Academic misconduct in projects: perspective of students and teachers of Introductory Computer Science Courses

Inés Friss de Kereki
Facultad de Ingeniería
Universidad ORT Uruguay
Montevideo, Uruguay
kereki_i@ort.edu.uy

Alejandro Adorjan
Facultad de Ingeniería
Universidad ORT Uruguay
Montevideo, Uruguay
adorjan@ort.edu.uy

***Abstract*—Academic misconduct is a frequent practice. Some examples are: to copy a work in whole or in part and present it as your own, or to buy academic work online. Introductory Computer Science courses are not an exception of these inappropriate situations. In projects (coding activities done in groups outside class) we use automatic plagiarism detection tools such as MOSS and JPlag. In addition, we have included a face-to-face instance ("defense") where each student must make modifications in their own code in a maximum established time. In a previous work, we conducted a survey to all students of Computer Science II, where it turned out that 24% of 115 students expressed that "defenses" were perceived as "not useful for validating authorship". Based on this result, we decided to explore and deepen in the perceptions of both students and teachers on evaluations and propose strategies to help mitigate fraud.**

***Index Terms*— computer science I, computer science II, plagiarism, programming**

I. INTRODUCCION

El fraude académico es un problema frecuente dentro de las instituciones educativas [1]. "El plagio, la colusión y otras formas de conducta académica inadecuada han sido un aspecto lamentable pero inevitable del ámbito académico estudiantil" [2]. El plagio, se puede definir "como el reuso de procesos o resultados sin especificar explícitamente el autor original y la fuente" [3].

Estas conductas inapropiadas perjudican seriamente el proceso educativo en distintos aspectos, desde no desarrollar las competencias curriculares y habilidades esperadas de los estudiantes, hasta el impacto negativo respecto a la reputación de la institución [4]. A nivel universitario el plagio es preocupante, así como el crecimiento y la extensión de estos incidentes en la disciplina a nivel profesional [5].

El fraude "no está confinado a la presentación de ensayos o producción escrita, es a su vez un problema que afecta directamente a las disciplinas del área de educación en computación, donde los estudiantes deben programar la solución a un problema, el cual deberá ser evaluado en relación a su calidad y correctitud" [6].

Los estudiantes presentan distintas posturas en relación a la problemática del fraude. Si bien existen estudiantes extremadamente honestos y que siguen los códigos de ética, otros eligen el camino del plagio, y en ciertas ocasiones utilizan tecnologías o metodologías sofisticadas para copiar en tareas y evaluaciones [7]. Es frecuente que en los cursos de programación los estudiantes traten de copiar una parte o todo el código fuente de otros estudiantes o encontrar un código fuente similar en Internet sin citar la fuente [8]. "Los estudiantes tienen la costumbre de copiar y compartir sus soluciones de código en vez de hacerlas ellos mismos" [1]. En este contexto, el punto de inflexión entre la colaboración y el plagio no siempre es claro, y el proceso de detección, recolección, análisis, investigación y confirmación del plagio suele ser complejo, pues implica que los casos sospechosos deben ser siempre verificados manualmente por un profesor [8].

Hay herramientas automáticas de detección de plagio de codificación tales como MOSS [9] y JPlag [10]. Estas herramientas permiten detectar ciertos aspectos de plagio a nivel de codificación entre el conjunto de tareas realizadas por estudiantes. Además de su uso, uno de los enfoques para verificar autoría suele ser la realización de evaluaciones personalizadas de los trabajos realizados fuera del ámbito universitario [4].

En los cursos de Programación I y Programación II en la Universidad ORT Uruguay, además de las herramientas de detección automática y actividades de difusión como prevención, hemos incluido una instancia presencial por cada proyecto o tarea extensiva (denominada "defensa") donde cada estudiante debe realizar modificaciones en el propio código del programa entregado en un tiempo máximo establecido de una hora [11].

En diciembre de 2016 realizamos una encuesta a todos los estudiantes de Programación II en la que detectamos que, según 24% de 115 estudiantes, estas instancias eran percibidas como de poca utilidad para validar la autoría de las tareas [11].

En este trabajo, nos proponemos explorar más en detalle acerca de las percepciones, tanto de los estudiantes como de los docentes, acerca de las evaluaciones de los trabajos de carácter

978-1-5386-3057-0/17/$31.00 ©2017 IEEE

extensivo y grupal, con el objetivo de proponer estrategias y sugerencias para mitigar la problemática del fraude.

Este artículo está organizado de la siguiente forma: en la sección 2 se presentan conceptos sobre fraude académico, en la sección 3 se describen los cursos iniciales de Programación en la Universidad ORT Uruguay, en la sección 4 se presenta la investigación realizada y sus resultados, y finalmente en la sección 5 se ofrecen conclusiones y líneas de trabajo.

II. Fraude Académico

El plagio se puede presentar en distintas formas. Es posible enumerar varios aspectos como ser copiar una obra o parte de ella y presentarla como propia, comprar trabajos y atribuirse autoría, o incluir coautores que no han participado en la elaboración del trabajo [12][13]. Fraser extiende la definición de plagio para considerarlo como el acto de copiar el trabajo de otra persona independientemente de si el autor original es consciente de dicha práctica o no [14].

Colusión es una actividad de grupo no permitida [14]. Konecki et al la definen como una categoría particular de plagio, y es cuando tareas individuales son realizadas por dos o más estudiantes [1]. Esta práctica estaría ubicada entre la colaboración y el plagio y, su límite es flexible, lo cual es problemático, según indica Fraser [14].

En el área de computación, Parker y Hamblen definen un programa plagiado como un programa producido a partir de otro programa con pocos cambios. Entre los factores que contribuyen a la copia señalan el acceso sencillo a redes, facilidad de uso de los editores y clases de tamaño grande [15]. Clarke y Lancaster refieren a contratar a otras personas para que escriban el código [16]. Hammond define "ciber plagiarismo", como copia de materiales disponibles en Internet [17].

En general, las instituciones universitarias poseen un código de honestidad académica, sin embargo aún se siguen presentando incidentes de plagio, fraude o deshonestidad académica [18]. MIT refiere que se consideran faltas graves acciones tales como plagio, copia, colaboración no autorizada e interferencia deliberada con la integridad del trabajo de otros [19]. El código de honor de Stanford señala que los estudiantes, individual y colectivamente no darán ni recibirán ayuda en las evaluaciones, en la preparación de reportes ni en cualquier otro trabajo que se utilice para ser calificado [20]. En particular, el código de honor de Universidad ORT Uruguay toma aspectos similares [21].

Si bien la erradicación del plagio es probablemente un objetivo inalcanzable, el enfoque ético y los métodos de detección son posibles caminos que permiten mitigar el problema [5]. Sureda et al [22] presentan un estudio detallado en relación a la visión del docente acerca de las causas que llevan a los estudiantes a realizar plagio en los trabajos académicos y concluyen que se debería "recobrar ciertos principios de ética, compromiso y esfuerzo que parecen haber quedado desterrados del mundo académico". Si bien las comunidades académicas del área de ingeniería plantean principios y sugerencias en relación a la ética y a la actitud frente al tema tanto profesional como estudiantil, la presentación de estos contenidos se encuentra restringida en cada uno de los cursos de la currícula recomendada [23], [24].

Dentro de los métodos de detección, hay variadas recomendaciones. Encontrar manualmente plagio en código fuente es muy trabajoso, más aún debido a intentos de "ocultar" los cambios [25]. Esta detección a mano no sólo es ineficiente sino lleva mucho tiempo y esfuerzo [26]. Hay disponibles múltiples herramientas para detección automática tales como MOSS [9], JPlag [10], PDE4Java [26] o Parikshak [27].

Otra posible sugerencia es la de eliminar los trabajos no supervisados para pasar a exámenes en laboratorio [28]. Esta propuesta implicaría que sólo se podrían realizar tareas de muy pequeña escala. Además, Carroll y Appleton proponen como buenas prácticas: reescribir y modificar las tareas cada vez que el curso es dictado, diseñar evaluaciones con múltiples soluciones e informar a los estudiantes acerca de las políticas institucionales [2]. Los frecuentes cambios de tareas son también una estrategia recomendada por Zakova [29]. Sant propone pruebas de "reutilización" de código: a partir del uso de código base para crear una nueva solución con especificaciones diferentes [30].

Koss y Ford consideran que el desarrollo de código es un proceso, no un evento aislado, por lo cual se podrían revisar las diferentes versiones para determinar y verificar dicho el proceso [31]. Manoharan [18] propone la realización de evaluaciones personalizadas generadas automáticamente, donde cada estudiante tiene un único problema a resolver reduciendo así los incidentes de plagio comparados con las tareas tradicionales.

Kumar y Sobhan indican, entre otras propuestas penalizar a quienes sean responsables y crear conciencia sobre los derechos de "copyright" y leyes [32], aunque también indican que ninguna de ellas eliminará la tentación de cometer fraude. En particular, refieren a organizar seminarios y simposios con el objetivo de crear conciencia acerca del impacto negativo del plagio [32], propuesta también sugerida por Pandey et al [33]. Robert [34] refiere también a establecer políticas consistentes y explícitas contra el plagio así como asegurarse que los estudiantes las comprendan.

La propia perspectiva de los estudiantes acerca del plagio es analizada en Zhang et al [35]. En dicho trabajo se comparan las percepciones de los estudiantes chinos con los de Reino Unido y concluyen que el reuso de código sin reconocer el origen así como referir en forma incorrecta es un problema común en los dos países. También indica que los estudiantes no están seguros sobre cuáles son los límites entre actividades académicas legítimas y plagio.

Comas y Sureda citan como factores explicatorios del plagio desde la perspectiva de los estudiantes: aspectos y comportamientos (ej. mala gestión del tiempo, elevado número de actividades a hacer), oportunidades debido a la tecnología para ubicar, copiar y pegar información y aspectos relacionados con los profesores y, o, características de la asignatura [36]. Joy et al sugieren clarificar las ideas sobre compartir y plagiar [37].

Como se presentó, el fraude o deshonestidad académica en el área de Educación en Ingeniería es un tema preocupante con múltiples posibles enfoques y sin garantías acerca de cómo evitarlo. En la siguiente sección se describen las características

de los cursos de Programación I y II en la Universidad ORT Uruguay y se presenta cuál es la metodología actual para prevención y detección del fraude.

III. Programación I y Programación II

Programación I es una materia de primer semestre en las carreras de Ingeniería y Licenciatura en Sistemas, Ingeniería Eléctrica, Electrónica y Telecomunicaciones. Tiene por objetivo aprender a programar usando técnicas de programación orientada a objetos y desarrollar aplicaciones en Java.

La temática incluye: seudocódigo, estructuras de control y variables, introducción a Java, uso de clases estándar, creación de clases, asociación, agregación, herencia, uso de colecciones, ordenación y búsqueda. Dura 15 semanas, en las cuales se incluyen 4 horas de clases teóricas y 2 horas de laboratorio en cada semana.

Para aprobar el curso deben realizarse 2 proyectos (o trabajos extensivos) de programación en Java (de 15 y 25 puntos), en equipos de 2 estudiantes y un parcial individual (de 45 puntos). También se considera la actuación en clase (15 puntos). Para obtener la aprobación del curso, deben obtenerse como mínimo 70 puntos entre 100, y en caso de obtenerse 86 o más, la materia es "exonerada", es decir, no hay que dar examen posterior.

El primer proyecto obligatorio consiste en la implementación de 3-4 clases en Java, relacionadas por asociación. El segundo implica ampliar el diseño incorporando más clases y relaciones de agregación/composición y herencia.

Programación II, materia del segundo semestre de las mismas carreras, tiene por finalidad profundizar en los conceptos de orientación a objetos vistos en el primer semestre, e incluir elementos más avanzados en cuanto a construcción de programas, algoritmos y nociones de diseño.

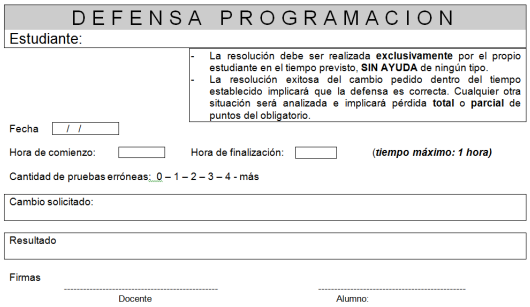
La temática consiste en: repaso de sintaxis Java, matrices, algoritmia avanzada, nociones de diseño, concepto de patrón, diseño de interfaz, manejo de archivos, persistencia, serialización y excepciones. La duración y forma de evaluación es similar a la de Programación I.

El primer proyecto obligatorio de esta materia consiste en la implementación de un juego, con alta complejidad de diseño y algoritmia. El segundo obligatorio consiste en incluir interfaz tipo Windows, persistencia, manejo de archivos y modificaciones al juego. Todos los cursos se renuevan las propuestas de obligatorio.

Para todos los obligatorios se incluye una defensa individual, cuyo formulario se presenta en la Fig. 1: Defensa.

Para realizar la defensa, en el laboratorio cada estudiante recibe su obligatorio y una propuesta de cambio. El cambio puede referirse a corregir un error (ejemplo: "al borrar el primer cliente se cae el programa") o realizar una modificación (ejemplo: "al agregar una ficha al juego, mostrar la cantidad de fichas que están colocadas en esa misma columna"). Se anota la hora de comienzo. El cambio es el mismo para los dos integrantes del equipo.

El proceso es: cada alumno debe instalar el obligatorio y realizar el cambio en un tiempo máximo de una hora. Cuando lo termina, el docente lo verifica. Si está correcto, aprueba la defensa y obtiene sus puntos de obligatorio. Si está incompleto o incorrecto, tiene oportunidad de corregir, y recibirá una penalización de puntos. Esta penalización es discutida entre todos los docentes para unificar criterios. En caso de no poder hacer el cambio en el tiempo previsto, se pierde la defensa, por lo cual no se obtienen puntos en el obligatorio y se pierde el curso.


DEFENSA PROGRAMACION

Estudiante:

- La resolución debe ser realizada **exclusivamente** por el propio estudiante en el tiempo previsto, **SIN AYUDA** de ningún tipo.
- La resolución exitosa del cambio pedido dentro del tiempo establecido implicará que la defensa es correcta. Cualquier otra situación será analizada e implicará pérdida **total** o **parcial** de puntos del obligatorio.

Fecha / /

Hora de comienzo: Hora de finalización: *(tiempo máximo: 1 hora)*

Cantidad de pruebas erróneas: 0 – 1 – 2 – 3 – 4 - más

Cambio solicitado:

Resultado

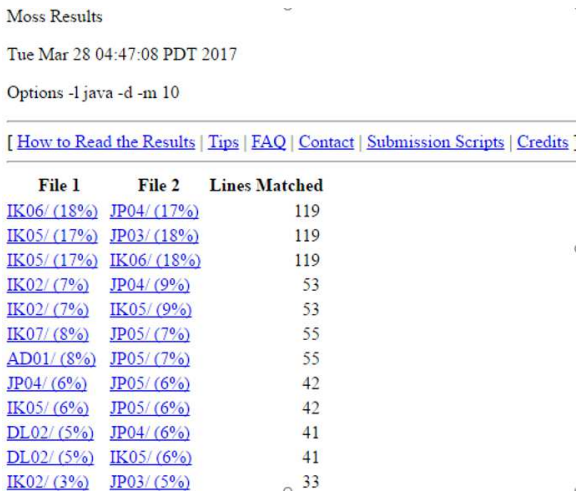
Firmas

Docente Alumno:

Fig. 1. Formulario para la defensa

Tanto para Programación I como para Programación II los grupos de cada clase no superan los 25-30 estudiantes, por lo que las defensas de un grupo llevan aproximadamente 2 horas en un laboratorio con 15-20 computadoras. En total, cada comienzo de curso, hay aproximadamente 225-250 alumnos entre Programación I y Programación II.

Para la detección automática de plagio en nuestros cursos utilizamos MOSS [9] y JPlag [10]. Los docentes renombran cada entrega de sus estudiantes con sus propias iniciales, en forma consecutiva (ej. JP01, JP02, …), y se verifica conjuntamente con todos las demás entregas de estudiantes que cursan en ese semestre con otros docentes. Se obtiene un reporte similar al presentado en la Figura 2: Resultados del MOSS. El propio informe de la herramienta explica cómo leer los resultados: se incluye por cada pareja de entregas, el número de líneas de código que coinciden. Además, a cada archivo le asigna un porcentaje, que es el porcentaje del código en un archivo considerado como coincidente con el código del otro archivo. En todos los casos, los números más altos significan mayores coincidencias de código y se analizan a mano.


Moss Results

Tue Mar 28 04:47:08 PDT 2017

Options -l java -d -m 10

[How to Read the Results | Tips | FAQ | Contact | Submission Scripts | Credits]

File 1	File 2	Lines Matched
IK06/ (18%)	JP04/ (17%)	119
IK05/ (17%)	JP03/ (18%)	119
IK05/ (17%)	IK06/ (18%)	119
IK02/ (7%)	JP04/ (9%)	53
IK02/ (7%)	IK05/ (9%)	53
IK07/ (8%)	JP05/ (7%)	55
AD01/ (8%)	JP05/ (7%)	55
JP04/ (6%)	JP05/ (6%)	42
IK05/ (6%)	JP05/ (6%)	42
DL02/ (5%)	JP04/ (6%)	41
DL02/ (5%)	IK05/ (6%)	41
IK02/ (3%)	JP03/ (5%)	33

Fig. 2. Resultados del MOSS

En los cursos de Programación I y II, además del uso de las herramientas automáticas citadas y las defensas, se difunde en clase el código de honor y se realizan otras actividades de prevención, como, por ejemplo, cambios de tareas, presentación y discusión de un decálogo de buenas prácticas [11].

En caso de detectarse una situación inapropiada, es analizada por los docentes y, según la gravedad, en conjunto con la Directora de la Escuela, quien establece la penalización. Dicha penalización va desde la pérdida del curso hasta la suspensión de los derechos académicos por un plazo extendido.

En resumen, la propuesta pedagógica utilizada en dichos cursos, a través de las instancias de defensas personalizadas relacionadas con los trabajos grupales realizados fuera del ámbito universitario, en combinación con la utilización de las herramientas de detección automática de análisis de código y las actividades de difusión, permiten en cierta forma mitigar la problemática del plagio en los cursos iniciales de programación. La cantidad de casos descubiertos con MOSS [9] y JPlag [10] han disminuido en los últimos semestres (en particular, en los últimos 4 años -2013 a 2016- no se han detectado casos con las herramientas [11]), pero con las defensas se continúan detectando dos o tres casos en cada grupo de 25-30 estudiantes [11].

IV. Investigación y resultados

A. *Antecedentes*

En la encuesta presentada en [11] de diciembre de 2016, resultó que para 24% de 115 estudiantes, las defensas eran percibidas como "no útil" para validar la autoría de las tareas. A partir de los resultados obtenidos en dicha encuesta, entendimos necesario profundizar la investigación para tratar de determinar su visión sobre la defensa y sobre aspectos de plagio, con enfoque similar, entre otros, al propuesto por Joy et al [37]. En febrero de 2017 realizamos una encuesta ampliada anónima a los estudiantes que cursan la materia "Taller de Programación", asignatura obligatoria de la currícula que continúa a las materias Programación I y Programación II. Es requisito para tomar el taller que se tengan aprobados los cursos de las 2 materias citadas. Estos estudiantes conocen las defensas pues tuvieron en total 4 instancias de defensa -o más si recursaron- en dichos cursos previos. Así mismo, entrevistamos a 4 docentes de las materias Programación I y II para conocer su visión sobre las ventajas, desventajas y posibles sugerencias de mejora. En los siguientes apartados se presentan los resultados de las encuestas a estudiantes, entrevistas a docentes, reflexiones y análisis global.

B. *Perspectiva de los estudiantes*

Una vez realizada la encuesta, de los 104 estudiantes inscriptos en "Taller de Programación", 51 entregaron la misma. Se los consultó sobre cuál perciben es el objetivo de la defensa, permitiendo respuestas de texto libre. Analizando las 47 respuestas obtenidas (4 no respondieron esta primera parte), se pueden agrupar en 5 categorías según el énfasis:

1) *Validar autoría individual*
   en esta categoría se destaca sobre si se es el autor o no. Ejemplos de frases expresadas por los alumnos son: "comprobar si el alumno hizo el obligatorio", "saber si el estudiante hizo el obligatorio", "verificar que el alumno realizó el trabajo";
2) *Validar autoría grupal*
   el foco aquí está en el trabajo de todos los integrantes del equipo. Ejemplos: "asegurarse que todos los del equipo trabajaron", "comprobar que trabajaron los 2 del equipo", "conocer si cada uno de los integrantes trabajó en el obligatorio".
3) *Conocimiento del código*
   esta categoría refiere específicamente al código. Ejemplos: "confirmar que conoce el código", "confirmar que el alumno vio el código de su obligatorio", "comprobar si sabe el código".
4) *Aprender*
   aquí el énfasis está en el aprendizaje. Ejemplos: "reafirmar conocimientos", "probar que sé del tema y sé hacer cambios", "saber desenvolverse ante un error de cada uno";
5) *Detectar plagio*
   esta categoría refiere directamente a percibir la defensa como herramienta frente al plagio. Ejemplos: "descartar sospechas de plagio", "evitar plagio", "verificar que no haya plagio".

Los porcentajes de las categorías se presentan en la Figura 3: Distribución de respuestas. Es interesante que el mayor porcentaje (34%, 16 estudiantes) es otorgado al aprendizaje.

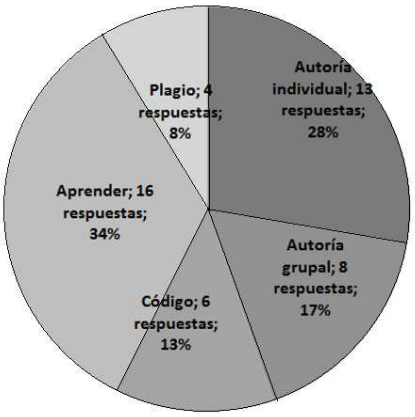


Fig. 3. Distribución de respuestas

A su vez, se les solicitó que brindaran sugerencias para mejorar las defensas. Las más frecuentes entre quienes respondieron esa consulta fueron: a) que sean más difíciles (5 casos), b) que no se pierda el curso por realizarla mal (3 casos), c) asegurarse que se deba conocer el código ("a veces se puede hacer sin conocer tan bien el código") (2 casos) y d) que sean más parejas: "algunos cambios son más sencillos que otros" (2 casos).

Además, en la última parte de la encuesta se les planteó analizar 8 casos específicos e indicar si lo consideraban "sí, es plagio, o una situación inválida o no ética", "no, no es plagio, no es inválido", o "no sé". En el diseño de esta parte se trataron de cubrir diversas situaciones y escenarios posibles.

En la siguiente figura (Fig.4 Casos) se resumen los casos, indicando por cada uno, la cantidad n de respuestas brindadas y su porcentaje correspondiente. A continuación, se detalla cada uno.

Caso	1		2		3		4		5		6		7		8	
	n	%	n	%	n	%	n	%	n	%	n	%	n	%	n	%
sí	35	76	36	72	6	13	27	57	24	55	2	4	39	85	0	0
no	5	11	12	24	29	60	13	28	11	25	45	94	5	11	48	98
no sé	6	13	2	4	13	27	7	15	9	20	1	2	2	4	1	2

Fig. 4. Casos

El primer caso es: "María encuentra en Internet un código que resuelve una parte del obligatorio y lo utiliza como propio, no indica el link". Es claramente un caso de una situación inválida, pues falta indicar la referencia. 76% de los estudiantes lo percibe también de esta forma.

El caso 2 refiere a una persona que encontró listados en una impresora del laboratorio, toma ideas de ese código y las utiliza como propias. También aquí es una situación no válida y fue vista de esta manera por 72% de los estudiantes.

El caso 3 trata de "Luisa pidió ayuda a su hermano que es experto en computación para resolver el obligatorio. El hermano le diseñó el algoritmo en seudocódigo y ella lo codificó". Esta situación fue interpretada por la mayoría de los estudiantes como válida, aunque sería una situación indebida, pues la idea y diseño del algoritmo fue realizada por otra persona, que no se indica claramente que es citada. Luisa solamente hizo la codificación a partir del seudocódigo. Este es el caso que más confusión generó a los estudiantes, ya que 27% indicó no saber si es problema o no y 60% dijo incorrectamente que no era problema.

En el caso 4 se especifica que el obligatorio debe ser realizado por 2 personas, pero dado que "Luis, Pablo y Jorge son amigos hace años y han trabajado juntos previamente, deciden seguir trabajando juntos y hacer dos entregas similares". Este caso de colusión fue interpretado como indebido por 57% de los estudiantes.

En el caso 5, se indica que una persona sube la consulta de una parte del obligatorio a un foro de programación, donde recibe varias soluciones y las utiliza como propias. 55% de los estudiantes perciben esta situación como no válida.

En el caso 6, "Juana encontró en un libro de C++ un algoritmo similar al que se requiere para el obligatorio, lo traduce a Java y lo utiliza, citando la fuente". 94% indican correctamente que no es un caso de situación problema.

En el caso 7, se señala que "Hugo contrata a un programador experto para que le resuelva el obligatorio y entrega ese código como propio." 85% de los estudiantes indican que este es un caso de plagio, no válida y, o no ética.

Finalmente, en el caso 8, la situación es: "Valeria asiste a las clases de apoyo para consultar dudas sobre la implementación. Aplica las guías y desarrolla los algoritmos requeridos.". 98% de los estudiantes indican correctamente que no es una situación problemática.

Analizando los distintos casos, la situación de recibir ayuda externa inapropiada (presente por ejemplo en los casos 3: diseño de la solución por otra persona y 5: solución propuesta por otra persona en un foro) es la que más dudas o confusión genera a los estudiantes, quienes no las perciben claramente como situaciones de fraude académico. Igualmente, casos tales como el 1 o el 2, donde no se citan la fuente, presenta 24-28% de estudiantes que no los detectan como inconveniente o no saben si lo es. El trabajo en equipo de 3 o más personas (caso 4) no es visto como problemático o inconveniente para 43% de los estudiantes.

En resumen, según las respuestas brindadas por los estudiantes a los casos presentados, hay situaciones de fraude académico que no están siendo percibidas por ellos de esta forma (casos tales como ayuda externa, trabajo en equipo indebido, y, o ausencia de citación). A modo preliminar podrían vincularse estas percepciones con la visión de "no utilidad" de la defensa.

C. Perspectiva de los Docentes

Desde la perspectiva docente, mientras algunos trabajos refieren a que los docentes atribuyen el plagio académico como una consecuencia del propio comportamiento del rol y las estrategias didácticas y metodológicas utilizadas [22], otros reportan que la visión de los estudiantes y docentes en general difieren, y que las opiniones subjetivas juegan un rol más relevante en relación a la actitud de plagio en trabajos académicos [38].

Para poder obtener la visión desde esa perspectiva, en el presente trabajo seleccionamos y entrevistamos a 4 docentes de los cursos de Programación I y Programación II ("Doc1" y "Doc2" altamente experimentados: más de 10 años de docencia, "Doc3" con experiencia media y "Doc4" con poca experiencia: dos años de docencia). Les consultamos sobre ventajas y desventajas de las defensas, así como posibles sugerencias para la mejora de la evaluación.

Entre los aspectos acerca de las ventajas de realizar defensas, Doc1 destaca "el hecho que sean individuales y que impliquen instalar el obligatorio y modificarlo, pues permite ver si el estudiante conoce las herramientas y está acostumbrado a su uso". Doc2 indicó que "permite identificar si los dos estudiantes del equipo conocen el obligatorio" y, más aún, en caso de trabajo individual, "permite identificar casos de ayuda externa". Además los enfrenta a una instancia de la etapa de mantenimiento del sistema. Para Doc4, permite detectar autoría: "quien no hizo absolutamente nada del obligatorio queda detectado". Doc3 agrega además que esta instancia permite tener contacto con aquellos alumnos que no vienen a clase regularmente.

O sea, las ventajas citadas refieren a constatar autoría, así como acercarlos a la vida profesional y a tener contacto más directo con los estudiantes.

Respecto a desventajas, Doc1 indica que "si bien el sistema no es perfecto, es -por lejos- el mejor que hemos tenido", pues "permite a quienes conocen y trabajaron en el obligatorio resolver la defensa rápidamente, y al menos obliga a quienes no lo hicieron a preocuparse por conocer o entender el trabajo".

Doc2 señala la presión por el tiempo acotado puede llevar en algún caso a no poder realizar la modificación, no por desconocer el código sino por el estado anímico (ansiedad, nervios). Destaca además que "este aspecto se lo puede considerar como una instancia de aprendizaje pues es una situación que le va a ocurrir al estudiante en su futuro trabajo profesional".

Esta misma situación la señala Doc3, donde "hay casos de alumnos que trabajan en el obligatorio, pero por nervios se bloquean y terminan perdiendo puntos que quizás no merecían perder, aunque es parte de la instancia". También señala que

durante las defensas, hay "momentos de intercambio de estudiantes", donde casi simultáneamente terminan de hacer los cambios varios estudiantes y el docente debe atender muchos casos en poco tiempo, para poder verificar las defensas y permitir el ingreso de nuevos estudiantes al laboratorio.

Doc4 refiere que a veces, puede tocarle a un estudiante un cambio de una parte que entendían y pasar la defensa aunque haya hecho muy poco del obligatorio.

En suma, las desventajas incluyen factores anímicos o personales que puede influir negativamente, casos de aprobar la defensa "injustamente" así como el manejo de la propia defensa debido a la cantidad de estudiantes.

Como sugerencias, Doc1 plantea que, con el formato actual parece evaluarse más que autoría, "la comprensión y el conocimiento de la solución, lo cual no es poco", por lo cual "se podría agregar que expliquen la mecánica de trabajo y cómo lo dividieron"; "si bien esto no va obligarlos a trabajar en forma pareja, puede que sirva para hacerlos pensar y reflexionar sobre el tema". Doc2 refiere a realizar en clases previas simulacros de defensa donde se incorpore el factor tiempo para hacer la modificación.

Asimismo, Doc 3 sugiere controlar que no tengan material a mano (ej. celular) para evitar posibles ayudas indebidas, incluir más docentes en la defensa y que en la documentación se incluya el detalle de qué hizo cada uno, de forma de evaluar sobre lo que el propio alumno dice que hizo. Propone también como sugerencia a evaluar a futuro si se pudiera hacer la defensa eligiendo al azar un representante de cada grupo y que su defensa sea para los dos del equipo, como forma de fomentar más el trabajo en equipo. Doc4 indica que se podría verificar con otras asignaturas para conocer otras modalidades.

En resumen, entre las sugerencias se indican incluir explicación y detalles de la forma de trabajo para usar ese dato como insumo para la defensa, incorporar simulacros y más docentes y analizar la posibilidad de que un integrante presente la defensa por el equipo.

D. Aspectos comunes de las perspectivas

Como se presentó, las perspectivas de los alumnos acerca de las defensas se pueden categorizar en: validación de autoría individual, grupal, conocimiento del código, aprender o detección de plagio. Pueden relacionarse con las respuestas de los docentes.

Los aspectos destacados por los docentes como ventajas (ejs.: constatar autoría, conocimiento del obligatorio, acercamiento a la vida profesional) podrían vincularse a las categorías "validación autoría individual", "aprender" y "detección de plagio".

La categoría de "conocimiento del código" está contemplada en las posibles situaciones problemáticas citadas por los docentes que implican su conocimiento: por ejemplo: presión de tiempo, nervios y tener que modificar justo la parte que hicieron. También, la categoría "validación de autoría grupal" podría verse en las sugerencias de los docentes: "detallar trabajo en equipo" o "que un representante defienda por equipo".

Las sugerencias presentadas por los estudiantes acerca de homogenizar la dificultad y el nivel de los cambios no son planteadas expresamente por los docentes, siendo éstos interesantes aspectos a tener en cuenta cuando se diseñen los requerimientos para las defensas. También, la consideración sobre la posible sobrecarga de atención simultánea, referida por los docentes no fue visualizada por los estudiantes.

V. Conclusiones y Trabajo Futuro

El tema de fraude académico en el sistema institucional es un hecho muy delicado y severo. Si bien en el área de Ingeniería y, en particular, en el área de programación no existe una herramienta infalible de detección automática de similitud de código, la combinación de estrategias didácticas y educación de los aspectos éticos sigue siendo uno de los posibles caminos a transitar para mitigar el fraude académico.

En este trabajo se presentan las perspectivas de los estudiantes y de los docentes en relación a la evaluación de trabajos extensivos en cursos iniciales de programación. Ellas se focalizan en aspectos relativos a la validación de autoría, conocimiento del código, aprendizaje y plagio.

Adicionalmente, por parte de los estudiantes hay ciertas situaciones que no son percibidas como fraude. Según sus respuestas, el límite entre la colaboración, colusión y fraude no parece estar alineado con la perspectiva docente. Desde la perspectiva docente la instancia de defensa es una metodología establecida como detección de autoría. Los estudiantes que no realizaron el trabajo o recibieron ayuda muestran en esa instancia que no tienen las competencias necesarias para poder realizar un cambio simple en un breve lapso.

El uso de las herramientas automáticas permite complementar la instancia anterior, detectando incidentes de plagio de código independientemente de poder realizar una defensa correcta del mismo.

Como posibles mejoras a ser incorporadas en los cursos y defensas se sugiere:

1) Incluir más información y actividades para discutir y aclarar los límites entre colaboración válida e inválida.
2) En las defensas incluir cambios más amplios y complejos (que cubran varios aspectos y no un único punto del obligatorio).
3) Agregar instancias en el curso en las cuales los propios estudiantes sugieran cambios a ser realizados por otros, a los efectos de que ellos mismos piensen y validen la complejidad del cambio.
4) Tener dos docentes en forma presencial durante la defensa con el objetivo de tener un mayor control y fluidez.
5) Prestar atención a la duración de las defensas realizadas por los estudiantes como posible predictor de no autoría (a un mayor tiempo de resolución se podría poner en duda el esfuerzo realizado en la tarea realizada y así estar más atento a posibles situaciones de fraude).

Se propone implementar estas sugerencias y, como trabajo futuro, encuestar y evaluar nuevamente. Adicionalmente, se planea realizar un estudio más detallado que permita establecer cuáles son las principales causas que podrían llevar a un estudiante a cometer fraude académico en los cursos iniciales de programación, para diseñar estrategias que permitan mitigar esas situaciones. También, buscar más herramientas para concientización de la relevancia de la temática, no sólo a nivel curricular sino como una de las más importantes también en la futura vida profesional.

REFERENCIAS

[1] M. Konecki, T. Orehovacki y A. Lovrencie, "Detecting computer code plagiarism in higher education", Procs of ITI 2009, Croatia, 2009

[2] J. Carroll y J. Appleton, "Plagiarism: A Good Practice Guide". JISC Report, 2001

[3] IEEE, "Introduction to the Guidelines for Handling Plagiarism Complaints", https://www.ieee.org/publications_standards/publications/rights/plagiarism.html, accedido diciembre 2016

[4] B. Halak, y M. El-Hajjar, "Plagiarism detection and prevention techniques in Engineering Education". 2016 European Workshop of Microelectronics Education, 2016.

[5] M. A. Dyrud, "Plagiarism: It's not just for students", Proceedings of the IEEE 2014 International Symposium on Ethics in Science, Technology and Engineering, IEEE Press, 2014

[6] M. Joy, G. Cosma, J. Yin-Kim Yau y J. Sinclair, "Source Code Plagiarism - a student perspective", IEEE Transactions on Education, 54(1), 2011

[7] R. Kelley y B. Dooley, "The Technology of cheating", IEEE Int. Symp. on Ethics in Science, Technology and Engineering, 2014

[8] M. Novak, "Review of source-code plagiarism detection in academia", MIPRO 2016, Croatia, 2016

[9] MOSS, https://theory.stanford.edu/~aiken/moss/, accedido en abril 2017

[10] JPlag, jplag.ipd.kit.edu, accedido abril 2017

[11] I. Friss de Kereki, "Detecting academic misconduct in introductory computer science courses", 3rd. Conf. of Plagiarism across Europe and Beyond, Rep. Checa, Mayo 2017

[12] P. Alfaro Torres y T. de Juan Juárez, "El plagio académico: formar en competencias y buenas prácticas universitarias", RUIDERAE: Revista de Unidades de Información. Num 6, 2014

[13] R. Comas, J. Sureda, A. Casero y M. Morey, "La integridad académica entre el alumnado universitario español", *Estudios Pedagógicos,* XXXVII, No. 1, 207-225, 2011

[14] R. Fraser, "Collaboration, Collusion and Plagiarism in Computer Science Coursework", Informatics in Education, Vol. 13, No. 2, 179–195, 2014

[15] A. Parker y J. Hamblen, "Computer Algorithms for plagiarism detection". IEEE Transactions on Education, Volume 32, No. 2, May 1989

[16] R. Clarke y T. Lancaster, "Eliminating the successor to plagiarism? Identifying the usage of contract cheating sites", Second International Plagiarism Conf.: Prevention, Practice and Policy, Newcastle, UK, 2006

[17] M. Hammond, "Cyber-Plagiarism: are FE Students getting away with words?", Association of Northern Ireland Colleges, 2002

[18] S. Manoharan, "Personalized Assessment as a Means to Mitigate Plagiarism", IEEE Transactions on Education, Issue 99, 2016

[19] MIT, "Academic Integrity at MIT", https://integrity.mit.edu/handbook/academic-integrity-mit/what-academic-integrity, accedido diciembre 2016

[20] Stanford University, "Stanford Honour Code", https://communitystandards.stanford.edu/student-conduct-process/honor-code-and-fundamental-standard, accedido diciembre 2016

[21] Universidad ORT Uruguay, "Código de honor", Accedido marzo 2017, http://www.ort.edu.uy/varios/pdf/codigodehonor.pdf

[22] J. Sureda, R, Comas y M. Morey, "Las causas del plagio académico entre el alumnado universitario según el profesorado". Revista iberoamericana de educación, 50, 197-220, 2009

[23] ACM-IEEE, "Computer Science Curricula 2013", http://www.acm.org/education/CS2013-final-report.pdf, accedido abril 2017

[24] ACM-IEEE (2), "Computer Engineering Curricula 2016", https://www.computer.org/cms/professional-education/curricula/ComputerEngineeringCurricula2016.pdf, accedido abril 2017

[25] U. Inoue y S. Wada, "Detecting Plagiarisms in elementary programming courses", 9th Int. Conf on Fuzzy Systems and Knowledge Discovery, 2012

[26] A. Jadalla y A. Elnagar, "PDE4Java: Plagiarism detection engine for Java source code: a clustering approach". Proc of iiWAS2007, 2007

[27] S. Sharma, C. Shekhar Sharma y V. Tyagi, "Plagiarism detection tool "Parikshak"". 2015 Int. Conf on Communication, Information & Computing technology, Mumbai, India, 2015

[28] F. Culwin, A. MacLeod y T. Lancaster, "Source code plagiarism in UK HE Computing Schools". 2nd Annual LTSN-ICS Conf., London, 2001

[29] F. Záková, J. Pistej y P. Bisták, "Online tool for student's source code plagiarism detection". ICETA 2013, Slovakia, 2013

[30] J. Sant, "Code Repurposing as an Assessment Tool", ICSE, Italy, 2015

[31] I. Koss y R. Ford, "Authorship is continuous: managing code plagiarism", IEEE Security and Privacy, vol. 11, pp. 72-74, March-April 2013, doi:10.1109/MSP.2013.26, 2013

[32] S. Kumar Dey y M. Abdus Sobhan, "Impact of unethical practices of plagiarism on Learning, Teaching and Research in Higher Education: some combating strategies". ITHET 2006, 2006

[33] A. Pandey, M. Kaur y P. Goyal, "The menace of plagiarism", 4th Int. Symp. on Emerging Trends and Technologies in Libraries and Information Services, 2015

[34] E. Roberts,"Strategies for promoting academic integrity in CS Courses", 32th FIE, Boston, USA, 2002

[35] D. Zhang, M. Joy, G. Cosma, R. Boyatt, J. Sinclair y J. Yau, J.: "Source-code plagiarism in universities: a comparative study of student perspectives in China and the UK", Assessment and Evaluation in Higher Education, Vol 39, N. 6,pp. 743-758, 2014

[36] R. Comas-Forgas y J. Sureda-Negre, "Academic Plagiarism: Explanatory Factors from Students' Perspective", Journal of Academic Ethics, September 2010, Volume 8, Issue 3, pp 217–232, 2010

[37] M. Joy, J. Sinclair, R. Boyatt, J.Y-K. Yau y, G. Cosma, "Student perspectives on source-code plagiarism", Int. Journal for Educational Integrity, Vol 9, No 1, 2013

[38] M. Bohlin, y S. Widén, "University teacher and student judgements on misleading behavior in study situations", HÖGSKOLAN VÄST, Nr 2016:4, 2016.