Experiencia de modularización del curso de programación en el primer semestre de Ingeniería Civil: Beneficios y desafíos pendientes

Claudio Álvarez, Facultad de Ingeniería y Ciencias Aplicadas & Centro de Investigación en Educación y Aprendizaje, Universidad de los Andes, Chile, calvarez@uandes.cl
Matías Recabarren, Facultad de Ingeniería y Ciencias Aplicadas, Universidad de los Andes, Chile, mrecabarren@uandes.cl
Pilar Gazmuri, Facultad de Ingeniería y Ciencias Aplicadas, Universidad de los Andes, Chile, mgazmuri@uandes.cl

RESUMEN

La enseñanza de la programación ha adquirido gran relevancia en la educación en ingeniería durante las últimas décadas, dada la masiva disponibilidad de recursos y herramientas computacionales aplicables en prácticamente todas las subdisciplinas. Sin embargo, los contenidos y competencias de un primer curso de programación en ingeniería son completamente nuevos para la mayoría de los estudiantes novicios, y se caracterizan por su estructura secuencial, jerarquizada y acumulativa. Estas características causan que muchos estudiantes se vean desafiados y sobrepasados al no lograr un ritmo de aprendizaje adecuado. Para enfrentar esta dificultad, los autores del presente estudio sostienen que el curso de programación puede beneficiarse de una arquitectura modular. Un formato modular permite a los estudiantes repetir el/los módulo(s) en que tienen dificultades, sin tener que repetir el curso completo en el semestre siguiente. Además, dado que todos los módulos del curso se pueden paralelizar y repetir sucesivamente, los estudiantes pueden progresar en el curso a su propio ritmo. En el presente artículo describimos la experiencia de modularización del curso Programación de la carrera de Ingeniería Civil de la Universidad de los Andes, con 228 estudiantes de primer año de plan común, y cinco docentes, en el semestre de otoño-invierno de 2017. Si bien la modularización no se tradujo en una mayor tasa de aprobación semestral, permitió evaluar el progreso de los estudiantes con frecuencia y rigurosidad, detectar los contenidos y competencias en donde tienen las mayores dificultades de aprendizaje, e identificar intervenciones focalizadas para mejorar los resultados en el futuro.

PALABRAS CLAVES: Transformación Curricular, Curso Modular, Enseñanza de Programación

INTRODUCCIÓN

La flexibilización curricular es un proceso que en Chile se ha venido desarrollando desde hace algunas décadas, con efectos que hoy contrastan con los inicios de la enseñanza de la ingeniería en el país (Gaete & Morales, 2011). Durante la mayor parte del siglo XX los currículums de escuelas y facultades de ingeniería en Chile se estructuraban en planes anuales, con asignaturas obligatorias y fijas, sin cursos electivos (UTFSM, 2017). Además, las carreras de ingeniería civil eran ofrecidas por un reducido número de instituciones a un número muy limitado de estudiantes provenientes de hogares acomodados. Hoy en cambio, las cohortes de primer año en las escuelas y facultades de mayor prestigio se acercan al millar de estudiantes, y la masividad de las carreras ha generado mayor variabilidad en el perfil de ingreso de los estudiantes, tanto en lo relativo a sus aptitudes como a su preparación pre-universitaria. Frente a esto, la flexibilidad curricular ha permitido que los alumnos puedan escoger su carga académica en cada período académico, de acuerdo a su capacidad personal. Además, las escuelas ofrecen a los estudiantes cursos propedéuticos y otras medidas para mejorar los resultados. Aun así, las carreras de ingeniería civil en Chile mantienen una duración promedio de 8,5 años, mientras que la duración nominal es de 5,5 a 6 años (SIES, 2014). La mejoría de este indicador motiva la búsqueda de innovaciones docentes efectivas por parte de las facultades y escuelas de ingeniería en Chile.

A partir de la década de 1960 en EE.UU. comenzó un proceso de mayor flexibilización de la enseñanza universitaria, a través de la modularización de los programas y cursos (Allen, 1971; Creager & Murray, 1971). De acuerdo con Goldschmid & Goldschmid (1973), un módulo es una unidad independiente y autocontenida, cuya planificación incluye una serie de actividades didácticas y de evaluación, diseñadas para ayudar al estudiante a lograr objetivos de aprendizaje bien definidos. Un curso modular difiere de uno tradicional en los siguientes aspectos (Goldschmid & Goldschmid, 1973; Dochy et al., 1989): (1) permite lograr un mayor control sobre el aprendizaje de los estudiantes, exigiéndoles demostrar los conocimientos y competencias pertinentes al módulo antes de promoverlos al módulo siguiente, (2) permite que los estudiantes repitan únicamente los módulos que reprueban sin requerir que repitan el curso completo, (3) facilita la asignación flexible de los docentes a módulos durante el período académico de acuerdo a su experticia y dominio de los temas, (4) ofrece mayor posibilidad de reutilización y mejoramiento continuo de los materiales docentes, (5) promueve una mayor frecuencia de las evaluaciones a la que son sometidos los estudiantes junto con la posibilidad de aumentar las evaluaciones formativas, y (6) ofrece a los estudiantes la posibilidad de que puedan aprender a su propio ritmo en cada curso y con un mayor grado de personalización en su formación.

En el presente artículo describimos la primera experiencia de modularización del curso de Programación de la carrera de Ingeniería Civil de la Universidad de los Andes (UANDES), con 228 estudiantes de primer año de plan común, y cinco docentes, en el primer semestre (marzo a julio) del año académico 2017. Las secciones a continuación ofrecen una descripción de antecedentes del curso de Programación, las características de la intervención realizada, y resultados tempranos del proceso de modularización, junto con una discusión de dichos resultados. El proceso de modularización no disminuyó la reprobación histórica del curso durante el primer semestre, pero los autores estiman que la reprobación en el curso para el total de alumnos de primer año sí podrá disminuir al cierre del año académico. La primera implementación de la modularización ha sentado una línea de base sobre la cual los docentes han podido discurrir una serie de medidas para mejorar los resultados académicos en años venideros, incluyendo el perfeccionamiento de los mecanismos de práctica de habilidades, retroalimentación y evaluación.

ANTECEDENTES

La enseñanza de la programación ha adquirido gran relevancia en la educación en ingeniería durante las últimas décadas, dada la masiva disponibilidad de recursos y herramientas computacionales aplicables en prácticamente todas las subdisciplinas, y el gran potencial de dichas herramientas para apoyar la resolución de problemas de creciente complejidad y sofisticación. En un primer curso de programación es común que los estudiantes de ingeniería deban desarrollar un conjunto de habilidades cognitivas y meta-cognitivas orientadas a resolver problemas a través del pensamiento computacional y la programación (Robins et al., 2003).

El primer curso de programación en Ingeniería Civil en la UANDES se ha impartido desde los inicios de la carrera, a mediados de la década de 1990. El curso tuvo un enfoque pedagógico tradicional hasta el año 2011. Se comenzó enseñando el lenguaje de programación C, y años después Java. El sistema de evaluación se basaba en pruebas parciales y un examen final. Todas estas evaluaciones eran manuscritas, con lápiz y papel. Con los años se fueron introduciendo actividades en el laboratorio, aunque sin una mayor ponderación en la evaluación final.

A partir de 2012 el curso incorporó la enseñanza del lenguaje de programación Python 2, y adoptó una metodología de aprendizaje mezclado (*blended learning*, o *b-learning*; Porter et al., 2013). El curso se estructuró en *sesiones* (bloques temáticos con objetivos de aprendizaje definidos) con

duración de una a dos semanas, con dedicación de los estudiantes a actividades de aprendizaje dentro y fuera de la sala de clases. Es decir, los estudiantes debían estudiar para las clases el material del texto guía, resolviendo ejercicios del mismo texto, y viendo vídeos cortos publicados por profesores en YouTube, con síntesis de la teoría y desarrollo de ejemplos. Luego de estudiar en forma individual, y con anterioridad a la cátedra, los estudiantes debían someterse a una evaluación formativa en línea, con preguntas de selección múltiple. Luego, en la cátedra el profesor retroalimentaba a la clase basándose en los resultados de la evaluación en línea.

Tabla N°1. Cifras de aprobación y reprobación de estudiantes de la UANDES en el curso introductorio de programación.

Período	N Cohorte	Aprobados	Reprobados	% Aprobación	% Reprobación
2011-2	175	74	101	42,3%	57,7%
2012-2	168	66	102	39,3%	60,7%
2013-2	186	116	70	62,4%	37,6%
2014-2	202	109	93	54,0%	46,0%
2015-2	200	144	56	72,0%	28,0%
2016-1	263	166	97	63,1%	36,9%

En los años 2012 a 2014, parte importante de la evaluación del curso consistió en problemas que los estudiantes debían resolver durante el tiempo de clases, en forma individual o en parejas. Durante este período se constató que los estudiantes no demostraban mayor autonomía en su preparación para las clases. Sólo los mejores estudiantes leían el texto guía, estudiaban y resolvían problemas antes de la clase. Por ello, en 2015 y 2016 se reenfocó el propósito de la cátedra, dedicando los docentes mayor tiempo a revisar los contenidos básicos, realizar demostraciones de solución de problemas, y finalmente, permitir que los estudiantes resolvieran problemas de dificultad básica; algunos de ellos evaluados y con una baja ponderación en la calificación final del curso. La evaluación se realizó en forma semanal mediante resolución individual de problemas de programación en el laboratorio de computación. En 2015, las sesiones de laboratorio tenían una duración de 50 minutos, y los problemas eran de dificultad básica-intermedia. En 2016, hubo mayor disponibilidad de infraestructura de laboratorios, con lo que se pudieron planificar laboratorios evaluados semanales con duración cercana a dos horas. La exigencia de las evaluaciones en laboratorio del año 2016 fue mayor a la de 2015. En los años 2011 a 2016, el curso tuvo examen final, en el que los estudiantes debían obtener nota mínima 3,0 (en escala de 1,0 a 7,0) para aprobar el curso. Bajo ciertas condiciones, p.ej., nota de presentación cercana a 4,0, se permitía a los estudiantes rendir un examen recuperativo, que les permitía aprobar el curso con nota máxima 4,0.

La Tabla N°1 resume los resultados de aprobación y reprobación del curso introductorio de programación en la carrera de Ingeniería Civil de la UANDES en el período 2011-2016. Se incluyen los datos de los semestres en que el curso se impartió preferentemente para los estudiantes novicios de acuerdo a la malla curricular; se debe notar que el año 2016 se implementó un nuevo plan de estudios que reubicó el curso de programación en el primer semestre de la malla. En la Tabla N°1 es posible observar que las tasas de reprobación altas en el período 2011-2012, del orden de un 60%, han decaído, fluctuando en torno al 30% en el período 2015-2016. Atribuimos estos resultados a la simplificación curricular del curso (desde 2013 se ha excluido la enseñanza de programación orientada a objetos), a la mayor frecuencia y cantidad de evaluaciones que con anterioridad a la transformación del curso a aprendizaje mezclado.

Tabla N°2. Porcentaje de evaluaciones con nota mínima (1,0/7,0) en cada mes, y porcentaje de estudiantes con nota en examen final bajo el mínimo aprobatorio (3,0/7,0), en período 2013-2016.

Año/ Semestre	Tipo Evaluación	% de notas 1,0 en cada mes				Núm Eval.	Notas de ejercicios				% Examen < 3,0
							Final		Todas		
		1	2	3	4		M	DE	M	DE	
2013-2	Ejercicio en cátedra	7%	16%	23%	27%	14	4,31	0,92	4,25	2,03	38%
2014-2	Ejercicio en cátedra	15%	30%	38%	50%	13	3,85	1,24	3,47	2,08	43%
2015-2	Laboratorio (50 mins)	20%	20%	22%	19%	9	4,41	1,36	3,81	2,12	28%
2016-1	Laboratorio (110 mins)	18%	18%	36%	41%	8	4,21	1,59	3,58	2,09	22%

En la Tabla N°2 se puede apreciar que cada semestre hubo una cantidad creciente de estudiantes que decidió no asistir a clases y/o no rendir evaluaciones (y por ello obtuvo nota 1,0 en ellas). Por otro lado, en el examen, se puede observar que una parte considerable de los estudiantes llegó muy mal preparada y no logró la nota mínima necesaria para aprobar el curso.

Los contenidos y habilidades del curso de programación se caracterizan por su estructura secuencial, jerarquizada y acumulativa. A la luz de los antecedentes ya discutidos, intrepretamos que estas características del curso hacen que muchos estudiantes se vean desafiados al no lograr un ritmo de aprendizaje adecuado, lo que los lleva a perder la capacidad de asimilar nuevos conocimientos, y por ello deciden abandonar las actividades del curso prematuramente. Frente a este problema, los autores del presente estudio consideran que la modularización del curso de Programación permitirá mejorar las condiciones de aprendizaje y resultados del curso, por las siguientes razones:

- La modularización del curso permite a los estudiantes repetir el/los módulo(s) en que tengan dificultades, sin tener que repetir el curso completo en el semestre siguiente. Es decir, un estudiante puede retomar el curso en el semestre siguiente convalidando todos los módulos que haya aprobado hasta entonces.
- Dado que todos los módulos del curso se pueden repetir y paralelizar durante el semestre, los estudiantes podrán progresar por los módulos del curso a su propio ritmo. En consecuencia, los estudiantes tendrán una mayor motivación para perseverar en el sistema modular.

INTERVENCIÓN REALIZADA

El curso “Programación” de la UANDES fue transformado al formato modular con cuatro módulos sucesivos de tres semanas cada uno, y con la estructura ilustrada en la
Figura N°1. El total de 150 horas de dedicación del curso (5 créditos SCT), fue distribuido entre los módulos, con 30 horas para el primer módulo y 40 horas para los tres siguientes. La Tabla N°3 especifica los objetivos de aprendizaje de cada uno de los módulos del curso en forma resumida.

El equipo docente del curso incorporó a un profesor asistente de tiempo completo (autor principal del presente artículo), y otros cuatro profesores de jornada parcial. El cuerpo de ayudantes fue conformado por 12 ayudantes correctores, 5 ayudantes de tutoría, 6 ayudantes de laboratorio y

un ayudante coordinador. Todos los ayudantes eran alumnos de pregrado de ingeniería civil, de tercer año en adelante y mayoritariamente de la especialidad de Ciencias de la Computación.

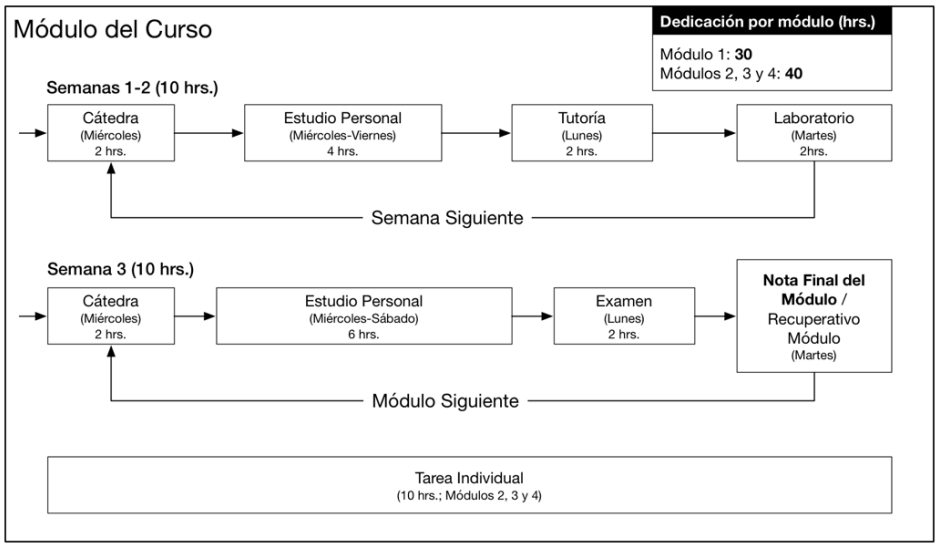

Figura N°1. Descripción de un módulo del curso Programación.

En el período académico semestral, el cual incluye 15 semanas de clases, fue posible planificar cinco períodos de tres semanas cada uno, a fin de implementar el curso en formato modular. Dentro de este espacio de cinco períodos en el semestre, los alumnos podían repetir a lo sumo uno de los cuatro módulos sin reprobar el curso completo. A los alumnos que terminaron el semestre con únicamente el último módulo del curso pendiente, se les ofreció rendir dicho módulo durante el receso de invierno en el mes de julio. Sin embargo, estos alumnos sólo podían optar a aprobar el curso con nota máxima 4,0 (en escala del 1,0 al 7,0).

Tabla N°3. Objetivos de aprendizaje de cada módulo del curso Programación.

Módulo	**Objetivos de Aprendizaje (Al finalizar el módulo, el estudiante será capaz de...)**
I	1. Especificar algoritmos computacionales con flujo lineal, decisivo e iterativo con diagramas de flujo. 2. Escribir programas computacionales que procesen datos numéricos y de tipo *string* desde la entrada estándar, evalúen expresiones aritméticas y lógicas, realicen conversiones de tipos de datos y permitan la ejecución de código condicional.
II	1. Escribir programas computacionales, con código iterativo y con separación de código en funciones que reciban parámetros y retornen valores.
III	1. Escribir programas computacionales que permitan almacenar y acceder a datos en listas (arreglos) simples y multidimensionales, realicen manipulación de *strings*, y que puedan leer y escribir archivos de texto.
IV	1. Escribir programas computacionales que utilicen diccionarios para almacenar datos. 2. Escribir programas computacionales que permitan realizar operaciones de cálculo numérico multidimensional utilizando vectores y matrices. 3. Escribir programas que generen gráficos cartesianos, de barras e histogramas.

Cada módulo tenía tres cátedras de dos horas cada una en salas de clase ordinarias, cuatro horas de tutorías en el laboratorio de computación, cuatro horas de ejercicios evaluados en el

laboratorio de computación (30% de la nota de cada módulo), y dos horas para el examen final en el laboratorio de computación (70% de la nota final de cada módulo). Además, se consideraron 4 a 6 horas semanales para el estudio personal fuera de clases, utilizando el libro guía, apuntes de clases, ejercicios propuestos y vídeos didácticos. Finalmente, en los módulos 2 a 4 se planificó una dedicación de 10 horas para tareas individuales en un plazo de 10 a 12 días.

Las tutorías incluían una sección de revisión de conceptos mediante preguntas abiertas que los estudiantes debían responder individualmente, y luego tres a cuatro ejercicios de programación que los estudiantes podían incluso continuar resolviendo después de la sesión de laboratorio presencial. Es importante notar que a las tutorías realizadas en los laboratorios de computación se presentaban estudiantes de distintos módulos en cada horario, por lo que los ayudantes debían estar preparados para resolver dudas sobre distintos contenidos del curso y asistir a los alumnos en la resolución de conjuntos de problemas de módulos distintos.

El laboratorio evaluado al día siguiente de cada tutoría comprendía en un solo problema de programación estructurado en etapas sucesivas de complejidad creciente, el cual exigía a los estudiantes demostrar dominio de las competencias y conocimientos vistos durante la semana. El formato de los problemas fue inspirado en el de la Olimpiada Chilena de Informática (OCI, 2017), que a su vez se basa en el de la *International Olympiad in Informatics* (IOI, 2017). Los problemas bajo este formato se describen mediante un enunciado compuesto por las siguientes secciones: contexto y descripción general del problema, formatos de entrada y salida con ejemplos, y rúbrica de evaluación. Los enunciados de los laboratorios fueron en general extensos; el más corto tenía 297 palabras, el más largo 1497, y la media fue de 992 palabras.

La evaluación de problemas de laboratorio fue realizada en la mayoría de los casos con la plataforma Contest Management System (CMS; Maggiolo & Mascellani, 2012), la cual ha sido utilizada en competencias de programación antes mencionadas, y permite evaluar los programas de los estudiantes en forma automatizada contra conjuntos de casos de prueba predefinidos. Para esto, los programas de los estudiantes debían cumplir en forma muy estricta los formatos de entrada y salida descritos en los enunciados de los problemas, en caso contrario, los programas no obtenían puntaje. Con CMS los estudiantes podían obtener la evaluación inmediata de sus programas, y al mismo tiempo se podía enfocar el esfuerzo de los ayudantes correctores en la evaluación de los exámenes al final de cada módulo en plazos no superiores a 36 hrs. Este fue un cambio significativo en la logística del curso en comparación a los semestres anteriores, en que los ayudantes tomaban un tiempo de 14 a 21 días para evaluar un laboratorio.

Los exámenes constaban de uno o dos problemas en un tiempo de dos horas. Los problemas de examen tenían un nivel de exigencia similar al de los laboratorios, pero su orientación era evaluar en mayor amplitud, y en forma integrativa, las competencias y conocimientos que los estudiantes debían adquirir en el módulo. Al igual que los problemas de laboratorio, los exámenes tenían enunciados detallados y extensos. El examen más corto tuvo enunciado de 494 palabras, y el más largo 1320. La extensión media fue de 898 palabras.

Las tareas no eran requisito para aprobar los módulos del curso, sino que se incorporaban al promedio final en forma independiente (conjuntamente el 15% de la nota final), y únicamente si el promedio obtenido en ellas era superior al promedio de notas finales de los módulos. Algunos de los ejercicios de tutorías fueron evaluados y en conjunto tenían una ponderación de 10% en la nota final y bajo el mismo criterio que las tareas. Es decir, se consideraban en el promedio final del curso sólo si el promedio de dichos ejercicios era superior al promedio de los módulos.

RESULTADOS

La Figura N°2 ilustra el progreso de los estudiantes en cada uno de los seis períodos del curso, y la Tabla N°4 contiene información sobre la repitencia de los estudiantes en los cuatro módulos. El módulo con mayor repitencia relativa al total de estudiantes que lo cursó, fue el tercero (68,9%), seguido por el primero (57,9%), el segundo (47,1%), y el cuarto (35,9%). La tasa de aprobación del curso, de un 28,5%, fue la más baja registrada semestralmente durante el período 2011-2017. Sólo 15 estudiantes fueron capaces de aprobar el curso sin reprobar ningún módulo. En el otro extremo, hubo 7 estudiantes que no lograron aprobar el módulo 1 durante el semestre.

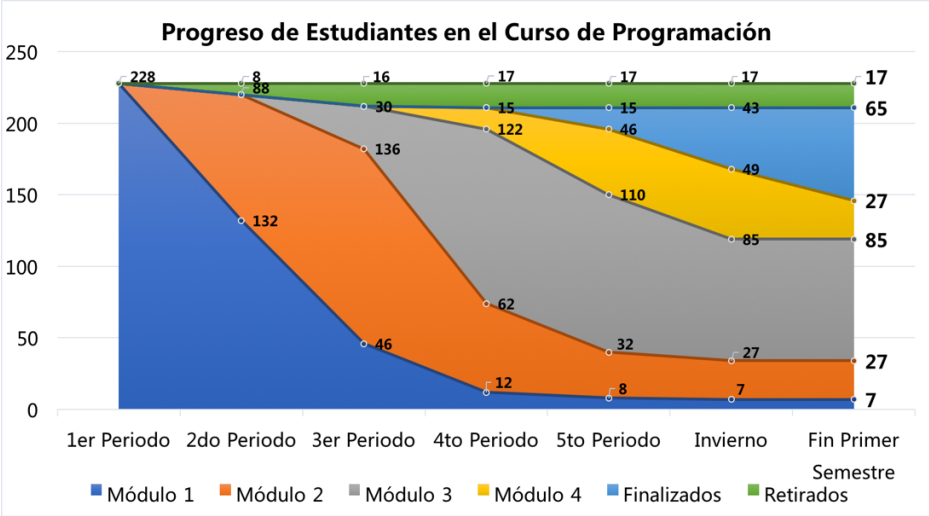

Figura N°2. Progreso de la cohorte del curso Programación durante el primer semestre (marzo-julio) de 2017.

Tabla N°4. Repitencia de módulos del curso Programación.

Módulo 1 (N=228)		
Repitencia (veces)	% de alumnos	Num. Alumnos
0	42,1%	96
1	37,7%	86
2	14,9%	34
3	1,8%	4
4	0,4%	1
5	3,1%	7

Módulo 2 (N=204)		
Repitencia (veces)	% de alumnos	Num. Alumnos
0	52,9%	108
1	32,4%	66
2	8,8%	18
3	4,9%	10
4	1,0%	2

Módulo 3 (N=177)		
Repitencia (veces)	% de alumnos	Num. Alumnos
0	31,1%	55
1	42,4%	75
2	26,0%	46
3	0,6%	1

Módulo 4 (N=92)		
Repitencia (veces)	% de alumnos	Num. Alumnos
0	64,1%	59
1	22,8%	21
2	13,0%	12

DISCUSIÓN

Varios autores destacan la importancia de las habilidades de lenguaje para la comprensión e interpretación correcta de los enunciados de problemas de programación, lo cual es condición necesaria para resolver los problemas en forma eficaz (Kuechler & Simkin, 2010). Además, los problemas de programación en ingeniería suponen un dominio básico de ciertas materias de matemática de nivel escolar. Por otro lado, estudios sobre modularización de cursos han enfatizado en la importancia de las habilidades metacognitivas y autorregulatorias de los estudiantes para ajustarse efectivamente a los ritmos exigentes de cursos modulares (Dochy et al., 1989). Con la finalidad de explorar diferencias en estas habilidades entre los estudiantes, previo a su ingreso a la universidad (y al curso de programación), comenzamos esta discusión presentando un análisis retrospectivo de los puntajes de la Prueba de Selección Universitaria (PSU) obtenidos por ellos (ver Tabla N°5 y Figura N°3). Consideramos a los 178 estudiantes que ingresaron a la carrera de ingeniería civil en 2017 por proceso de admisión ordinaria con PSU.

Tabla N°5. Media y desviación estándar de puntajes PSU de estudiantes del curso Programación.

		M			DE		
Grupo	**N**	**PSU Rank**	**PSU Mat**	**PSU Len**	**PSU Rank**	**PSU Mat**	**PSU Len**
Todos	178	649,7	699,9	603,7	77,3	43,7	63,3
Reprobados	120	641,5	691,9	595,5	74,0	37,7	58,8
Aprobados	58	666,8	716,5	620,8	81,7	50,4	69,2

Observamos diferencias significativas cercanas a 25 puntos en las medias de puntajes de PSU en ranking ($t(103,5)=2,00$, $p<0,05$), matemáticas ($t(88,8)=3,30$, $p<0,05$), y lenguaje ($t(98,0)=2,40$, $p<0,05$) entre estudiantes aprobados y reprobados, a favor de los primeros. Las diferencias en puntaje medio de ranking PSU sugieren que los estudiantes aprobados podrían contar con mejores estrategias metacognitivas que los reprobados (p.ej., mayor disciplina, dedicación a los estudios y perseverancia). También en la PSU de lenguaje la diferencia entre los dos grupos es bastante notoria a favor de los estudiantes que aprobaron, aunque ambos grupos muestran considerable variabilidad. Además, los puntajes de lenguaje son claramente más bajos que los de matemática, con diferencias en torno a los 100 puntos en todos los grupos (ver Tabla N°5). Finalmente, en la PSU de matemáticas se observan menores diferencias a favor de los estudiantes aprobados, y una menor variabilidad en la distribución de ambos grupos.

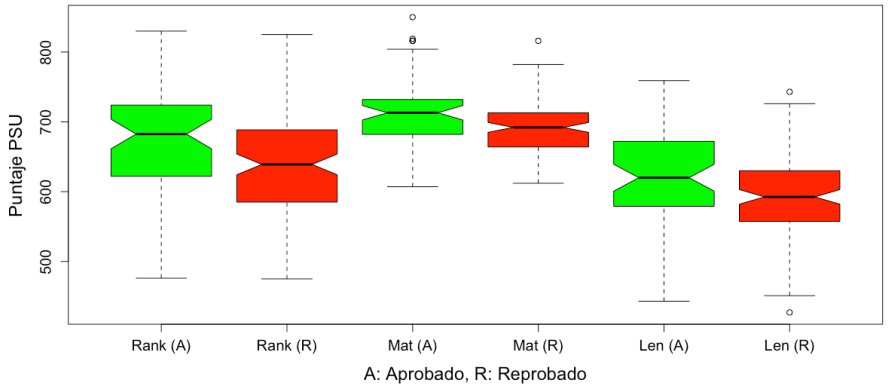

Figura N°3. Distribución de puntajes de PSU de estudiantes del curso Programación.

Con respecto al sistema de evaluación del curso, los alumnos manifestaron constantemente dos preocupaciones a los docentes y ayudantes, por vía de correos electrónicos y conversaciones en laboratorios y cátedras: dificultad para comprender los enunciados de las evaluaciones, y que en el curso había falta de materiales de estudio para preparar laboratorios y exámenes. Con respecto a la primera preocupación, los problemas en las evaluaciones fueron formulados considerando una contextualización en el mundo real. Por ejemplo, en un banco, en una clínica, o en algún proceso productivo. La descripción del contexto, las reglas del problema, y las instrucciones a seguir requerían descripciones que podían extenderse por más de una página. Además, en el caso de los laboratorios, dado que el formato de problema requería describir los formatos de entrada y salida, y rúbricas de puntaje, los enunciados se alargaban hasta cuatro páginas. Muchos estudiantes tenían dificultad para leer y comprender los enunciados con la suficiente atención a los detalles. Esto se vio reflejado en las evaluaciones, con estudiantes que consultaban dudas a docentes y ayudantes sobre aspectos que se encontraban descritos explícitamente en los enunciados. Consideramos que estas dificultades podrían subsanarse mediante la adopción de formatos de redacción con lenguaje simplificado y mayor concisión.

En relación a la segunda preocupación de los estudiantes, durante el semestre se realizaron exámenes y ejercicios de laboratorio que consistían en una variación menor de las evaluaciones realizadas en períodos anteriores (y publicadas en el sitio web del curso). En dichas evaluaciones los estudiantes obtenían en promedio malos resultados. Esto revela que los estudiantes no aprovechaban las evaluaciones de períodos anteriores como material de estudio, posiblemente porque se veían sobrepasados por la dificultad de los problemas. Consideramos que podemos responder a este problema con medidas tales como disponer de compendios de problemas graduados por dificultad, impartir sesiones dedicadas a la enseñanza de estrategias efectivas de resolución de problemas, la disponibilidad permanente de la plataforma CMS como medio de ejercitación con retroalimentación instantánea, y la dedicación de los estudiantes a resolver problemas en forma colaborativa durante las tutorías.

Si bien la tasa de aprobación del curso modular fue menor que las registradas en el período 2011-2016, estimamos que los efectos del modelo modular podrían verificarse analizando los resultados sobre una base anual. Consideramos que la mayoría de los estudiantes que reprobaron en el curso modular tendrán la posibilidad de aprobarlo en el semestre siguiente, con uno o dos meses de dedicación adicional. Estos serían 112 estudiantes, o 49% de la cohorte, con lo que la tasa de aprobación total podría acercarse al 70% en el mes de octubre. Al cierre del año académico, la tasa de aprobación total podría superar el 90%.

CONCLUSIONES

Conforme a la filosofía y las prácticas de la investigación de diseño educacional (The Design-Based Research Collective, 2003), las innovaciones pedagógicas pueden requerir múltiples iteraciones con variados objetivos, revisiones y ajustes, a fin de determinar los principios de diseño que se adecúan mejor a las necesidades de estudiantes y docentes. En el caso de nuestro curso de programación, si bien los objetivos relativos a la mejora en la tasa de aprobación semestral del curso no han sido logrados en forma inmediata, hemos validado la practicabilidad del enfoque, y su aceptación tanto por parte de los estudiantes como los docentes. Tempranamente, hemos constatado dos fortalezas del sistema: como efecto colateral de las múltiples instancias de evaluación reprobatorias (exámenes) en el semestre, los estudiantes se ven forzados a estudiar constantemente para adquirir las competencias de cada módulo del curso con la debida profundidad. Además, nos es posible evaluar el progreso de los estudiantes con frecuencia y rigurosidad, y detectar los contenidos y competencias en donde tienen las mayores

dificultades de aprendizaje. Esto nos abre posibilidades para realizar intervenciones focalizadas en la docencia, en el trabajo de ejercitación de competencias, y en la construcción de evaluaciones, con el propósito de mejorar los resultados del curso en forma iterativa y progresiva durante los próximos períodos académicos.

AGRADECIMIENTOS

Este trabajo fue parcialmente financiado por el proyecto CONICYT-FI 11160211.

REFERENCIAS BIBLIOGRÁFICAS

Allen, D.W. (1971). The modular instructional unit, a new approach. Stanford School Scheduling System. Paper presentado en AERA, 1971.

Bickerstaff, S., Fay, M. P., & Trimble, M. J. (2016). Modularization in developmental mathematics in two states: Implementation and early outcomes. *(Working Paper No. 87)*.

Cornford, I. R. (1997). Ensuring effective learning from modular courses: a cognitive. *Journal of Vocational Education & Training*, *49*(2), 237–251. http://doi.org/10.1080/13636829700200014

Creager & Murray, D.L. (1971). The use of modules in college Biology teaching. Washington: Commission on Undergraduate Education in the Biological Sciences, The American Institute of Biological Sciences, March 1971.

Dochy, F. J., Wagemans, L. J.., & De Wolf, H. C. (1989). Modularisation and student learning in modular instruction in relation with prior knowledge. Open Univ., Heerlen (Netherlands). Centre for Educational Technological Innovation.

Gaete, M., & Morales, R. (2011). Articulación del sistema de educación superior en Chile: posibilidades, tensiones y desafíos. *Calidad en la educación*, (35), 51-89.

Goldschmid, B., & Goldschmid, M. L. (1973). Modular instruction in higher education: A review. *Higher Education*, *2*(1), 15–32. http://doi.org/10.1007/BF00162534

Kuechler, W. L., & Simkin, M. G. (2010). Why Is Performance on Multiple-Choice Tests and Constructed-Response Tests Not More Closely Related? Theory and an Empirical Test. *Decision Sciences Journal of Innovative Education*, *8*(1), 55–73.

Maggiolo, S., & Mascellani, G. (2012). Introducing CMS: A Contest Management System. *Olympiads in Informatics*, *6*.

OCI (2017). Olimpiada Chilena de Informática. Recuperado de htttp://www.olimpiada-informatica.cl

IOI (2017). International Olympiad in Informatics. Recuperado de http://www.ioinformatics.org

Porter, W. W., Graham, C. R., Spring, K. a., & Welch, K. R. (2014). Blended learning in higher education: Institutional adoption and implementation. *Computers and Education*, *75*, 185–195.

Robins, A., Rountree, J., Rountree, N., Robins, A., Rountree, J., & Rountree, N. (2016). Learning and Teaching Programming : A Review and Discussion Learning and Teaching Programming : A Review, *3408*(April). http://doi.org/10.1076/csed.13.2.137.14200

SIES (2014). Panorama de la Educación Superior en Chile 2014. *División de Educación Superior*, Ministerio de Educación.

The Design-Based Research Collective. (2003). Design-based research: An emerging paradigm for educational inquiry. *Educational Researcher*, 5-8.

UTFSM (2017). Historia. Recuperado de http://www2.elo.utfsm.cl/generalidades/historia.htm