Ishvel: un Framework para la Elaboración de Tareas en Cursos Introductorios de Programación

Gonzalo Fernández C.
Departamento de Informática
Universidad Técnica Federico Santa María
Santiago, Chile
gonzalo.fernandezc@sansano.usm.cl

***Resumen*—Las tareas son parte importante de un curso introductorio de programación, ya que miden el aprendizaje del estudiante, mientras pone en práctica el conocimiento adquirido. Un curso introductorio requiere elaborar varias tareas por semestre, y se debe velar porque estas sean de calidad, ya que impactan enormemente en el interés del estudiante por aprender a programar. Es por esto que se propone Ishvel, un framework que, de manera metodológica, le facilite a los profesores el proceso de elaboración de tareas en cursos introductorios de programación, mientras garantiza la calidad de las mismas.**

***Palabras Clave*—elaborar tareas, framework, curso introductorio de programación, educación**

I. Objetivos

Objetivo General

- Desarrollar un Framework [1] para la elaboración de tareas en cursos introductorios de programación

Objetivos Específicos

- Identificar y definir criterios que puedan ser usados para evaluar una tarea.
- Definir una métrica que permita evaluar una tarea de acuerdo al cumplimiento de criterios.
- Definir una metodología para la elaboración de tareas utilizando el framework Ishvel.
- Validar cómo la aplicación de la metodología definida elabora mejores tareas.

II. Definición Inicial del Problema

Los cursos introductorios de programación son, en general, el primer acercamiento de un estudiante al mundo de la programación, y por lo tanto, se debe velar porque se desarrollen de la manera más prolija posible, considerando entre otras cosas, la elaboración y uso de tareas de calidad. Las tareas son parte importante del proceso de aprendizaje y enseñanza [2], pues permiten medir el aprendizaje de los estudiantes y proporcionarles una retroalimentación significativa [4]. Además, en los cursos introductorios de programación, son la principal actividad en donde los estudiantes ponen en práctica lo que han aprendido sobre programación, lo que hace que las tareas jueguen un rol importante en su interés por aprender, pudiendo llevar al estudiante tanto a querer sobresalir resolviendo las evaluaciones del curso, como a desertar debido a su percepción de la programación por tareas de mala calidad [10, 11], entre otros efectos como los mencionados en el Árbol del Problema (ver Figura 1).

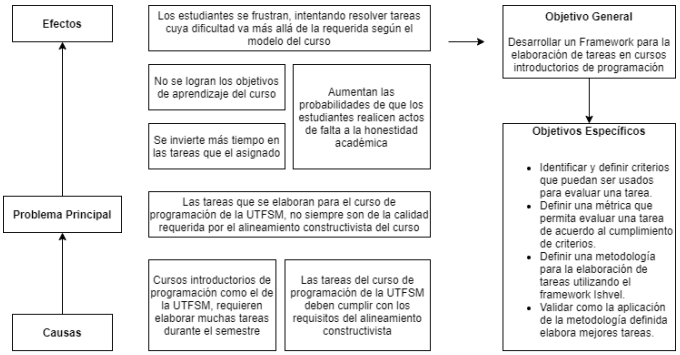

Figura 1. Árbol del Problema. Fuente: Elaboración propia.

En la Universidad Técnica Federico Santa María (UTFSM), se dicta un curso introductorio de programación a lo largo de todos sus campus. Este recibe de forma masiva a todos los estudiantes de primer año de ingeniería, y se dicta de igual forma independiente del campus o carrera del estudiante. El curso, que se lleva a cabo de forma online, está dividido en Unidades Virtuales de Aprendizaje (UVA), las cuales estructuran las actividades de cada semana (ver Figura 2), y su diseño se basa en el alineamiento constructivista [1]. Este último plantea la necesidad de que las evaluaciones del curso reflejen los objetivos de aprendizaje del mismo y, que a su vez, sean estos los que definan las actividades y tareas a realizar. Esto brinda a las tareas un rol tanto formativo como sumativo de evaluar, y requiere que estas cumplan ciertos estándares para justificar que cumplen con los objetivos de aprendizaje [1].

[1]Marco de trabajo que plantea una metodología validada y un conjunto de herramientas, las cuales están diseñadas para la correcta aplicación de dicha metodología

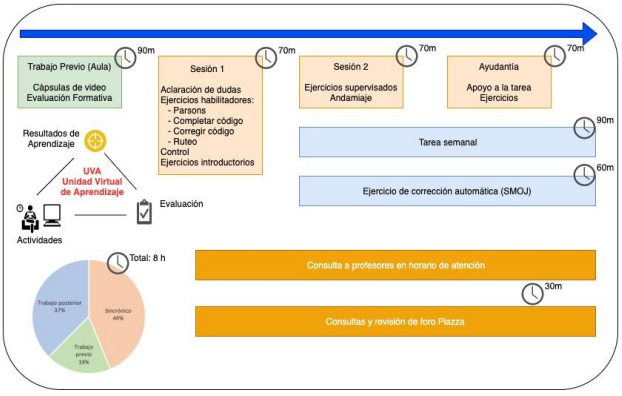

Figura 2. Modelo Formativo IWI-131. Fuente: Reglas del curso de programación de la UTFSM (2021-2).

Sin embargo, pese a que el curso cuenta con una gran cantidad de profesores, no son más de 5 los que se encargan de proponer tareas a la coordinación. Y, dado que cada semana se requiere una nueva tarea para la UVA en curso, se vuelve complicado elaborar tareas de calidad a tiempo, que cumplan con el alineamiento constructivista del curso, y que cuya resolución no requiera más tiempo que el estipulado en la UVA. Además, la elaboración de tareas se lleva a cabo sin métricas que permitan medir la calidad de cada una, o una metodología de elaboración de tareas que garantice la calidad de las mismas en el tiempo.

Entre los problemas que puede conllevar el uso de tareas que no son de calidad, se encuentran:

- Pérdida del interés en aprender a programar [7], de modo tal que el estudiante deje de ver el ramo como un curso de aprendizaje, y su actitud hacia él sea nétamente para aprobar.
- Frustración a lo largo del ramo, al punto en que el estudiante puede decidir desertar del curso y de la programación en general [11].
- Aumentar las probabilidades de que los estudiantes realicen actos que falten a la honestidad académica para resolverla [8].
- Complicar la elaboración de la rúbrica con la cual la tarea será evaluada, lo que puede conllevar a entregar un feedback menos valioso al estudiante

Es por esto que se requiere de un marco de trabajo que permita elaborar tareas de calidad, el cual brinde tanto una metodología como herramientas que faciliten a los profesores esta labor, ahorrándoles tiempo y garantizando a los estudiantes del curso una mejor experiencia de aprendizaje.

III. Discusión Bibliográfica Preliminar

Diversos autores han abordado la elaboración y medición de tareas de calidad, enfocándose tanto en factores emocionales del estudiante [6, 7, 10, 11], como en aspectos particulares de cualquier tarea en sí [2, 5, 9]. Y en general, siempre se destacan 3 aspectos principales:

- Las tareas deben tener alguna aplicación real, o bien, resolver un problema real que le dé sentido al tiempo que el estudiante invertirá en resolverla.
- Las tareas deben ser interesantes, un problema puede expresarse utilizando un contexto de la actualidad, de lo que los estudiantes en general considerarían interesante como lo son sus bandas, juegos, tendencias, etc.
- Las tareas deben tener un nivel de dificultad adecuado, ni muy difíciles como para frustrar al estudiante, ni muy fáciles como para no cumplir con los objetivos de aprendizaje de la misma.

Por lo tanto, lo que se busca con el framework a desarrollar, es establecer un marco de trabajo que garantice elaborar una tarea que cumpla con, al menos, aquellos 3 aspectos. Un primer enfoque sería que el framework elaborara automáticamente la tarea, sin embargo, proyectos como Moulinog [12] han sido muy limitados en lograr esto, pues están limitados a, en base a 1 tarea, generar múltiples tareas que son en esencia lo mismo con la salvedad de cambiar ciertos valores y palabras, que no garantizan que 1 misma solución no sea esencialmente capaz de resolver 2 tareas distintas de las generadas.

Sin embargo, la idea de tener tareas base motiva a investigar acerca de si es buena idea re-utilizar tareas, generando un pequeño banco de tareas de calidad, pero teniendo en cuenta que éstas deben ser del tipo *Tweakable assignments* [3], es decir, la mayoría de sus partes pueden re-utilizarse, a excepción de ciertas partes las cuales, en base a ciertas especificaciones, deben modificarse en la nueva tarea de modo tal que una solución a la tarea original, simplemente no funcione con la tarea nueva que se está elaborando, manteniendo así la calidad de la tarea original, y evitando llegar a aumentar las probabilidades de actos fraudulentos por parte de los estudiantes, quienes pueden hacerse de las soluciones de tareas anteriores [8].

Dicho esto, la forma en que el framework debiese funcionar, es primero presentando al profesor la metodología con la cual elaborar la tarea, y segundo, brindarle una herramienta que le permita al profesor, tanto observar métricas de utilidad a medida que redacta, como una validación de que los objetivos de aprendizaje esperados para esa tarea, se cumplan en base a diversas validaciones, como lo sería la revisión del código que resuelve el problema, entre otros.

El framework además de definir métricas que pueda medir de forma automátizada sobre cualquier tarea, también debe ofrecer de forma sencilla una manera de que la tarea, una vez haya sido aplicada, sea evaluada por otros actores. Existen diversos componentes que pueden apoyar en esta evaluación posterior, cada uno con diversos porcentajes de desempeño en base a la cantidad de datos de medición que pueden aportar [4]. Ésta obtención de retroalimentación sobre la tarea se puede obtener mediante formularios auto-generados por el framework, así como su posterior análisis debe ser condensado de manera automática para así estar al tanto de las fortalezas y debilidades de la tarea, según la percepción de estudiantes y profesores, principalmente en la resolución de la misma por

parte del estudiante, como en su elaboración por parte del profesor.

IV. Plan de Trabajo

Actividad	Semanas
Elaboración del Estado del Arte	2
Recopilar criterios que permitan evaluar la calidad de una tarea	1
Definir métricas que permitan evaluar la calidad de una tarea	1
Definir los casos de uso del framework Ishvel	1
Analizar todas las tareas disponibles del curso de programación UTFSM	1
Elaborar y probar prototipos del framework	1
Documentar la metodología del framework Ishvel	1
Implementación del framework para elaborar las tareas de programación	8
Análisis de resultados	2
Elaboración del documento de memoria	3
Correcciones finales	1
Total	**22**

Tabla I: Actividades del Plan de Trabajo. Fuente: Elaboración propia

V. Tiempos SCT

Actividad	Tiempo [hrs]
Planificación	2
Búsqueda de información	3
Análisis	15
Desarrollo	10
Edición	1
Desarrollo del video	3
Total	**39**

Tabla II: Tabla de Tiempos SCT.

VI. Video

Youtube: https://youtu.be/4SkeVD_pvVw

Referencias

[1] John Biggs. *Teaching for Quality Learning at University*. Ene. de 2003.

[2] Allison Boye. *How Do I Create Meaningful and Effective Assignments?* https://www.depts.ttu.edu/tlpdc/Resources/Teaching_resources/TLPDC_teaching_resources/CreatingEffectiveAssignments.php. [Online; consultado el 19 de julio del 2021 a las 23:00hrs]. 2020.

[3] Michelle Craig y Briana B. Morrison. "EngageCSEdu¡br¿¡br¿Assignments: To Re-Use or Not Re-Use? That is the Question." En: *ACM Inroads* 12.3 (ago. de 2021), págs. 18-20. ISSN: 2153-2184. DOI: 10.1145/3477429. URL: https://doi.org/10.1145/3477429.

[4] National Academy of Engineering. *Developing Metrics for Assessing Engineering Instruction: What Gets Measured Is What Gets Improved*. Washington, DC: The National Academies Press, 2009. ISBN: 978-0-309-13782-9. DOI: 10.17226/12636. URL: https://www.nap.edu/catalog/12636/developing-metrics-for-assessing-engineering-instruction-what-gets-measured-is.

[5] Christopher D. Hundhausen, Adam S. Carter y Olusola Adesope. "Supporting Programming Assignments with Activity Streams: An Empirical Study". En: *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. SIGCSE '15. Kansas City, Missouri, USA: Association for Computing Machinery, 2015, págs. 320-325. ISBN: 9781450329668. DOI: 10.1145/2676723.2677276. URL: https://doi.org/10.1145/2676723.2677276.

[6] Paivi Kinnunen y Beth Simon. "Experiencing Programming Assignments in CS1: The Emotional Toll". En: *Proceedings of the Sixth International Workshop on Computing Education Research*. ICER 10. Aarhus, Denmark: Association for Computing Machinery, 2010, págs. 77-86. ISBN: 9781450302579. DOI: 10.1145/1839594.1839609. URL: https://doi-org.usm.idm.oclc.org/10.1145/1839594.1839609.

[7] Lucas Layman, Laurie Williams y Kelli Slaten. "Note to Self: Make Assignments Meaningful". En: *SIGCSE Bull.* 39.1 (mar. de 2007), págs. 459-463. ISSN: 0097-8418. DOI: 10.1145/1227504.1227466. URL: https://doi-org.usm.idm.oclc.org/10.1145/1227504.1227466.

[8] Simon. "Designing Programming Assignments to Reduce the Likelihood of Cheating". En: *Proceedings of the Nineteenth Australasian Computing Education Conference*. ACE '17. Geelong, VIC, Australia: Association for Computing Machinery, 2017, págs. 42-47. ISBN: 9781450348232. DOI: 10.1145/3013499.3013507. URL: https://doi.org/10.1145/3013499.3013507.

[9] Daniel E. Stevenson y Paul J. Wagner. "Developing Real-World Programming Assignments for CS1". En: *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*. ITICSE '06. Bologna, Italy: Association for Computing Machinery, 2006, págs. 158-162. ISBN: 1595930558. DOI: 10.1145/1140124.1140167. URL: https://doi.org/10.1145/1140124.1140167.

[10] Lisa Torrey. "Student Interest and Choice in Programming Assignments". En: *J. Comput. Sci. Coll.* 26.6 (jun. de 2011), págs. 110-116. ISSN: 1937-4771.

[11] Rebecca Vivian, Katrina Falkner y Nickolas Falkner. "Computer Science Students' Causal Attributions for Successful and Unsuccessful Outcomes in Programming Assignments". En: *Proceedings of the 13th Koli Calling International Conference on Computing Education Research*. Koli Calling '13. Koli, Finland: Association for Computing Machinery, 2013, págs. 125-134. ISBN: 9781450324823. DOI: 10.1145/2526968.2526982. URL: https://doi.org/10.1145/2526968.2526982.

[12] Gonzague Yernaux, Wim Vanhoof y Laurent Schumacher. "Moulinog: A Generator of Random Student Assignments Written in Prolog". En: *Proceedings of the 22nd International Symposium on Principles and Practice of Declarative Programming*. PPDP '20. Bologna, Italy: Association for Computing Machinery,

2020. ISBN: 9781450388214. DOI: 10.1145/3414080.3414100. URL: https://doi.org/10.1145/3414080.3414100.