad, y preparación y exposición oral de una unidad didáctica de dicha programación).

Para la prueba de conocimientos específicos, los temarios de las distintas especialidades están regulados en la Orden ECD/191/2012:

- Los temarios de Informática se encuentran en los Anexos I y II de la Orden de 1 de febrero de 1996, por la que se aprueban los temarios que han de regir en los procedimientos de ingreso, adquisición de nuevas especialidades y movilidad para determinadas especialidades de los Cuerpos de Profesorado de Enseñanza Secundaria y Profesorado Técnico de Formación Profesional (pág. 15 del suplemento al BOE).
- Los temarios de Tecnología se encuentran en el Anexo III de la Orden de 9 de septiembre de 1993 por la que se aprueban los temarios que han de regir en los procedimientos de ingreso, adquisición de nuevas especialidades y movilidad para determinadas especialidades de los Cuerpos de Maestros, Profesorado de Enseñanza Secundaria y Profesorado de Escuelas Oficiales de Idiomas, regulados por el Real Decreto 850/1993, de 4 de junio (pág. 27419).

Como el mismo Ministerio de Educación y FP reconoce en su propuesta de reforma para la mejora de la profesión docente (Ministerio de Educación y FP, 2022), los temarios tienen una antigüedad excesiva, es necesaria su actualización para incorporar los avances científicos y los nuevos enfoques

adoptados en cada una de las ramas del conocimiento. Muchas de las materias y asignaturas adscritas a las especialidades han ido evolucionando con el tiempo, mientras que sus temarios se han quedado estancados a mediados de los años noventa. Esto último se aprecia claramente en la especialidad de Informática, cuyos temarios de 1996 no incluyen ninguno de los avances destacados en esta materia en las últimas décadas.

Por último, cabe mencionar que la especialidad cursada en el Máster no es vinculante en el acceso a la carrera docente, aspecto que también propone reformar el Ministerio de Educación y FP (2022): "Es necesario relacionar la especialidad cursada en el Máster con el acceso a la carrera docente, estableciendo la adscripción de cada especialidad de los cuerpos docentes de enseñanza secundaria y de formación profesional a una especialidad del Máster. En el caso de los funcionarios interinos y las titulaciones declaradas equivalentes a efectos de docencia para el ingreso en determinados cuerpos se puede identificar la especialidad del Máster, además de o en lugar de la titulación universitaria. En el caso del acceso a la función pública se podría tener en cuenta en la fase de concurso."

b. Experiencias en otros países de formación del profesorado en informática

Heintz, Mannila & Färnqvist (2016) llevaron a cabo una revisión de las maneras en que diversos países han introducido la informática en las etapas de Educación Primaria y Secundaria, y cómo se prepara a su profesorado.

En Inglaterra, *Computing at School* (CAS, http://computingatschool.org.uk) es una comunidad que promueve la informática en la escuela. La comunidad permite que los usuarios se registren y participen en los foros de discusión, y proporciona recursos didácticos. Además, se han creado los CAS *hubs*, que son comunidades locales de profesores que desean compartir ideas sobre la enseñanza de la informática. El éxito del proyecto CAS se debe en parte a la red de centros, ya que las reuniones locales reducen el aislamiento del profesorado y aumenta su compromiso. Los recursos didácticos incluyen planes de clase y directrices para diferentes niveles, empezando por *Barefoot Computing* (https://www.barefootcomputing.org/) para Educación Primaria y *Computing at School* (https://www.computingatschool.org.uk/) que proporciona apoyo tanto en Educación Primaria como Secundaria. Además, parte del profesorado también ha desarrollado y compartido material didáctico (véase, por ejemplo, http://code-it.co.uk).

En Australia, se han puesto en marcha iniciativas como un MOOC de desarrollo profesional y una revisión sistemática de los recursos para el plan de estudios de informática. Asimismo, en Nueva Zelanda se desarrolló un curso de postgrado a distancia que permitía obtener una cualificación formal en la enseñanza de la informática. Para proporcionar un material didáctico adecuado, se desarrolló la *CS Field Guide* (https://www.csfieldguide.org.nz/es/). Se trata de un sitio interactivo desarrollado para proporcionar información al nivel requerido por los nuevos estándares de la informática, incluyendo material para el profesorado.

En Estados Unidos, el *K-12 Computer Science Framework* (https://k12cs.org/) contempla el desarrollo profesional del profesorado. Los conceptos y las prácticas incluidos en el marco ayudan a los diseñadores de programas y a las personas noveles a organizar y comprender los diversos conocimientos del campo de la informática. El documento se redactó intencionadamente teniendo en cuenta que lo utilizarían tanto el profesorado familiarizado con la informática como las personas nuevas en este campo.

La *Computer Science Teachers Association* (CSTA) ha abordado la cuestión de la preparación del profesorado de informática en los EEUU a través de un grupo especial (https://csteachers.org/page/standards-for-cs-teachers). El informe recomienda formar comunidades de profesorado de informática que puedan transmitir la naturaleza de la disciplina y que incluso puedan ser tutelados por profesorado universitario.

Centrándonos en Europa, en Noruega se preparó un MOOC y también se ha creado una asociación

llamada *Lær Kidsa Koding* para proporcionar material didáctico, así como contribuir al desarrollo profesional (http://kidsakoder.no/). Para la preparación del profesorado en Polonia, se cuenta con un estándar de educación en informática que es similar a los estándares de la ISTE (http://www.iste.org). También existe un procedimiento de acreditación que evalúa la preparación de cada profesional para la enseñanza de la informática.

En Estonia, el HITSA (*Information Technology Foundation for Education*, https://www.hitsa.ee/) desarrolla materiales de aprendizaje y ofrece cursos de desarrollo profesional para profesorado. En Finlandia, el Consejo Nacional de Educación y el Ministerio de Educación y Cultura financian proyectos y programas de desarrollo profesional para proporcionar al profesorado recursos y maneras de integrar la programación en la enseñanza. Además de cursos presenciales especializados, en 2015/2016 se iniciaron cursos en línea de formación de profesorado sobre programación, tanto en finés como en sueco, atrayendo a un gran número de participantes. Además de las iniciativas apoyadas por el Estado, también hay actividades que se ofrecen por parte de entidades privadas, universidades y organizaciones.

En Europa está teniendo un papel activo la coalición *Informatics for All*. Se creó en 2018 a iniciativa de ACM *Europe Council*, CEPIS *Education Committee*, e *Informatics Europe*; IFIP se unió en 2020. El objetivo de la coalición es que la informática se considere una disciplina fundamental que curse todo el alumnado en la escuela. La informática debería ser reconocida como una disciplina esencial que desempeña un papel importante en la educación del siglo XXI, tan importante como las matemáticas, las ciencias y los idiomas. Su informe *The Informatics Reference Framework for School* (https://www.informaticsforall.org/the-informatics-reference-framework-for-school-release-february-2022/) contiene recomendaciones concretas sobre cómo integrar la informática en el currículo escolar.

Asimismo, su informe *Designing and Implementing a Concrete Informatics Curriculum for School* (https://www.informaticsforall.org/designing-and-implementing-a-concrete-informatics-curriculum-for-school/) señala que es necesario que el profesorado de informática de la enseñanza obligatoria reciba formación en este campo, además de poder contar con apoyo pedagógico y metodológico sobre la enseñanza de esta disciplina. En el informe se presentan algunos ejemplos para desarrollar el currículum de Informática. En cuanto a los objetivos generales, que deben considerarse antes de establecer los contenidos, y que son una descripción a un nivel superior de las metas a alcanzar, se indica que se trata de dar una visión global de la informática y del uso de la informática, así como los enfoques pedagógicos a la hora de establecer las actividades de aprendizaje: actividades prácticas con y sin ordenadores, trabajo en equipo, creatividad y abstracción. Las cuestiones relativas a la igualdad, diversidad e inclusión son cruciales también al establecer estos objetivos generales.

c. Conocimiento de contenido, pedagógico y tecnológico

Las universidades ofrecen grados y másteres para formar al futuro profesorado de forma genérica pero también de forma específica para distintas materias: idiomas, matemáticas, educación física, etc. Se asume que no basta con saber de una disciplina, sino que hay que saber enseñarla. Esta sencilla idea se ha ido concretando en varios modelos, sobre todo los modelos PCK y TPACK.

El modelo *Pedagogical and Content Knowledge*, PCK (Shulman, 1986), distinguía entre conocimiento de contenido y conocimiento pedagógico. Posteriormente, se ha desarrollado el modelo *Technological, Pedagogical and Content Knowledge*, TPACK (Koehler & Mishra, 2009), que presentamos a continuación, ya que realiza un desglose más detallado de los distintos conocimientos implicados en la acción educativa. Desgloses parecidos se habían realizado previamente en el ámbito de la enseñanza de la informática, aunque sin llegar a formular un marco teórico (Gal-Ezer & Harel, 1998).

El modelo TPACK distingue entre conocimiento del contenido, conocimiento pedagógico y conocimiento tecnológico. Estos tipos de conocimiento pueden formularse por separado, pero también intersectan, como muestra la siguiente figura. Veamos estos tipos de conocimiento, ilustrados con algún ejemplo extraído de la enseñanza de la programación.

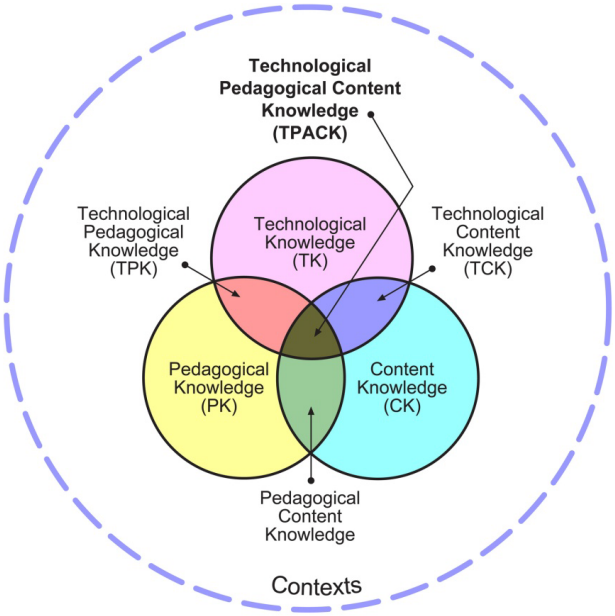

Fig. 1. El marco TPACK y sus componentes. Fuente: Koehler and Mishra (2009).

Veamos estos tipos de conocimiento, ilustrados con algún ejemplo extraído de la enseñanza de la programación.

El *conocimiento de contenido* es el conocimiento que tiene el profesorado sobre la materia a enseñar. Este conocimiento debe ser coherente con el conocimiento aceptado por los expertos en la materia. Obviamente, este conocimiento varía con la etapa educativa y la especialidad de los estudios. Incluye hechos, conceptos, teorías y marcos, formas de determinar la evidencia y la verdad, y métodos de trabajo. Por ejemplo, el aprendizaje de la programación incluye conceptos y lenguajes de programación, así como elementos de estilo de programación o técnicas básicas de pruebas, entre otros.

El *conocimiento pedagógico* es el conocimiento del profesorado sobre los procesos y métodos educativos, tanto del aprendizaje por parte del alumnado como de la enseñanza por parte del profesorado. Incluye cuestiones como una comprensión del proceso de aprendizaje en los estudiantes, planificación docente, gestión de la clase, métodos de enseñanza y evaluación. Por ejemplo, el profesorado debe conocer métodos de aprendizaje activo, como el aprendizaje basado en problemas o el aprendizaje colaborativo, y las condiciones en que deben aplicarse para que el aprendiz tenga un papel activo cognitivamente y no solo en su comportamiento.

El *conocimiento pedagógico de contenido* permite al profesorado usar el conocimiento pedagógico de forma eficaz para el aprendizaje de una materia concreta. Por ejemplo, pueden usarse la programación en pareja para el aprendizaje colaborativo o los problemas de Parsons para el estudio o evaluación individual de la programación. También permite al profesorado reinterpretar la materia para su enseñanza eficaz en contextos concretos, encontrando formas alternativas de presentarla, y adaptando los materiales educativos para afrontar las preconcepciones y malas concepciones de los alumnos. Esta capacidad es importante para adaptarse a alumnos con edades o sexos distintos o de perfiles cognitivos o motivacionales distintos. Por ejemplo, una profesora de informática puede

utilizar un lenguaje de programación “tradicional” (como Java o C) para enseñar programación a estudiantes de informática, pero puede preferir el uso de Processing o Python si sus estudiantes son, respectivamente, de bellas artes o de estadística. Asimismo, el conocimiento de las dificultades que suele encontrar cada aprendiz en el aprendizaje de la operación de asignación o de la recursividad le permite anticiparlas, adaptando sus contenidos y actividades educativas correspondientes.

El *conocimiento tecnológico* es el tipo de conocimiento que cambia más rápidamente, en especial en el uso de las tecnologías digitales. Este conocimiento es equiparable a la competencia digital en el uso de medios digitales, pero también se utilizan otras tecnologías. Algunas tecnologías de uso diario tienen siglos de existencia, como el uso del papel para escribir, mientras que otras son más recientes, como el uso de tarjetas programadas para acceder a locales.

El *conocimiento tecnológico de contenido* es el conocimiento del profesorado sobre la forma en que la tecnología ha afectado a una materia, sobre todo a la representación y manipulación de la información. Para la enseñanza de la informática, suelen utilizarse tecnologías hardware y software. Por ejemplo, una forma de presentar los programas y los algoritmos de forma más tangible y, en principio, más comprensible es mediante su visualización, dando lugar a sistemas de visualización de programas o de animación de algoritmos.

El *conocimiento tecnológico pedagógico* es el conocimiento de formas eficaces de utilizar una tecnología para la enseñanza de cierta materia, así como sus limitaciones. En concreto, este conocimiento es clave para un uso eficaz de distintas aplicaciones desarrolladas para educación, como simuladores, juegos serios, etc. Por ejemplo, las animaciones de algoritmos sólo son eficaces si cada estudiante es cognitivamente activo, lo cual puede conseguirse de varias formas, como permitiéndole que controle el avance y retroceso de una animación, permitiendo que pueda introducir sus propios datos de entrada u obligándole a responder a preguntas durante la animación.

El *conocimiento tecnológico pedagógico de contenido* se sitúa en la intersección de los conocimientos de contenido, pedagógico y tecnológico. El profesorado que logra situarse en esta intersección es capaz de desarrollar una acción educativa eficaz. No existe una forma única de combinar los tres aspectos en una asignatura, sino que cada persona puede hacerlo de forma distinta. El uso combinado de los tres aspectos es dinámico y cualquier cambio en uno de estos aspectos obliga a un reequilibrio de los otros dos. Suele ser más frecuente que acontezca un cambio tecnológico, pero también son frecuentes otros cambios. Por ejemplo, el avance tecnológico informático puede hacer que un lenguaje o entorno de programación se quede obsoleto, en cuyo caso podría llegar a obligar a un cambio de lenguaje o entorno de programación. Asimismo, una nueva versión puede proporcionar nuevas posibilidades que afecten a los ejercicios o las prácticas de la asignatura. Asimismo, si se decide un cambio de contenido, como el lenguaje de programación, obligará a una reestructuración del aspecto tecnológico y probablemente del pedagógico.

Este informe pretende proporcionar una introducción a los sectores de intersección de la Fig. 1 referidos a la enseñanza de la informática. Por tanto, se presentan de forma algo detallada el conocimiento pedagógico de contenido y de forma menos detallada el conocimiento tecnológico de contenido y la intersección entre ambas. El objetivo del informe es ilustrar dichos tipos de conocimiento en su aplicación a la educación en informática, sin centrarse en ninguna etapa educativa concreta. Obviamente, su aplicación en una etapa concreta exige un análisis detallado.

3. ELEMENTOS DESTACADOS DE LA DIDÁCTICA DE LA INFORMÁTICA

En esta sección se presentan varios elementos destacados de la didáctica de la informática. La presentación se centra en la programación, debido a que es el área informática más ampliamente investigada en el campo educativo, pero no hay que olvidar que la informática contiene otras áreas.

En primer lugar, se presenta el constructivismo, que presenta una teoría ampliamente aceptada sobre cómo se produce el aprendizaje. Es la base para la comprensión de las dificultades de los estudiantes y da mayor sentido a algunos de los avances científicos en enseñanza de la informática realizados en las décadas pasadas.

Los dos apartados siguientes presentan dos aspectos importantes relacionados con los estudiantes, la motivación y las malas concepciones, que deben ser tenidas en cuenta por el profesorado en su planificación docente. A continuación se presentan varios aspectos destacados para la enseñanza de la educación en informática: instrucción de la programación, aprendizaje en equipo, ejercicios específicos de la materia y la evaluación mediante prácticas. La lista no es exhaustiva, pero el objetivo es ilustrar formas concretas de didáctica específica de la informática. La sección se cierra con una breve presentación de destacados tipos de sistemas de apoyo al aprendizaje de la programación.

a. Constructivismo y métodos activos de aprendizaje

El constructivismo es una teoría del aprendizaje que declara que el conocimiento *no se transmite*, sino que *se construye* (Ben-Ari, 2001). Por tanto, no basta con que la profesora transmita sus conocimientos en clase para que sus estudiantes los aprendan, sino que cada estudiante debe construir su conocimiento durante su aprendizaje en forma de esquemas mentales adecuados. Una consecuencia inmediata del constructivismo es que los estudiantes tienen un papel activo en la construcción de sus nuevos conocimientos. Se han propuesto diversos *métodos activos* de aprendizaje para favorecer la construcción del conocimiento, siendo conocidos el aprendizaje basado en problemas, el aprendizaje basado en proyectos, el aprendizaje colaborativo y el aprendizaje por descubrimiento.

Otras consecuencias prácticas del constructivismo como marco teórico no son tan evidentes. Por un lado, no basta con utilizar los métodos activos de aprendizaje de cualquier forma, sino que deben usarse de forma adecuada para que sean eficaces (Prince, 2004). Por ejemplo, el alumno debe ser cognitivamente activo para que se produzca construcción del conocimiento, y no sólo activo en su comportamiento (Mayer, 2004). En consecuencia, los métodos de aprendizaje por descubrimiento puro son poco eficaces, siendo preferible proporcionar una guía mínima al alumno. En general, no puede decirse que haya métodos mejores o peores. Todos los métodos de aprendizaje presentan ventajas e inconvenientes, incluso la clase magistral (Laurillard, 2012). Lo deseable es que la profesora conozca los métodos de aprendizaje y los utilice adecuadamente, como se verá en el siguiente subapartado.

La construcción del conocimiento es un proceso incremental. Siempre construimos el nuevo conocimiento sobre nuestros conocimientos anteriores. Al estudiar algo completamente nuevo, es probable que no tengamos apenas conocimientos relacionados, pero confiamos en nuestras capacidades intelectuales generales. Por ejemplo, si se plantea a una persona un problema en un campo nuevo, usará métodos generales de resolución de problemas, aunque un experto podría utilizar algún método específico, más eficiente. Durante el proceso de aprendizaje, cada aprendiz va construyendo esquemas mentales de conocimiento y de resolución de problemas que son específicos

de la materia (Sweller, Ayres & Kalyuga, 2011). La disponibilidad de esquemas específicos de resolución de problemas en una materia es lo que diferencia a expertos y aprendices. El paso de uno a otro es un proceso largo, que dura años (Dreyfus & Dreyfus, 1986). Se ha conjeturado que el plazo es de diez años (Winslow, 1996), así que incluso una persona recién graduada todavía está lejos de alcanzar el grado de experta en su campo.

La construcción del conocimiento no es un proceso fácil. Por un lado, el conocimiento siempre tiene un gran componente memorístico, pero también debe tener algún grado de comprensión para que no sea un conocimiento inerte. Además, el conocimiento se construye sobre los conocimientos previos, si existen. La existencia de conocimientos previos o preconcepciones puede ayudar a construir el nuevo conocimiento, pero otras veces puede confundir, sobre todo cuando los nuevos conocimientos tienen el mismo nombre que otros ya existentes pero distintos. Asimismo, la construcción del nuevo conocimiento se realiza de forma gradual, de forma que entre el desconocimiento inicial hasta el conocimiento final se pasa por etapas en las que el conocimiento en construcción puede ser incompleto, ambiguo e incluso incoherente (Norman, 1983). El conocimiento previo o la detección por parte del profesorado de las malas concepciones desarrolladas por el alumnado puede ayudarle a planificar su enseñanza para eliminarlas o reducirlas.

b. Motivación

La informática suele ser percibida como difícil y abstracta, además de estar lastrada por una serie de estereotipos que deben superarse con información. En diferentes estudios realizados preguntando a jóvenes sobre su visión de la informática, las palabras que más aparecen a la hora de definir a un profesional de la informática son “hombre”, “inteligente”, “tímido”, “empollón (*nerd*)” “friki (*geek*)” y “asocial”. Respecto al trabajo de las personas que se dedican profesionalmente a la informática, las palabras que aparecen son “aburrida”, “solitaria”, “cerebral”, “dedicada” y “obsesiva” (Alshahrani, Ross & Wood 2018; Taylor-Smith, Smith & Smith, 2019; Wong 2017).

Buena parte de los estereotipos están asociados al género. Sinclair y Kavala (2015) mostraron que los dos problemas principales de sesgo de género en los estudios de informática son: los propios estereotipos de género (la informática es para niños) y las expectativas culturales (que la familia, o el profesorado desaliente a las niñas a estudiar esta carrera). Trabajar en la eliminación de estos sesgos de género ha demostrado tener una influencia muy positiva en la matriculación de mujeres en estudios STEM (Charlesworth & Banaji, 2019).

A pesar de estos estereotipos, lo que más influye en la elección de la informática como futura profesión es cómo se ve cada persona a sí misma realizando estos estudios, es decir, la denominada autoeficacia (*self-efficacy*). Según Bandura, la autoeficacia la componen “...los juicios de las personas sobre sus capacidades para alcanzar ciertos niveles de rendimiento” (Bandura, 1986). Como los conceptos de informática tienen fama de difíciles, existe la visión de que para poder dedicarse a la informática hace falta tener habilidades matemáticas, de programación, pensamiento lógico, organización y emprendimiento (Peters, 2019). Una persona que no se visualice a sí misma con estas capacidades difícilmente elegirá esta profesión.

El estudio de Alshahrani, Ross & Wood (2018) nos habla de que, para tener una visión positiva de la informática e influir en la autoeficacia de cada persona, es de vital importancia haber vivido una experiencia positiva en asignaturas de informática. La experiencia vivida depende básicamente del profesorado, el planteamiento de la asignatura y del resto de estudiantes.

La motivación del alumnado para cualquier materia depende de que tenga una visión de su utilidad y de que sienta atracción por la misma. Por ello, si queremos formar profesionales en informática o en cualquier campo con conocimientos de informática hay que atacar los estereotipos, trabajar en que el alumnado se vea con capacidad para resolver problemas en un entorno informático y desarrollar asignaturas bien planteadas, con profesorado formado que permita crear experiencias positivas con la informática.

Igualmente, hay que pensar en las personas que no se van a dedicar profesionalmente a la informática pero que van a vivir su día a día rodeados de la informática interactuando con ella. Estas personas necesitan una formación en las bases de la informática, pero también verse a sí mismas como diseñadores y colaboradores, no como meros usuarios. Para ello necesitan estar motivados por la informática, viéndose capaces de aprender un conocimiento básico de la misma y de interactuar con soltura y confianza con los dispositivos y programas informáticos (autoeficacia).

c. Malas concepciones

Como se comentó en el apartado dedicado al constructivismo, cada aprendiz construye internamente su conocimiento. Se dice que un aprendiz construye su propio modelo mental de la materia explicada. Un modelo mental se construye de forma incremental, según avanza el aprendizaje, por lo que es normal que sea incompleto, ambiguo e incoherente en numerosas ocasiones (Norman, 1983). En cambio, la explicación de la materia que realiza la profesora o que está escrita en un libro debe ser completa, precisa y coherente. Además, debe ser compatible con el consenso de las personas expertas en la materia (aunque pueda estar modificada para su mejor presentación, por ejemplo, de forma simplificada o expresada mediante metáforas). Dicha explicación es el modelo conceptual que se presenta a los estudiantes.

Es difícil determinar el modelo mental de cada alumno, ya que no es algo observable, siendo normalmente necesario realizar algún tipo de investigación. Afortunadamente, el número de modelos mentales que se pueden construir sobre cierta materia es limitado. Por ejemplo, se han estudiado los modelos mentales que se construyen para comprender la recursividad (Götschi, Sanders & Galpin, 2003). Aparte del modelo mental correspondiente a la ejecución real de la recursividad en los lenguajes de programación (un doble proceso de “ida y vuelta”), existen otros modelos mentales que difieren entre sí en las partes de la ejecución que no comprende bien el estudiante, por ejemplo, olvidar el proceso de vuelta o cambiar el orden de las operaciones a realizar entre los resultados de las sucesivas llamadas recursivas.

Dado que los modelos mentales no son tangibles, no se dice que un modelo mental de una persona sea correcto o no, sino viable o inviable para comprender o predecir ciertos fenómenos. De hecho, un modelo mental puede ser viable en algunos casos, pero no en otros. Por ejemplo, un aprendiz que haya construido un modelo mental de la recursividad similar a la ejecución de un bucle dispondrá de un modelo mental viable para la recursividad final (o de cola), pero no para otras formas de recursividad.

A veces no se dispone de una teorización en forma de modelos mentales de alguna parte de la materia, pero se conocen algunas dificultades del alumnado. En este caso, podemos centrarnos en la comprensión de conceptos individuales. Se dice que quien aprende construye concepciones. En caso de comprender mal dicho concepto, el aprendiz habrá construido malas concepciones. Por otro lado, recordemos que cada aprendiz construye su conocimiento sobre el conocimiento que previamente

tiene. Por tanto, la construcción de concepciones se realiza a partir de las concepciones disponibles previamente, o preconcepciones.

Se han estudiado ampliamente las malas concepciones de los alumnos de programación (Clancy, 2004; Sorva, 2012). Hay un gran abanico de malas concepciones, algunas tan generales como suponer que el ordenador conoce la intención de la persona que lo ha programado o que entiende inglés (ya que los lenguajes de programación suelen tener sus palabras clave en inglés). Veamos algunas de las razones por las que surgen (obviamente, dejando aparte la posibilidad de que la materia no se haya explicado de forma adecuada).

Una primera razón es por una transferencia inadecuada del conocimiento de otra materia a la programación. Esta causa es esperable, ya que una gran parte de los estudiantes no tienen experiencia previa en programación, por lo que intentan utilizar conocimiento previo de otras materias. La transferencia de conocimiento se realiza principalmente desde el idioma y desde las matemáticas.

La transferencia desde el idioma conserva el significado que tienen en la lengua convencional ciertos términos (normalmente, las palabras clave). Por ejemplo, la palabra clave *while* (“mientras”) sugiere un proceso continuo. En consecuencia, algunos alumnos piensan que la condición asociada a un bucle *while* se está evaluando en todo momento de la ejecución del cuerpo del bucle, cuando en realidad solamente se comprueba tras haber ejecutado el cuerpo del bucle.

La transferencia desde matemáticas tiene lugar al suponer que algunos elementos del lenguaje de programación tienen el mismo significado que en matemáticas, por tener el mismo nombre o usar la misma notación. El ejemplo más conocido es la confusión entre la igualdad matemática y la asignación en lenguajes de programación que utilizan el mismo símbolo ‘=’, creyendo leer sinsentidos matemáticos como ‘x=x+1’. Al revés, es fácil olvidar o no darse cuenta de que la igualdad matemática ‘=’ se representa en algunos lenguajes con el símbolo ‘==’.

Otras veces, el aprendiz realiza una sobregeneralización errónea a partir de unos pocos ejemplos. Por ejemplo, si todos los ejemplos usados para explicar la recursividad son recursivos finales, un aprendiz puede generalizar el comportamiento de la recursividad exclusivamente a partir de estos ejemplos, infiriendo que es similar a un bucle. En este caso, el modelo mental construido sólo contiene el proceso “de ida” de la recursividad, no el “de vuelta”, resultando inviable para comprender el comportamiento de otras formas de recursividad. Una situación similar puede darse por un mal uso o por los aspectos problemáticos de metáforas usadas para explicar conceptos. Por ejemplo, la metáfora de la variable como una caja presenta el problema de que un estudiante pueda pensar que una variable puede contener simultáneamente varios valores (tras realizar varias instrucciones de asignación).

Las mayores dificultades se dan en aquellas construcciones cuya ejecución conlleva la realización de acciones “ocultas” a quien programa, como es el caso de la recursividad, la memoria dinámica o el polimorfismo de la programación orientada a objetos. Incluso en lenguajes basados en bloques tan sencillos como ScratchJr existe dificultad para comprender el efecto de algunos bloques de control (como el bloque Parar) o de metacontrol (como variar la velocidad de movimiento de un personaje).

Las malas concepciones se dan incluso entre personas que se pueden considerar expertas en programación (Shrestha et al., 2022). Estos profesionales esperan que el comportamiento de cierta característica en el lenguaje de programación que están aprendiendo sea igual que en algún otro lenguaje conocido, por ejemplo, por tener el mismo nombre. Por ejemplo, el paso de mensajes en ScratchJr es distinto al paso de mensajes presente en lenguajes concurrentes, siendo más parecido a una llamada a procedimiento.

d. Instrucción de la programación

Dada la relativa juventud de la informática (en comparación con otras disciplinas como las matemáticas o la física), no existe una teoría claramente aceptada de cómo debe enseñarse la informática. Por tanto, presentamos en este apartado algunas cuestiones destacadas.

En primer lugar, hay que señalar que la materia más difícil al aprender informática se da en una de sus primeras asignaturas: la programación de computadores. Este hecho se ha comprobado en sucesivos estudios realizados en diversos países y universidades (Soloway, 1983; McCracken et al., 2001; Lister et al., 2004).

Puede sorprender esta situación porque uno esperaría que la mayor dificultad se daría en asignaturas más avanzadas. Numerosos factores contribuyen a esta dificultad (Gomes & Mendes, 2007), entre otros:

- Falta de conocimientos previos. Ya se ha comentado este factor en el apartado anterior como fuente de malas concepciones.
- Aprendizaje simultáneo de conceptos y habilidades de programación, de un lenguaje de programación y de la resolución algorítmica de problemas.
- Dedicación de poca atención a la dinámica de los lenguajes de programación. El comportamiento dinámico de los lenguajes de programación es una característica que no existe en los materiales escritos en otras disciplinas, por ejemplo, las fórmulas o demostraciones matemáticas.
- Problemas abstractos. En otras materias se plantean problemas con datos concretos, pero los problemas algorítmicos suelen plantearse de forma que debe encontrarse una solución general, es decir, un algoritmo.
- Problemas cuya resolución exige un alto nivel cognitivo. En un estudio que comparó las actividades de evaluación de asignaturas de programación y de redes (Oliver et al., 2004), se constató que el esfuerzo cognitivo exigido (según la taxonomía de Bloom) era superior en las asignaturas de programación, ya que se pedía crear productos originales (programas) frente a las actividades predominantes de memorización o comprensión en las asignaturas de redes.

Como resultado de estas dificultades, se han realizado y validado diversas propuestas para enseñar la programación de forma que el esfuerzo cognitivo que se exige al alumnado sea progresivo. Nos limitaremos a señalar tres aportaciones destacadas.

En primer lugar, es necesario dar mayor importancia a explicar el comportamiento dinámico del lenguaje de programación usado en la asignatura introductoria a la programación. No basta con describir el léxico, la sintaxis y las restricciones semánticas de los distintos elementos del lenguaje, ilustrados con ejemplos. Es necesario describir de forma clara su semántica dinámica, es decir, su efecto sobre una hipotética máquina abstracta o virtual (duBoulay, O'Shea & Monk, 1981; Sorva, 2013). Ni siquiera es necesario que se defina esta máquina abstracta de forma completa, pero es necesario que se ilustre de forma precisa el efecto que cada instrucción produce sobre su estado.

La necesidad de explicar el comportamiento dinámico es especialmente crítica en construcciones del lenguaje cuya ejecución produce efectos que no aparecen en el texto del programa. Por ejemplo, la recursividad debe enseñarse de forma que se entienda claramente que es un proceso bidireccional (llamada y terminación) que puede explicarse de varias maneras: mediante reescritura en lenguajes funcionales, mediante su efecto en la pila de ejecución en lenguajes imperativos, etc. Con frecuencia,

la explicación del comportamiento dinámico de un lenguaje se acompaña de representaciones gráficas, que pueden implementarse e incluso generarse automáticamente en los sistemas de visualización de programas (Sorva, Karavirta & Malmi, 2013). Por ejemplo, el comportamiento de recursividad se ilustra de forma muy clara con varias representaciones gráficas, entre ellas los árboles de recursión (Velázquez-Iturbide & Pérez-Carrasco, 2016).

Una segunda aportación consiste en proporcionar al alumnado patrones y antipatrones de código. Es decir, en lugar de dejarles sin ninguna orientación para el desarrollo de programas, los patrones (Clancy & Linn, 1999) son soluciones generales para tareas frecuentes, por ejemplo, cómo recorrer un array para realizar alguna operación con todos sus elementos. El desarrollo de cierta maestría por parte de los aprendices responde al desarrollo de modelos mentales en forma de esquemas de programación que les ayudan a resolver problemas. El objetivo, por tanto, de presentar explícitamente patrones de código es ayudar a los alumnos al desarrollo de dichos modelos mentales. También son útiles los patrones de código a evitar, normalmente por responder a un mal estilo de programación. Estos patrones se llaman antipatrones.

Una tercera aportación se refiere a las actividades instruccionales que debe realizar el alumnado según se avanza en la enseñanza de los conceptos y las construcciones del lenguaje de programación. Con la programación basada en bloques, sobre todo, se ha popularizado la metodología llamada usar-modificar-crear (*use-modify-create*, véase Lee et al., 2011). Dado un enunciado que explica lo que debe hacer un programa, se proporciona la solución a los estudiantes para que lo usen y se familiaricen con la solución. Posteriormente se les pide que cambien detalles pequeños. Finalmente, se les pide que lo amplíen o modifiquen de forma sustancial.

Yendo un paso más allá, se ha investigado la relación entre distintas formas de interactuar con el código. Existen evidencias de que el alumnado debe ser capaz, en primer lugar, de predecir el efecto de partes de un programa y de escribir el rastro ("traza") de su comportamiento. También debe ser capaz de abstraer el propósito de parte de un código. Una vez que se tiene éxito en ambas tareas, los alumnos están en mejores condiciones de desarrollar programas (Clear et al., 2011; Fowler et al., 2022). Se han propuesto y validado incluso metodologías instruccionales basadas en esta precedencia de actividades educativas (Xie et al., 2019).

e. Aprendizaje de programación en equipo

Desde Piaget y otros psicólogos de la educación, se sabe que la interacción entre iguales que aprenden en una relación más simétrica es tan importante como la relación asimétrica que se da entre profesorado y estudiantado. El aprendizaje cooperativo se basa en el trabajo de equipos reducidos de estudiantes, generalmente de composición heterogénea en rendimiento y capacidad (aunque ocasionalmente pueden ser más homogéneos), utilizando una estructura de actividad que asegure al máximo la participación equitativa (que todo el mundo tenga las mismas oportunidades de participar) y potencie al máximo la interacción simultánea (que todas las personas aprendan y que lo hagan al máximo de sus posibilidades), además de aprender a trabajar en equipo (Pujolàs & Lago, 2011).

La programación en pareja (*pair programming*), que se usa tanto en la industria como en la educación, es una actividad con estructura cooperativa. Consiste en que dos personas trabajan simultáneamente en el desarrollo de un programa (o un proyecto mayor de desarrollo de software): una persona tiene el control del teclado y el ratón (líder, conductor) y la otra colabora revisando el código y haciendo un seguimiento del trabajo que se va realizando en relación con la planificación o el diseño inicial (McDowell et al., 2006). Estos roles se intercambian de manera regular tras breves períodos de tiempo

(5, 10 o 20 minutos). Cuando se utiliza la programación en pareja, hay que tener en cuenta el contexto (tipo de tarea, relaciones sociales en el aula, etc.) para establecer diversos aspectos como la duración de los períodos de tiempo entre intercambios de rol, la formación de las parejas, la estructuración de las tareas, entre otros (The Royal Society, 2017).

Spencer Kagan (2001) diseñó cerca de 150 estructuras cooperativas a partir de las cuales se pueden crear actividades de trabajo en equipos cooperativos. Se denominan *estructuras cooperativas* porque lo que se describen son los pasos que se deben seguir para asegurar la participación equitativa y la interacción simultánea (Pujolàs & Lago, 2011). Por ejemplo, en la estructura *lápices al centro* se comienza entregando una hoja de ejercicios a cada equipo (tantos como miembros hay en el grupo). En cada ejercicio, un alumno será el líder. El primero lee el ejercicio y entre todos deciden cómo van a resolverlo (no escriben nada, solo dialogan, los lápices están en el centro). Cuando se ponen de acuerdo, cogen los lápices y cada estudiante resuelve el ejercicio en su hoja. Y así sucesivamente.

El aprendizaje basado en proyectos, el aprendizaje basado en problemas y el aprendizaje basado en la investigación también pueden organizarse con trabajo en equipo, promoviendo la autonomía y aumentando la motivación. El estudiantado es el protagonista de las tareas, tiene capacidad de elección y, además, trabaja en situaciones contextualizadas en situaciones cotidianas.

f. Ejercicios específicos

La evaluación del alumnado es una tarea fundamental en cualquier acción educativa, que debe estar alineada con los objetivos educativos y las actividades instruccionales (Anderson et al., 2001). La evaluación varía según el grado de aprendizaje esperado. Según la taxonomía revisada de Bloom (Anderson et al., 2001), hay seis grados de aprendizaje (categorías de procesos cognitivos), desde el simple recuerdo hasta la creación de productos. Existen formas universales de evaluar que pueden usarse en todos estos niveles, como son las preguntas de respuesta múltiple.

Esbozamos qué tipos de ejercicios de esta naturaleza pueden plantearse sobre programación o algoritmia en las distintas categorías de la taxonomía revisada de Bloom. Hay que tener en cuenta que la resolución de un ejercicio frecuentemente implica varios procesos cognitivos (Masapanta-Carrión & Velázquez-Iturbide, 2019):

- Categoría Recordar. Estas preguntas piden al estudiante que recuerde o reconozca algún hecho o concepto. Por ejemplo, “Describe los formatos de un literal de tipo carácter en Java” o “Identifica qué identificador de los siguientes es una palabra clave del lenguaje de programación”. En realidad, cualquier ejercicio de programación con código exige al estudiante que active este proceso cognitivo porque necesita conocer el léxico, sintaxis y semántica estática del lenguaje de programación.
- Categoría Comprender. Por ejemplo, puede pedirse que el aprendiz traduzca un trozo de código en formato de diagrama de flujo a código en algún lenguaje de programación o que identifique qué secuencia de instrucciones es equivalente a otra dada. De hecho, los ejercicios de traducción de texto a código no son muy frecuentes en programación textual, pero lo son en la programación basada en bloques.
- Categoría Aplicar. El alumno aplica de forma sistemática algún método que, según el caso, puede ser totalmente determinista o admitir grados de libertad. Por ejemplo, se le pide que evalúe una expresión con operadores infijos o que presente el rastro (traza) correspondiente a la ejecución de un trozo de código con ciertos datos de entrada. Obsérvese que estos ejercicios ni siquiera exigen una comprensión del código presentado sino simplemente la

aplicación sistemática de la semántica operacional del lenguaje de programación. Otro ejemplo es el análisis de la complejidad de cierto algoritmo (cuyo análisis no requiera creatividad). En este caso, la aplicación del método de análisis de complejidad no es automática.

- Categoría Analizar. En esta categoría se debe analizar algún producto o proceso. Por ejemplo, se pide al aprendiz determinar qué variables son accesibles desde el cuerpo de un método o determinar el orden de aplicación de los operadores infijos en una expresión.
- Categoría Evaluar. Se valoran uno o varios productos o procesos con respecto a algún criterio. Por ejemplo, puede juzgarse el estilo de programación utilizado en un programa o comparar dos algoritmos que resuelven un mismo problema.
- Categoría Crear. El ejemplo más representativo consiste en desarrollar un algoritmo que resuelva cierto problema. Admite diversas variantes, como especificar la cabecera del método principal o la técnica de diseño algorítmico de utilizar. Sin embargo, la creación puede ser un proceso cognitivo necesario no sólo para ejercicios de desarrollo de software. Por ejemplo, surge en el análisis de complejidad o en la verificación de un algoritmo si conlleva pasos no triviales, es decir, cierta creatividad.

Estos ejercicios pueden utilizarse para evaluar a los estudiantes (evaluación sumativa) pero también para que ejerciten la materia y aprendan, en cuyo caso la realimentación recibida sobre sus errores es muy importante (evaluación formativa). Obsérvese que los ejercicios de diseño de algoritmos corresponden a la categoría más alta de procesos cognitivos, Crear, por lo que es una mala práctica limitarla a este tipo de ejercicios. Un alto porcentaje del alumnado puede ser capaz de desarrollar procesos cognitivos de otras categorías sin haber llegado a la más alta. Por tanto, es preferible utilizar también ejercicios de otros tipos. La resolución semanal de ejercicios de programación sencillos es una aproximación didáctica a la programación que se ha denominado “muchos ejercicios pequeños” (*many-small exercises*) (Allen et al., 2018).

Dentro del conocimiento pedagógico de contenido del profesorado de informática, es deseable que conozca tipos de ejercicios desarrollados específicamente para la disciplina. Veamos brevemente otros ejercicios específicos de programación:

- Ejercicios predictivos (Lister et al., 2004). Dado un trozo de código y unos datos iniciales, se pide el valor de algunas variables o los mensajes impresos como resultado de su ejecución. También puede preguntarse por el resultado devuelto por una función para unos parámetros concretos o el valor de algún parámetro tras terminar la ejecución de un método.
- Problemas de Parsons (Parsons & Haden, 2006). Se describe el efecto esperado de un trozo de código y un conjunto de instrucciones, pidiendo que se ordenen de forma que produzcan dicho efecto.
- Ejercicios de programación con andamiaje. Se describe el comportamiento esperado de un programa y se da solamente parte del mismo. Este tipo de ejercicios puede concretarse de varias formas. Por ejemplo, puede darse una parte ya desarrollada (el esqueleto), de forma que cada estudiante sólo deba desarrollar la parte que falta, por ejemplo, el cuerpo de un bucle (Lister et al., 2004). Otro formato con menos andamiaje consiste en indicar que la solución debe utilizar elementos concretos del lenguaje de programación (por ejemplo, iteración o recursividad), elementos vertebradores de la solución (como la cabecera que deben tener uno o varios métodos) o la cabecera de los métodos a usar para tareas auxiliares.

Algunos ejercicios pueden diseñarse con igual facilidad para variados tipos de lenguajes de programación, por ejemplo, los ejercicios predictivos para los lenguajes imperativos textuales y algunos lenguajes de bloques (Weintrop & Wilensky, 2015). Sin embargo, su diseño puede ser a veces

más difícil o incluso imposible. Por ejemplo, el lenguaje basado en bloques ScratchJr no contiene variables, por lo cual solamente pueden diseñarse ejercicios predictivos sobre el comportamiento visible o sonoro de los personajes.

Algunos de estos tipos de ejercicios han jugado un papel destacado en la investigación en enseñanza de la informática. En concreto, puede destacarse el papel de los ejercicios de rastreo y los ejercicios predictivos. Para algunos tipos de ejercicios se han desarrollado aplicaciones educativas que les dan soporte, como ha sucedido con los ejercicios de análisis o los problemas de Parsons.

La lista anterior de tipos de ejercicios no es exhaustiva, pero contiene algunos de los más destacados. Se han propuesto varias taxonomías para clasificar los ejercicios de programación, que muestran un amplio abanico de ejercicios (Bower, 2008; Ragonis, 2012; Ruf, Berges & Hubweiser, 2015).

g. Corrección de las prácticas

En la realización de las prácticas por parte del estudiantado es necesario implementar un sistema de retroalimentación que le permita mantenerse informado sobre su progreso (o su falta de progreso) (Valero-García & Díaz de Cerio, 2005). Cada estudiante necesita saber si lo ha hecho bien o mal, y por qué, y necesita saberlo pronto para poder aprender de sus propios errores.

Si bien esta retroalimentación puede ser proporcionada por el profesorado, la carga de trabajo es elevada y es difícil conseguir la característica más importante de este tipo de evaluación (formativa): que la información llegue a los estudiantes a tiempo.

Las estrategias de autoevaluación y coevaluación pueden usarse para que cada estudiante reciba la retroalimentación con prontitud. En la autoevaluación es la propia persona quien determina en qué medida su trabajo está bien o mal y en la coevaluación cada estudiante evalúa el trabajo de uno o varios de sus compañeros. En ambos casos, cada estudiante hace la evaluación siguiendo las instrucciones que proporciona el profesor.

La autoevaluación y la coevaluación tienen otras ventajas. Ya que el estudiantado sigue las instrucciones del profesorado para llevarlas a cabo, van interiorizando los criterios de evaluación que están íntimamente ligados a los objetivos de aprendizaje. Además, el hecho de participar en la evaluación ayuda al estudiantado a desarrollar su capacidad de reflexión y, con ello, su autonomía.

La autoevaluación se utiliza cuando la solución a los ejercicios es única o admite pocas variaciones (esto sucede cuando los ejercicios trabajan los niveles más bajos de la taxonomía de Bloom: conocimiento y comprensión). El profesorado debe proporcionar la solución oficial que usará el alumnado para compararla con la suya, de manera que las diferencias sean posibles errores. Hacer la autoevaluación consistirá en identificar las diferencias, reconocer las que sean fruto de errores y justificar las que no lo sean. Además, cada estudiante debe escribir una conclusión en donde reflexione sobre los errores cometidos (y que intentará no volver a repetir).

Es importante ayudar al estudiantado a que aprenda a autoevaluarse (Gibbs & Simpson, 2005). Las primeras veces no lo harán bien, al menos no todo el mundo, por lo que es importante que se revisen los primeros informes y se les ayude en caso de que lo necesiten. También se pueden proporcionar ejemplos de buenas y malas autoevaluaciones.

Cuando el ejercicio puede tener múltiples soluciones válidas (suelen ser ejercicios del nivel de aplicación), no se puede usar una solución oficial como base de la evaluación. En este caso, se deben

hacer explícitas las características que ha de tener dicha solución. Una forma de presentar dichas características es mediante rúbricas.

Una rúbrica es una tabla de doble entrada. En la primera columna se sitúan los criterios de realización, que son los aspectos sobre los que se ha de pensar para analizar la calidad de lo que se evalúa (ejercicio, problema, etc.). Cada una de las siguientes columnas corresponderá a un nivel de calidad de cada uno de los criterios de realización.

Confeccionar una buena rúbrica no es fácil ya que debe estar expresada en términos que comprenda el estudiantado (Sanmartí & Mas, 2016). Por ejemplo, en los niveles de calidad de una característica no se puede especificar algo como "está bien" o "está muy bien". El profesorado es experto y tiene una concepción de lo que está bien y lo que está muy bien (aunque quizá no sea la misma que tenga otra persona), pero el alumnado no lo es por lo que hay que hacer explícito qué es aquello que caracteriza algo que está bien o que está muy bien.

Ya que la evaluación en base a una rúbrica es menos objetiva al haber cuestiones que pueden ser opinables, la estrategia que se adapta mejor es la de la coevaluación. La tarea de cada estudiante evaluador consiste en identificar, en la solución de la persona a evaluar, los criterios de realización y determinar el nivel de calidad en función de lo establecido en la rúbrica. El informe de evaluación se completará argumentando las decisiones tomadas en la evaluación de cada criterio.

Otro instrumento que se puede usar para la coevaluación son las listas de criterios (*checklist*), en donde se hacen explícitos los criterios de realización en un nivel alto y para diferenciar el grado de calidad se dan una serie de calificativos (por ejemplo: muy bueno, bueno, regular o deficiente; siempre, casi siempre, en pocas ocasiones, nunca).

Ya que el objetivo de la evaluación formativa es la de informar a cada estudiante sobre su progreso para que identifique lo que aún no sabe hacer bien y tenga información para poder superar sus dificultades, no se recomienda que el resultado de la autoevaluación y la coevaluación formen parte de sus notas. Es positivo dar un refuerzo por el hecho de hacerlas (y hacerlas bien) que puede reflejarse como parte de la nota, pero lo que repercutirá negativamente es el hecho de calificar los ejercicios. El error no debe penalizar cuando se está aprendiendo (Sanmartí, 2020).

h. Sistemas de apoyo al aprendizaje de la programación

Dentro del conocimiento tecnológico de contenido para programación, resumimos algunos de los tipos de sistemas software más destacados, aunque la lista no es exhaustiva (Malmi, Utting & Ko, 2019).

En primer lugar, podemos citar lenguajes y entornos de programación para educación. Algunos lenguajes y entornos de programación se desarrollan específicamente para educación (Kelleher & Pausch, 2005). Varían desde los que facilitan la edición de los programas (p.ej. mediante programación basada en bloques o la generación de mensajes de compilación más comprensibles) al desarrollo de nuevos modelos de programación (p.ej. la programación estructurada en los años 70 o la programación basada en bloques actualmente) o facilitar una mejor comprensión de la ejecución de los programas. Algunos lenguajes incluso se diseñan pensando en colectivos educativos específicos, como Processing para artistas (Reas & Fry, 2014). En el caso de los tradicionales lenguajes textuales, los entornos suelen proporcionar una interacción más sencilla con el lenguaje de programación que los entornos profesionales. Por ejemplo, BlueJ es un entorno de programación para Java (Barnes &

Kölling, 2016) que proporciona una interfaz visual e interactiva que facilita el desarrollo, prueba y depuración de programas orientados a objetos.

La mejor comprensión de la ejecución de los programas suele estar ligada a mayores facilidades para controlar el rastreo (*trace*) de la ejecución de los programas o a una presentación más concreta y comprensible. La mejora de este aspecto ha dado lugar a sistemas de visualización de programas (Sorva, Karavirta & Malmi, 2013). Estos sistemas generan representaciones gráficas de la ejecución de los programas que ayudan a superar el carácter abstracto de los programas mediante su visualización, facilitando de esta manera su comprensión. Por ejemplo, el sistema Jeliot genera visualizaciones de Java que se utilizan en Educación Secundaria (Ben-Ari et al., 2011). Otro tipo de sistemas relacionado son los sistemas de animación de algoritmos, que muestran el comportamiento de los algoritmos de forma abstracta, es decir, sin una relación directa con el código fuente. En ambos casos, se ha desarrollado una intensa investigación sobre su uso educativo, ya que se ha concluido que más importante que el formato de las propias representaciones gráficas es el uso que se haga de ellas (Hundhausen, Douglas & Stasko, 2022; Naps et al., 2002).

Otros sistemas están concebidos como ayuda al aprendizaje de conceptos concretos. Estos sistemas suelen basarse en ejercicios pequeños, que cada estudiante debe responder o resolver. Por ejemplo, existen colecciones de ejercicios cuya corrección es automática y proporcionan realimentación a cada estudiante sobre sus aciertos y errores. Un ejemplo es el popular sitio web CodingBat (https://codingbat.com/) , que contiene lecciones, vídeos y ejercicios de programación dedicados a distintos aspectos de Python y Java. Cualquier aprendiz puede resolver los problemas planteados, que son corregidos automáticamente.

Los sistemas de corrección automática son sistemas en red que permiten plantear ejercicios de programación, que el alumnado debe resolver y cuyos programas son corregidos automáticamente (Paiva, Leal & Figueira, 2022). Su uso principal es comprobar la corrección de los programas propuestos mediante pruebas con juegos de datos (creados por el propio alumno o por la profesora). Sin embargo, existen sistemas que permiten evaluar los programas con otros criterios (p.ej., eficiencia o estilo) de forma dinámica o estática. Desde un punto de vista docente, también suelen incorporar la detección de plagios (Novak, Joy & Kermek, 2019). Resultan muy útiles cuando se dispone de clases con un gran número de estudiantes, pero su usabilidad suele ser menor que otros sistemas comentados anteriormente.

4. IMPLANTACIÓN DE LA INFORMÁTICA EN ETAPAS EDUCATIVAS PREUNIVERSITARIAS

En el presente informe se ha realizado un recorrido por la naturaleza de la informática, como disciplina relativamente reciente y en constante evolución, la necesidad de disponer de profesorado preparado para impartir esta materia y el desarrollo de una didáctica específica de esta disciplina, enmarcada en los fundamentos y principios de la didáctica general. Tras la recopilación de información y la correspondiente argumentación, en este apartado vamos a resumir los aspectos más destacables y proponer 10 acciones que deberían emprenderse. Algunas son universales, mientras que otras son específicas de nuestro país.

a. Reflexiones para la mejora

En relación con la informática como disciplina y la formación universal en la misma, destacamos:

- **Uso del término Informática.** Últimamente existe una tendencia a utilizar otros términos como sustitutos de informática, no siempre coincidentes con la misma. Por ejemplo, computación o ciencia de la computación, pensamiento computacional, inteligencia artificial o digitalización. Tanto a nivel general como académico, el término informática es mucho más preciso en español, por lo que debe reivindicarse activamente su uso y recuperación.
- **Diferenciación entre competencia digital e informática.** A pesar de la clara diferencia entre ambos términos, se siguen confundiendo, a veces por desconocimiento, a veces de forma interesada.
- **Enseñanza de la informática desde edades tempranas.** La educación en informática debe aprenderse desde edades tempranas, de forma adecuadamente modulada a cada edad. Recordemos que la reivindicación del estudio de la informática es principalmente por el bien general de la sociedad y la ciudadanía, ampliando su cultura general.
- **Necesidad de especialistas en informática, con una mayor presencia de las mujeres.** Un objetivo secundario de un mayor conocimiento de la informática en las edades preuniversitarias es facilitar que se despierten vocaciones tecnológicas. Esta cuestión es especialmente crítica entre las mujeres, cuya presencia en los estudios de informática ha seguido una tendencia descendente, siendo actualmente muy baja.

En cuanto a los aspectos didácticos, tanto generales como específicos de la informática, proponemos:

- **Motivación por la informática.** La motivación por cualquier materia es un factor fundamental para su aprendizaje. El conocimiento de la informática permitiría reducir los estereotipos existentes sobre la misma. Un profesorado bien formado podría estar más motivado hacia la enseñanza de la informática y podría diseñar experiencias positivas relacionadas con la informática que reduzcan dichos estereotipos entre el alumnado y que contemplen diversos intereses y gustos.
- **Metodologías activas.** Si bien estos métodos de aprendizaje pueden aplicarse a cualquier materia, son especialmente relevantes en informática. Aunque la memorización es necesaria, debe recordarse que la informática es una disciplina básicamente orientada a la resolución de problemas.
- **Aprendizaje adecuado.** Una buena formación en informática del profesorado permitirá una mejor enseñanza de la informática, y por tanto un encauzamiento adecuado de las preconcepciones de los estudiantes y la prevención del desarrollo de malas concepciones (que dificultan su posterior desarrollo).

La formación del profesorado en informática es uno de los principales obstáculos detectados, tanto a nivel nacional como internacional, para la integración en el currículo de una materia de informática, convirtiéndose en un aspecto crucial. Por ello sugerimos:

- **Conocimiento de contenido, pedagógico y tecnológico.** La formación de profesorado en Informática requiere su formación en los tres componentes de conocimiento de contenido, conocimiento pedagógico y conocimiento tecnológico, y todas sus intersecciones.
- **Profesorado de Educación Infantil y Primaria.** Al igual que en los Grados en Educación Infantil y Primaria se introduce la didáctica de otras materias, debería integrarse una didáctica de la informática y, sobre todo, de la programación.
- **Profesorado de Educación Secundaria y Bachillerato.** Este profesorado opta a una especialidad según el grado en que se han titulado. Una acción desde el propio ámbito informático sería integrar asignaturas optativas en los Grados en Informática, para fomentar vocaciones entre este colectivo, tradicionalmente alejado de la educación. Además, el Máster Oficial de Formación del profesorado debería incluir un módulo específico de Informática, con su formación pedagógico-didáctica y prácticum asociados.
- **Actualización de temarios para las oposiciones.** Dichos temarios deberían incorporar los avances científicos y los nuevos enfoques surgidos en las últimas décadas.

b. Nueve propuestas para la mejora

La búsqueda de información, argumentación a favor y las reflexiones derivadas nos conducen a la propuesta de un decálogo de acciones para la mejora de la implantación de la informática en las etapas educativas preuniversitarias. Instamos a los creadores de opinión y a los tomadores de decisiones (directivos universitarios y autoridades educativas) a que las tengan en cuenta.

1. Diferenciar entre competencia digital e informática, aspectos ambos de enorme interés y que se complementan, siendo necesarios en la actualidad ciudadanos y profesionales que sepan desenvolverse en un entorno informatizado, pero que también tengan un conocimiento de las bases de la informática y hayan desarrollado habilidades de resolución de problemas con la misma.
2. Reivindicar el uso del término Informática, evitando caer en la tentación de utilizar términos atractivos que pueden pasar rápidamente de moda en el cambiante mundo tecnológico.
3. Enseñar informática desde edades tempranas en la educación básica y obligatoria, adaptada a cada edad, como forma de garantizar una educación informática para todos.
4. Avanzar hacia una informática más inclusiva, justa y social, promoviendo iniciativas que despierten vocaciones tecnológicas con especial énfasis en mujeres, reduciendo los estereotipos existentes.
5. Formar al profesorado en informática en los tres tipos de conocimientos implicados en la acción educativa: conocimiento del contenido (qué es la informática y sus principales campos), conocimiento tecnológico (cómo usar la informática en su propia docencia) y conocimiento pedagógico (cómo enseñar informática), así como sus intersecciones.
6. Promover el uso de métodos activos de aprendizaje para la enseñanza de la informática, como el aprendizaje basado en problemas, el aprendizaje basado en proyectos, el aprendizaje colaborativo y el aprendizaje por descubrimiento, ya que son las mejores estrategias de aprendizaje para crear los esquemas mentales adecuados en informática.
7. Incorporar la didáctica de la informática en los Grados en Educación Infantil y Primaria, al igual que ocurre con la didáctica de otras materias.

8. Promover la incorporación de asignaturas optativas de enseñanza de la informática en los Grados en Ingeniería en Informática, para así fomentar vocaciones docentes y dar a conocer y formar en este perfil profesional a sus titulados.
9. Actualizar los temarios de las oposiciones a los cuerpos de profesores de Educación Secundaria para incorporar los avances científicos y los nuevos enfoques surgidos en las últimas décadas, muchos de ellos relacionados con la evolución de la informática.

En un mundo cada vez más tecnológico y digital, el aprendizaje de la informática es un aspecto esencial para poder participar plenamente en esta sociedad y como habilidad valiosa para cualquier profesional. La formación de profesorado especializado en informática permitirá, por un lado, disponer de profesorado capacitado para enseñar esta materia de manera efectiva y, por otro, que este profesorado esté actualizado y pueda transmitir a sus estudiantes los conocimientos y habilidades necesarias para mantenerse al día y comprender el mundo en el que viven.

AGRADECIMIENTOS

Agradecemos la revisión de la primera versión del informe a: Antonio Bahamonde Rionda (Universidad de Oviedo), Sanja Dabic Radisc (Asociación de Profesores de Informática de la Comunidad Valenciana), Juan Manuel Dodero Beardo (Universidad de Cádiz), Ángel Fidalgo Blanco (Universidad Politécnica de Madrid), Antonio Hernández Fernández (Universitat Politècnica de Catalunya), Francisco Javier Soriano Camino (Universidad Politécnica de Madrid) y Francisco José Vico Vela (Universidad de Málaga).

REFERENCIAS

ACM-IEEE (2020). Computing Curricula 2020. https://doi.org/10.1145/3467967.

Allen, J.M., Vahid, F., Downey, K. & Edgcomb A. (2018) "Weekly programs in a CS1 class: Experiences with auto-graded many-small programs (MSP)". En Proceedings of 2018 ASEE Annual Conference & Exposition. https://peer.asee.org/31231.

Alshahrani A., Ross, I. & Wood, M.I. (2018). "Using social cognitive career theory to understand why students choose to study computer science". En ICER'18. https://doi.org/10.1145/3230977.3230994.

Anderson, L. W., Krathwohl, D. R., Airasian, P. W., Cruikshank, K. A., Mayer, R. E., Pintrich,P. R., Raths, R. & Wittrock, M. C. (2001) A Taxonomy for Learning, Teaching and Assessing. A Revision of Bloom's Taxonomy of Educational Objectives. Pearson Education, 2001.

Bandura, A. (1986). Social foundations of thought and action: A social cognitive theory. 1986: Prentice-Hall.

Barnes, D. J. & Kölling, M. (2016). Objects First with Java: A Practical Introduction using BlueJ, 6ª edición, Pearson, 2016.

Ben-Ari, M. (2001) "Constructivism in computer science education". Journal of Computers in Mathematics and Science Teaching, 20(1):45-73. https://doi.org/10.1145/274790.274308.

Ben-Ari, M., Bednarik, R., Ben-Bassat Levy, R., Ebel, G., Moreno, A., Myller, N. & Sutinen, E. (2011). "A decade of research and development program animation: The Jeliot experience", Journal of Visual Languages and Computing, 22;375–384, https://doi.org/10.1016/j.jvlc.2011.04.004.

Bower, W. (2008) "A taxonomy of task types in computing". En Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2008), pp. 281-285. https://doi.org/10.1145/1384271.1384346.

Charlesworth, T.E.S. & Bana-ji, M.R (2019). "Gender in Science, Technology, Engineering, and Mathematics: Issues, Causes, Solutions" J. of Neuroscience. 2019, 39 (37):7228-7243. https://doi.org/10.1523/JNEUROSCI.0475-18.2019.

Clancy, M. J. (2004) "Misconceptions and attitudes that interfere with learning to program". En Computer Science Education Research, Sally Fincher y Marian Petre (eds.), Londres: Routledge, pp. 85-100.

Clancy, M. J. & Linn, M. C. (1999) "Patterns and pedagogy". En Proceedings of the 30th SIGCSE Technical Symposium on Computer Science Education (SIGCSE 1999), pp. 37-42. https://doi.org/10.1145/384266.299673.

Clear, T., Whalley, J., Robbins, P., Philpott, A., Eckerdal, A., Laakso, M., & Lister, R. (2011). "Report on the final BRACElet workshop". Journal of Applied Computing and Information Technology, 15(1). http://www.citrenz.ac.nz/jacit/JACIT1501/2011Clear_BRACElet.html.

Denning, P. J., Comer, D. E., Gries, D., Mulder, M. C., Tucker, A. B., Turner, A. J., & Young, P. R. (1989). "Computing as a discipline", Communications of the ACM, 32(1):9-23, Ene. 1989. https://doi.org/10.1145/63238.63239.

Dreyfus, S.E. & Dreyfus H.L. (1986) Mind over Machine. Nueva York, NY: Free Press.

du Boulay, B., O'Shea, T. & Monk, J. (1981) "The black box inside the glass box: Presenting computing concepts to novices", International Journal of Man-Machine Studies, 14(3):237-249, https://doi.org/10.1016/S0020-7373(81)80056-9.

European Commission (2021). International Digital Economy and Society Index [DESI]. https://digital-strategy.ec.europa.eu/en/policies/desi Última visita: 18/7/2022.

European Commission (2022). Women in Digital Scoreboard 2021. https://digital-strategy.ec.europa.eu/en/news/women-digital-scoreboard-2021 Última visita: 18/7/2022.

Fowler, M., Smith IV, D.H., Hassan, M., Poulsen, S., West, M. & Zilles, C. (2022) "Reevaluating the relationship between explaining, tracing, and writing skills in CS1 in a replication study", Computer Science Education, 32(3):355-383, https://doi.org/10.1080/08993408.2022.2079866.

Gal-Ezer, J. & Harel, D. (1998) "What (else) should CS educators know?", Communications of the ACM, 41(9):77-84. https://doi.org/10.1145/285070.285085.

Gibbs, G., & Simpson, C. (2005). Conditions under which assessment supports students' learning. Learning and Teaching in Higher Education, (1), 3-31. https://eprints.glos.ac.uk/id/eprint/3609

Gomes, A. & Mendes, A.J. (2007) "Learning to program - problems and solutions". En International Conference on Engineering Education (ICEE 2007), http://icee2007.dei.uc.pt/proceedings/papers/411.pdf.

Götschi, T., Sanders, I. & Galpin, V. (2003) "Mental models of recursion". En Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2003), pp. 346-350 https://doi.org/10.1145/611892.612004.

Heintz, F., Mannila, L., & Färnqvist, T. (2016). "A review of models for introducing computational thinking, computer science and computing in K-12 education". En 2016 IEEE Frontiers in Education conference (FIE) (pp. 1-9). https://doi.org/10.1109/FIE.2016.7757410.

Hundhausen, C. D., Douglas, S. A. & Stasko, J. T. (2002). "A meta-study of algorithm visualization effectiveness", Journal of Visual Languages and Computing, 13(3):259-290, https://doi.org/10.1006/jvlc.2002.0237.

Kagan, S. (2001) Kagan Structures and Learning Together: What is the Difference? San Clemente, CA: Kagan Publishing. Kagan Online Magazine. www.KaganOnline.com.

Kelleher, C. & Pausch, R. (2005). "Lowering the barriers to programming: A Taxonomy of programming environments and languages for novice programmers", ACM Computing Surveys, 37(2):83-137, junio 2005, https://doi.org/10.1145/1089733.1089734.

Koehler, M.J. & Mishra, P. (2009) "What is technological pedagogical content knowledge?", Contemporary Issues in Technology and Teacher Education, 9(1):60-70 https://www.learntechlib.org/primary/p/29544/.

Laurillard, D. (2012) Teaching as a Design Science: Building Pedagogy Patterns for Learning and Technology. Londres, RU: Routledge.

Lee, I.,Marti, F.,Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J. & Werner, L. (2011) "Computational thinking for youth in practice", ACM Inroads, 2(1):32–37. https://doi.org/10.1145/1929887.1929902.

Lister, R., Adams, E.S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J.E., Sanders, K., Seppälä, O., Simon, B. & Thomas, L. (2004) "A multi-national study of reading and tracing skills in novice programmers", ACM SIGCSE Bulletin, 36(4):119-150, Dic. 2004 https://doi.org/10.1145/1041624.1041673.

Malmi, L., Utting, I. & Ko, A. J. (2019). "Tools and environments". En The Cambridge Handbook on Computing Education Research, Fincher, S. A. & Robins, A. V. (eds.), Cambridge, RU: Cambridge University Press, pp. 639-662.

Masapanta-Carrión, S. & Velázquez-Iturbide, J.A. (2019). "Evaluating instructors' classification of programming exercises using the revised Bloom's taxonomy". En: Proceedings of the 24th Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE 2019, ACM Press, 2019, págs. 541-547, https://doi.org/10.1145/3304221.3319748.

Mayer, R.E. (2004) "Should there be a three-strikes rule against pure discovery learning? The case for guided methods of instruction", American Psychologist, 59(1):14-19. https://doi.org/10.1037/0003-066X.59.1.14.

McCracken, M., Kolikant, Y.B.-D., Almstrum, V., Laxer, C., Diaz, D., Thomas, L., Guzdial, M., Utting, I., Hagan, D. & Wilusz, T. (2001) "A multinational, multi-institutional study of assessment of programming skills of first-year CS students", ACM SIGCSE Bulletin, 33(4):125-140, Dic. 2001. https://doi.org/10.1145/572133.572137.

McDowell, C., Werner, L., Bullock, H. E., & Fernald, J. (2006). "Pair programming improves student retention, confidence, and program quality". Communications of the ACM, 49(8), 90-95. https://doi.org/10.1145/1145287.1145293.

Ministerio de Educación y FP (2022). Documento para el debate: 24 propuestas de reforma para la mejora de la profesión docente. https://educagob.educacionyfp.gob.es/comunidad-educativa/profesorado/propuesta-reforma.html.

Naps., T. L., Roessling, G., AlmstrumV., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S. & Velázquez-Iturbide (2002), J. A., Exploring the role of visualization and engagement in computer science education, ACM SIGCSE Bulletin, 35(2):131-152, https://doi.org/10.1145/782941.782998.

Norman, D. (1983) "Some observations on mental models". En: Mental Models. D. Gentner, A. Stevens (eds), pp. 7-14. Hillsdale, NJ: Erlbaum.

Novak, M., Joy, M. & Kermek, D. (2019). "Source-code similarity detection and detection tools used in academia: A systematic review", 19(3): artículo 27, mayo 2019, https://doi.org/10.1145/3313290.

Oliver, D., Dobele, T., Greber, M. & Roberts, T. (2004) "Comparing course assessments: When lower is higher and higher, lower", Computer Science Education, 14(4):321-341. https://doi.org/10.1080/0899340042000303465

Paiva, J. C., Leal, J. P. & Figueira, Á. (2022). "Automated assessment in computer science education: A state-of-the-art review", ACM Transactions on Computing Education, 22(3): artículo 34, junio 2022, https://doi.org/10.1145/3513140.

Parsons, D. & Haden, P. (2006) "Parsons' programming puzzles: A fun and effective learning tool for first programming courses". En Proceedings of the 8th Australasian Conference on Computing in Education (ACE 2006), pp. 157-163 https://dl.acm.org/doi/abs/10.5555/1151869.1151890.

Peters. A-K. ”Students’ Experience of Participation in a Discipline—A Longitudinal Study of Computer Science and IT Engineering Students”. ACM Transactions on Computing Education. 19(1):1-28 2019. https://doi.org/10.1145/3230011

Prince, M. (2004) “Does active learning work? A review of the research”, Journal of Engineering Education, 93(3):223-231. https://doi.org/10.1002/j.2168-9830.2004.tb00809.x.

Pujolàs, P. & Lago, J. R. (Coord.) (2011). El programa CA/AC (“Cooperar para Aprender/Aprender a Cooperar”) para enseñar a aprender en equipo. Implementación del aprendizaje cooperativo en el aula. Universidad de Vic. https://cife-ei-caac.com/wp-content/uploads/2015/06/EL_APRENDIZAJE_COOPERATIVO.pdf.

Ragonis, N. (2012) “Type of questions – The case of Computer Science”, Olympiads in Informatics, 6:115-132. https://www.researchgate.net/profile/Noa-Ragonis/publication/241011964_Types_of_Questions_in_Computer_Science_Education/links/56683d2f08ae8905db8f7099/Types-of-Questions-in-Computer-Science-Education.pdf.

Reas, C. & Fry, B. (2014). Processing: A Programming Handbook for Visual Designers, 2ª edición, Cambridge; Massachusetts: The MIT Press.

Ruf, A., Berges, M., Hubwieser, P. (2015). “Classification of programming tasks according to required skills and knowledge representation”, En Brodnik A, Vahrenhold J (eds.) Proceedings of the 8th International Conference on Informatics in Schools (ISSEP 2015). Springer, LNCS 9378, pp. 57–68. https://doi.org/10.1007/978-3-319-25396-1_6.

Salazar Gomez, A.F., Bagiati, A., Breazeal,C., Radovan, J. & Kennedy, K. (2022) “Learning Journeys for Scalable AI Education: An MIT - USAF Collaboration” SEFI 2022. Barcelona 19-22 September 2022.

Sanmartí, N., & Mas, M. (2016). Les rúbriques per a una avaluació plantejada com a aprenentatge. Perspectiva Escolar, 390, 36–31. https://www.rosasensat.org/les-rubriques-per-a-una-avaluacio-plantejada-com-a-aprenentatge/.

Sanmartí Puig, N. (2020). Evaluar y aprender: un único proceso. Ediciones Octaedro.

Sinclair, J. & Kalvala, S. (2015). “Exploring societal factors affecting the experience and engagement of first year female computer science undergraduates”. Koli Calling ‘15. https://doi.org/10.1145/2828959.2828979.

Shrestha, N., Botta, C., Barik, T. & Parnin, C. (2022) “Here we go again: Why is it difficult for developers to learn another programming language?” Communications of the ACM, 65(3):91-99, marzo 2022. https://doi.org/10.1145/3511062.

Shulman, L.S. (1986) “Those who understand: Knowledge growth in teaching”, Educational Researcher, 12(2):4-14. https://doi.org/10.2307/1175860.

Soloway, E. (1983) "Cognitive strategies and looping constructs: An empirical study", Communications of the ACM, 26(11):853–860, Nov. 1983. https://dl.acm.org/doi/pdf/10.1145/182.358436.

Sorva, J. (2012) “Appendix A: Misconception catalogue”. En Visual Program Simulation in Introductory Programming Education, tesis doctoral, Department of Computer Science and Engineering, Aalto University, Finlandia, págs. 358-368. https://aaltodoc.aalto.fi/bitstream/handle/123456789/3534/isbn9789526046266.pdf.

Sorva, J. (2013) "Notional machines and introductory programming education", ACM Transactions on Computing Education, 13(2): artículo 8, 2013, https://doi.org/10.1145/2483710.2483713.

Sorva, J., Karavirta, V. & Malmi, L. (2013) "A review of generic program visualization systems for introductory programming education", ACM Transactions on Computing Education, 13(4): artículo 15, Nov. 2013. https://doi.org/10.1145/2490822.

Sweller, J., Ayres, P. & Kalyuga, S. (2011) Cognitive Load Theory. Nueva York, NY: Springer.

Taylor-Smith, E., Smith, S. & Smith, C. (2019). "Identity and Belonging for Graduate Apprenticeships in Computing: The experience of first cohort degree apprentices in Scotland." ITiCSE 2019. https://doi.org/10.1145/3304221.3319753.

The Royal Society. (2017). After the Reboot: Computing Education in UK Schools. https://royalsociety.org/topics-policy/projects/computing-education/ Policy Report.

Tucker, A. J. (1991). "Computing Curricula 1991", Communications of the ACM, 34(6):69-84, Jun. 1991. https://doi.org/10.1145/103701.103710.

Valero-García, M., & Díaz de Cerio, L. M. (2005). Autoevaluación y co-evaluación: estrategias para facilitar la evaluación continuada. En Actas del Simposio Nacional de Docencia en Informática (SINDI), Granada (pp. 25-32). https://personals.ac.upc.edu/miguel/materiales/docencia/articulos/SINDI2005.pdf.

Velázquez-Iturbide, J. Á., & Pérez-Carrasco, A. (2016) "How to use the SRec visualization system in programming and algorithm courses", ACM Inroads, 7(3):42-49. https://doi.org/10.1145/2948070.

Weintrop, D. &, Wilenski, U. (2015) "Using commutative assessments to compare conceptual understanding in blocks-based and text-based programs". En Proceedings of Eleventh Annual International Conference on International Computing Education Research (ICER 2015), pp. 101-110. https://doi.org/10.1145/2787622.2787721.

Winslow, L.E. (1996) "Programming pedagogy - A psychological overview", SIGCSE Bulletin, 28(3):17-22 y 25. https://doi.org/10.1145/234867.234872.

Wong, B. (2017). "'I'm good, but not that good': digitally-skilled young people's identity in computing", Computer Science Education, 26(4):299-317. 2017 https://doi.org/10.1080/08993408.2017.1292604.

Xie, B., Loksa, D., Nelson, G.L., Davidson, M.J., Dong, D., Kwik, H., Hui Tan, A., Hwa, L., Li, M. & Ko, A.J. (2019) "A theory of instruction for introductory programming skills", Computer Science Education, 29(2-3):205-253, https://doi.org/10.1080/08993408.2019.1565235.