Tabla 4. Plantilla para la especificación de requisitos

Especificación de Requerimientos Funcionales (ERS)	
Identificación del requisito	
Nombre del requisito	
Componente	
Característica asociada	
Descripción del requisito:	
Características	
Prioridad	☐ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional
Restricciones	
Interacción humano - tecnología	☐ SÍ ☐ NO
Interacción tecnología - tecnología	☐ SÍ ☐ NO

Fuente: Elaboración propia.

Tabla 5. Ejemplo de definición del entorno

FASES PARA LA DEFINICIÓN DE REQUISITOS	
FASE I	**Definición del entorno**
Un parqueadero es un espacio público/privado destinado a la prestación del servicio de estacionamiento y cuidado de vehículos (automóviles y motos); el servicio puede ser gratuito o con fines comerciales mediante arrendamiento. Los parqueaderos poseen un horario definido de prestación del servicio, y se rigen por normas de tránsito y señalización. En el momento en que el usuario desee solicitar el servicio de parqueadero, el empleado de estacionamiento verifica la disponibilidad de espacios, si existen espacios disponibles se le solicita al usuario la información pertinente que le identifique como dueño del vehículo y se le asigna el espacio para ocupar. Cuando el usuario retira el vehículo del parqueadero, se le estima el tiempo de permanencia y de acuerdo a este se le cobra una tarifa ya establecida. En caso de estacionar su vehículo de manera permanente, el usuario puede acordar contratos con el parqueadero mediante el cobro de mensualidades.	
FASE II	**Descripción del entorno tecnológico actual**
Hardware	
Computador, cámaras de vigilancia, monitores de vídeo, DVR (grabador de vídeo digital), rótulos indicativos para mostrar la cantidad de espacios disponibles.	
Software	
Aplicativo de escritorio para administración del parqueadero, software del sistema de circuito cerrado de televisión, Office, navegadores web, software contable.	

FASES PARA LA DEFINICIÓN DE REQUISITOS	
FASE III	**Establecer las necesidades del entorno**
Se busca controlar el ingreso y la salida del vehículo (moto, automóvil) del parqueadero mediante el reconocimiento y almacenamiento de la placa en el sistema. Cuando el vehículo abandone el parqueadero es necesario calcular su tiempo de estadía, como también, detectar la presencia de los vehículos en los puestos de estacionamiento mostrando información en tiempo real de cuáles puestos están ocupados y cuáles no y cuántos son. También se pretende un control energético en el parqueadero.	
FASE IV	**Descripción de la funcionalidad para implementar en el entorno**
Sistema automático que permita controlar el ingreso y la salida del vehículo (moto, automóvil) del parqueadero, al momento de su ingreso el sistema debe reconocer y guardar la placa del vehículo al acercarse al parqueadero; toma el registro de entrada y, en caso de abandonar el parqueadero, el registro de salida, lo que permite calcular el tiempo de estadía y generar el valor de estacionamiento de acuerdo a ese tiempo. El sistema debe detectar la presencia de vehículos en un área específica y mostrar la información del área, como la información en tiempo real de cuáles y cuántos puestos de estacionamiento están ocupados, con la respectiva placa del vehículo y el tiempo de estadía, e identificar los puestos que están desocupados. El sistema debe emitir alertas en caso de detectar presencia de humo (incendios), y mostrar el lugar en pantalla donde ocurra el siniestro y activar automáticamente el sistema de extinción de incendios. De igual manera, debe emitir avisos inmediatos en caso de que se detecten fugas de agua o inundaciones, y mostrar el lugar en pantalla donde ocurra el suceso. Facilitar el ahorro de energía permitiendo que las lámparas se enciendan de noche y se apaguen de día de manera automática, y que se enciendan cuando alguien entre al parqueadero y se apaguen cuando salga.	

Fuente: Elaboración propia.

Tabla 6. Ejemplo de especificación de requisitos

FASE V. ERS – Especificación de Requerimientos Funcionales	
Id. del requisito	RF01
Nombre del Requisito	Control de ingreso y salida del vehículo
Componente	Sensor de parking, cámaras de vigilancia
Característica asociada	Embebido, invisible, ubicua
Descripción del requisito	Controlar el ingreso del vehículo en el parqueadero y su salida de éste mediante el reconocimiento de la placa
Características	Los usuarios deberán acercar su vehículo a un punto donde el sistema pueda reconocer y detectar la placa, con el fin de tomar el registro correspondiente y permitir el ingreso y su salida cuando el usuario desee abandonar el parqueadero
Prioridad	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/Opcional
Restricciones	• El sistema estará diseñado para controlar el ingreso y la salida de automóviles y motos, mas no de bicicletas u otros medios de transporte.
Interacción humano - tecnología	SÍ ☐ NO ☒
Interacción tecnología - tecnología	SÍ ☒ NO ☐

FASE V. ERS – Especificación de Requerimientos Funcionales	
Id. del requisito	RF02
Nombre del Requisito	Verificación de disponibilidad
Componente	Sensores de ultrasonido, módulo de comunicaciones, PC, microcontrolador central, módulo Ethernet, LED (indica la disponibilidad del espacio), sistema de guía de parqueo PGS, rótulos indicativos para mostrar a los usuarios la cantidad de espacios disponibles y la dirección de dicho espacio
Característica asociada	Embebido, invisible, ubicua
Descripción del requisito	Informar cuando un puesto de estacionamiento esté ocupado o desocupado
Características	El sistema podrá verificar la disponibilidad de los puestos de estacionamiento mediante rótulos indicativos, que le muestran al usuario los puestos de estacionamiento que estén disponibles y la dirección en que se encuentran dichos puestos; adicional a esto, cada uno de los puestos tiene un bombillo LED que advierte su disponibilidad: Rojo: ocupado; Verde: disponible
Prioridad	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/Opcional
Restricciones	• En caso de que haya más de un puesto de estacionamiento disponible, el usuario podrá seleccionar el puesto que desee, mas el sistema no lo asignará automáticamente
Interacción humano - tecnología	SÍ ☐ NO ☒
Interacción tecnología - tecnología	SÍ ☒ NO ☐

Fuente: Elaboración propia.

IV. CONCLUSIONES

En este capítulo se hace un acercamiento conceptual al término de *ciudad inteligente* desde varios autores, donde se da a entender que existe un factor en común: son ciudades que interactúan con las Tecnologías de la Información y la Comunicación (TIC).

Se logra evidenciar, por medio de una revisión de literatura, que no existen metodologías propias para la especificación de requisitos de software para aplicaciones en ciudades inteligentes.

La guía técnica propuesta incorpora un conjunto de elementos y características propios de las aplicaciones para ciudades inteligentes, que no se contemplan en metodologías de ingeniería de requisitos actuales.

La guía técnica permite realizar una contextualización general del dominio donde se va a desarrollar la aplicación y luego detallar cada uno de los requisitos que le darán la funcionalidad al sistema.

La guía técnica propuesta busca reconocer el entorno tecnológico actual del dominio, así como validar la interacción (humano - tecnología y tecnología - tecnología) que implica la funcionalidad del requisito.

REFERENCIAS

[1] R. P. Dameri and C. Rosenthal-Sabroux, Eds., *Smart City. How to Create Public and Economic Value with High Technology in Urban Space.* Springer International Publishing Switzerland, 2014.

[2] Fundación Universia, "Smart Cities: estas son las ciudades más tecnológicas del mundo", España, 11 de agosto 2017. [En línea]. Disponible en: http://noticias.universia.es/ciencia-tecnologia/noticia/2017/08/11/1154968/smart-cities-ciudades-tecnologicas-mundo.html

[3] E. Ontiveros, D. Vizcaíno y V. López Sabater, *Las ciudades del futuro: inteligentes, digitales y sostenibles.* Madrid: Ed. Ariel, Fundación Telefónica, 2016.

[4] A. Preukschat, Smart cities. 2018. Disponible en: http://libroblockchain.com/smart-cities/.

[5] M. Bouskela, M. Casseb, S. Bassi, C. De Luca y M. Facchina, *La ruta hacia las Smart cities: Migrando de una gestión tradicional a la ciudad inteligente.* Inter-American Development Bank, 2016.

[6] M. Swatha and K. Pooja, "Smart car parking with monitoring system", in *IEEE International Conference on System, Computation, Automation and Networking (ICSCA 2018),* pp. 1-5.

[7] C. Izquierdo Martínez, "Aplicación de la IoT al ámbito del transporte. Auto-gestión del tráfico de vehículos inteligentes", tesis de doctorado, Universitat Politècnica de València, 2017.

[8] R. Corvalán, N. Sanabria, E. Ferrari, V. Titiosky, A. Amarilla, A. Cuevas, V. Sabaj y H. Fleitas, "Aplicación de criterios de optimización energética y seguridad en la iluminación y confort, en calles y avenidas", *Extensionismo, Innovación y Transferencia Tecnológica. Claves para el desarrollo,* vol. 5, pp. 177-186, 2019.

[9] D. S. Lawrence, N. G. La Vigne, M. Go and P. S. Thompson, "Lessons learned implementing Gunshot Detection Technology: Results of a process evaluation in three major cities", *Justice Eval. J.*, No. 1, pp. 109-129, 2018.

[10] M. Penza, D. Suriano, M. Villani, L. Spinelle and M. Gerboles, "Towards air quality indices in smart cities by calibrated low-cost sensors applied to networks", in *Proceedings of the IEEE Sensors 2014,* Valencia, Spain, 2-5 November 2014, pp. 2012-2017.

[11] L. Benítez y M. Ortega, "Las TIC y la gestión de los desafíos de sostenibilidad energética de las ciudades inteligentes, *Economía Industrial,* N.º 395, pp. 87-94, 2015.

[12] A. Guzman, A. Martínez, F. V. Agudelo, H. Estrada-Esquivel, J. P. Ortega and J. Ortiz, "A methodology for modeling ambient intelligence applications using i* framework", in *Proceedings of the Ninth International i* Workshop co-located with 24th International Conference on Requirements Engineering (RE 2016),* L. López y Y. Yu, eds., vol. 1674 of CEUR Workshop Proceedings, Beijing, China, September 12-13, 2016, CEUR-WS.org, pp. 61-66.

[13] K. Vlaanderen, S. Jansen, S. Brinkkemper and E. Jaspers, "The agile requirements refinery: Applying SCRUM principles to software product management", *Information and Software Technology,* vol. 53, pp. 58-70, 2011.

[14] M. G. Báez y S. I. Barba Brunner, "Metodología DoRCU para la Ingeniería de requerimientos", in *Workshop in Engineering of Requirements (WER).* 2001, pp. 210-222.

[15] L. González y G. Urrego, "Modelo de contexto y de dominio para la ingeniería de requisitos de sistemas ubicuos", *Revista Ingenierías Universidad de Medellín,* vol. 9, N.º 17, pp. 151-167, 2010.

[16] C. Evans, L. Brodie and J. Augusto, "Requirements Engineering for Intelligent Environments", in *Proceedings of the 10th International Conference on Intelligent Environments (IE'14),* 2014, pp. 154-161. IEEE Press. Disponible en: https://www.researchgate.net/publication/286679790_Requirements_Engineering_for_Intelligent_Environments

Capítulo III
Pruebas Automáticas para Evaluar Cursos de Programación de Computadores

Milton de Jesús Vera Contreras - miltonjesusvc@ufps.edu.co
Universidad Francisco de Paula Santander (UFPS) Cúcuta
Matías Herrera Cáceres - matiashc@ufps.edu.co
Docente-investigador, Universidad Francisco de Paula Santander (UFPS) Cúcuta
Óscar Alberto Gallardo Pére - oscargallardo@ufps.edu.co
Docente-directivo, Universidad Francisco de Paula Santander (UFPS) Cúcuta

I. INTRODUCCIÓN

En el año 2011, Marc Andreessen pronunció la popular frase "el software se está comiendo el mundo" (en inglés, "software is eating the world") [1], y desde entonces es una realidad que no se ha detenido, pues el software es el núcleo de las tecnologías de la información y la comunicación (TIC) que hoy protagonizan la cuarta revolución industrial o Industria 4.0 [2]. En contraste, en el contexto colombiano y latinoamericano hay un déficit y existen debilidades en materia de talento digital [3], [4], por lo que cobra especial importancia la formación en programación de computadores [5], un área en la que enfocan sus esfuerzos los empresarios y el gobierno [6].

Ahora bien, históricamente la formación en programación de computadores ha sido un problema educativo complejo [6]-[13]. Por lo tanto, al considerar el panorama descrito sobre el talento digital, es de gran interés y alto impacto cualquier esfuerzo por facilitar, agilizar y mejorar la educación en esa área. En ese orden de ideas, el programa de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander de Cúcuta (UFPS), en su ejercicio semestral de autoevaluación, realizó en el año 2017 un análisis detallado de las calificaciones en las asignaturas de Programación de Computadores. Los dos principales hallazgos de dicho análisis fueron el alto nivel de reprobación y las bajas calificaciones de los estudiantes que aprobaban. De acuerdo al análisis, más allá de buscar las causas, era urgente intervenir para mejorar la situación. En consecuencia, se formuló un proyecto de investigación-acción con el propósito de reducir la cantidad de estudiantes que reprobaban, mejorar el nivel de calificaciones e innovar con calidad en el aprendizaje y la enseñanza de la programación de computadores.

En el marco de dicho proyecto se priorizó la acción, para lo cual se propuso la iniciativa de experimentar usando pruebas automáticas como instrumento para la evaluación de los estudiantes. Dicha iniciativa se inspiró en tres tópicos *ad hoc*, conocidos previamente por los profesores involucrados en el proyecto:

- Las experiencias y lecciones aprendidas de la programación competitiva, en la cual los estudiantes se enfrentan a competencias de programación de computadores, cuyo éxito depende de la rapidez con la cual se resuelven varios problemas [14].
- La evaluación automática, que consiste en automatizar la retroalimentación que recibe el estudiante en sus tareas y evaluaciones de programación de computadores [15]-[18].
- La estrategia de aprendizaje experiencial o vivencial, en especial el aprendizaje basado en problemas y aprendizaje basado en retos [19], que enfatiza en aprender haciendo.

La estrategia educativa propuesta de usar pruebas automáticas se resume en que los

profesores diseñan pruebas para los ejercicios de programación de computadores con los que se evalúa a los estudiantes, dichas pruebas funcionan como una rúbrica de evaluación y son automáticas, no requieren intervención humana. En la medida en que un profesor o un equipo de profesores diseñen varios ejercicios y sus pruebas, se consigue con el tiempo un banco de ejercicios, lo cual propicia el entrenamiento autónomo de los estudiantes y la retroalimentación inmediata, sin dependencia del profesor. Además, se cambia la dedicación del tiempo del profesor: más tiempo diseñando ejercicios y pruebas y muy poco tiempo calificando y se elimina el sesgo en la evaluación, pues es una máquina la que evalúa según criterios objetivos planeados al diseñar las pruebas.

El propósito de este documento es divulgar los detalles y resultados de esta iniciativa, para lo cual el documento se organiza de la siguiente forma: En primer lugar, se expone el análisis realizado a las calificaciones en las asignaturas de programación de computadores. Luego se reseña la literatura revisada para soportar o descartar los tópicos *ad hoc* en los que se inspira la propuesta de pruebas automáticas. En tercer lugar, se describe en detalle la estrategia, las actividades de aprendizaje y herramientas tecnológicas. Posteriormente, se muestran los resultados obtenidos, contrastando el desempeño de los estudiantes con la evaluación que dichos estudiantes realizaron al profesor. Finalmente, se plantean las conclusiones, se discuten las implicaciones educativas y se proponen trabajos futuros.

II. ANÁLISIS DE LAS CALIFICACIONES EN ASIGNATURAS DE PROGRAMACIÓN DE COMPUTADORES (BEFORE TEST)

Al inicio de cada semestre académico, el programa de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander (UFPS), sede Cúcuta, en cumplimiento de la política institucional de autoevaluación, revisa con el equipo de profesores las fortalezas y debilidades en cada una de las asignaturas, como mecanismo de retroalimentación que permite emprender acciones de mejoramiento. En el caso específico de Programación de Computadores, el programa tiene cuatro asignaturas que se cursan en los primeros cuatro semestres: Fundamentos de Programación, Programación Orientada a Objetos I, Programación Orientada a Objetos II y Estructuras de Datos.

Para cada una de estas asignaturas se analizaron las calificaciones de todos los semestres durante el periodo comprendido entre el segundo semestre de 2012 y el segundo semestre de 2016, un total de 1.544 registros (Figuras 1 y 2), según el Sistema de Información Académica. En estos registros no se considera la deserción por cancelación de una asignatura antes de finalizar el semestre.

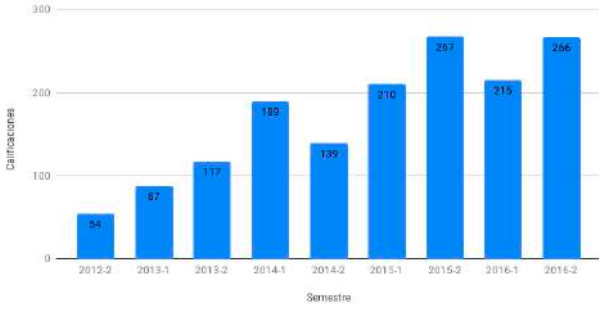

Fig. 1. Cantidad de registros de calificaciones por semestre

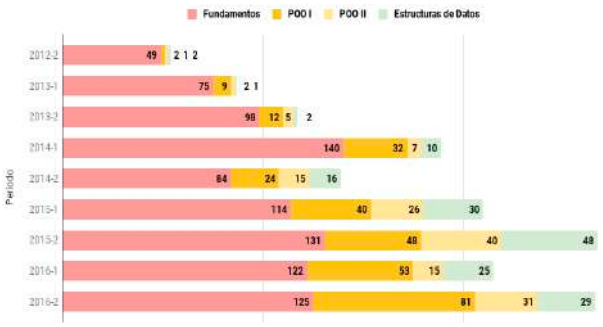

Fig. 2. Cantidad de registros de calificaciones por semestre y asignatura

Como se puede apreciar en las figuras, la cantidad de estudiantes se incrementa cada semestre, en lugar de mantenerse constante. Este comportamiento se explica por la retención de estudiantes que reprueban las asignaturas de Programación de Computadores. En consecuencia,

el análisis posterior se enfocó en tres preguntas: (i) ¿Cuál es el nivel de aprobación y reprobación de las asignaturas (se aprueba < 3.0)?, (ii) ¿Cuánto tarda un estudiante en aprobar una asignatura? (iii) ¿Cuál es el nivel de las calificaciones en estas asignaturas cuando el estudiante aprueba?

Para la primera pregunta, en promedio el 63 % de los estudiantes aprueban y el 37 % restante reprueba. El único curso que no cumple con este patrón de comportamiento es Fundamentos de Programación de primer semestre, el cual tiene una relación de 34 % aprobados y 66 % reprobados (Figura 3).

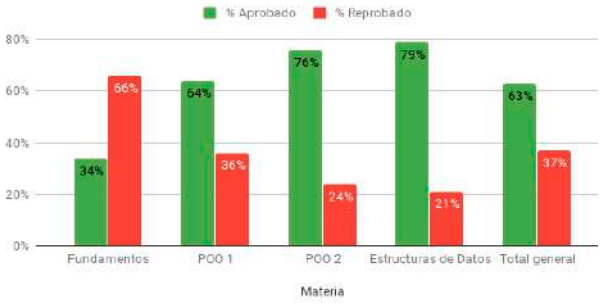

Fig. 3. Porcentaje de aprobación y reprobación por asignatura

Considerando la importancia del primer semestre, se analizó en detalle la asignatura Fundamentos de Programación. Se identificó que la situación mejoró en los últimos semestres y justo en el segundo semestre de 2016 se logró invertir ese comportamiento (Figura 4), con un 52 % aprobados y 48 % reprobados.

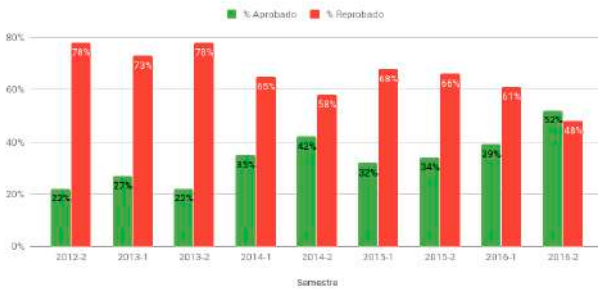

Fig. 4. Porcentaje de aprobación y reprobación en Fundamentos de Programación

La conclusión evidente para la primera pregunta es que hay una tasa alta de reprobación de las asignaturas de Programación de Computadores. Y la situación es más evidente al responder la segunda pregunta: Sólo nueve estudiantes aprobaron la primera vez una asignatura de Programación de Computadores, sin necesidad de repetirla, 234 estudiantes repitieron una vez, 101 repitieron dos veces, 50, tres veces y 39 estudiantes repitieron más de tres veces, con un máximo de nueve veces (Figura 5). Es importante aclarar que estas calificaciones incluyen todos los intentos que ofrece la UFPS para que un estudiante apruebe, como exámenes de habilitación y cursos en vacaciones.

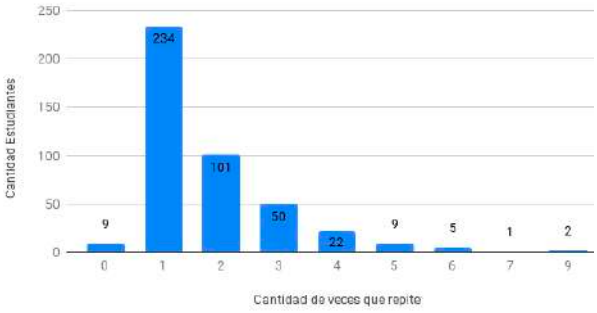

Fig. 5. Semestres requeridos para aprobar Programación de Computadores

En general, los 1.544 registros de calificaciones analizados correspondieron a 433 estudiantes y de esos registros 774 fueron de calificaciones reprobadas, lo cual arroja un promedio de 2,8 intentos para aprobar una asignatura de Programación de Computadores. Los otros 770 registros de calificaciones aprobadas se distribuyeron en cinco rangos para responder la tercera pregunta (Figura 6).

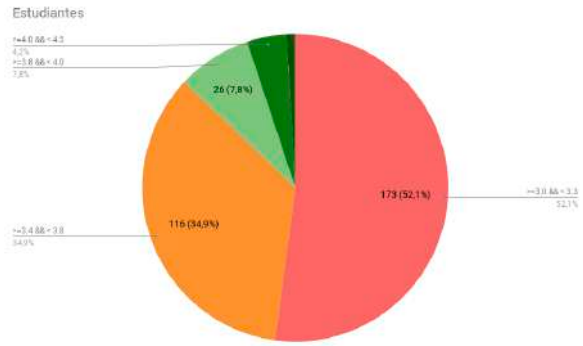

Fig. 6. Nivel de calificaciones aprobadas en asignaturas de Programación de Computadores

Siguiendo la Figura 6, el promedio de las calificaciones aprobadas es de 3,41, el porcentaje más alto (87,05 %) corresponde a calificaciones inferiores a 3,80 y sólo el 12,95 % obtiene calificaciones de nivel alto (>=3,80). La Figura 7 permite apreciar en detalle el promedio y la desviación estándar de las calificaciones aprobadas, reprobadas y el total, de cada una de las asignaturas. El promedio de las calificaciones reprobadas es 2,0 y el promedio total es 2,70.

Asignatura	Aprobado		Reprobado		Total	
	Promedio	Desviación	Promedio	Desviación	Promedio	Desviación
Fundamentos	3,37	0,39	1,93	0,64	2,44	0,89
POO 1	3,37	0,40	2,18	0,61	2,95	0,75
POO 2	3,51	0,44	2,38	0,43	3,28	0,63
Estructuras Datos	3,48	0,46	2,43	0,50	3,29	0,62
Total general	3,41	0,42	2,00	0,64	2,70	0,89

Fig. 7. Promedio y desviación estándar de calificaciones en asignaturas de Programación de Computadores

De acuerdo al análisis de las calificaciones, se derivaron tres conclusiones:

- Hay una tasa alta de reprobación de asignaturas de Programación de Computadores.
- Hay un bajo nivel en las calificaciones de los estudiantes que aprueban Programación de Computadores.
- Las dos asignaturas que deben intervenirse son: Fundamentos de Programación y Programación Orientada a Objetos, de primero y segundo semestre, respectivamente.

Considerando las conclusiones, más allá de buscar las causas de la situación identificada, para el Programa de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander (UFPS), Cúcuta, era fundamental intervenir para mejorar la situación. En ese sentido, se procedió a identificar en la literatura experiencias exitosas a fin de implementar acciones inmediatas para reducir la tasa de reprobación y mejorar el nivel de calificaciones, como se expone a continuación.

III. REVISIÓN DE LITERATURA SOBRE PROBLEMAS Y SOLUCIONES DE LA FORMACIÓN EN PROGRAMACIÓN DE COMPUTADORES

Se realizó una búsqueda en SCOPUS y ACM, usando las palabras programación (programming), aprender (learn) y enseñar (teach) y filtrando por los campos título (TITLE), resumen (ABSTRACT) y palabras clave (KEYWORD). Se encontró divulgación abundante y permanente desde los años sesenta del siglo pasado, en revistas tanto de ciencias de la educación como de ciencias de la computación, así como en revistas de tópicos mixtos y aplicados. En SCOPUS se encontraron menos registros (473) que en ACM (1.483), pero en ambos el interés por el tema es creciente (Figura 8). Asimismo, al usar otras opciones se obtienen muchos más resultados, pues existen publicaciones enfocadas solo en aprendizaje (*learn*) o solo en enseñanza (*teach*), incluso se encuentran publicaciones sin estas palabras y en su lugar hay términos como educación (*education*) entrenamiento (*training*), instrucción (*instruction*) y formación (*schooling*), entre otros.

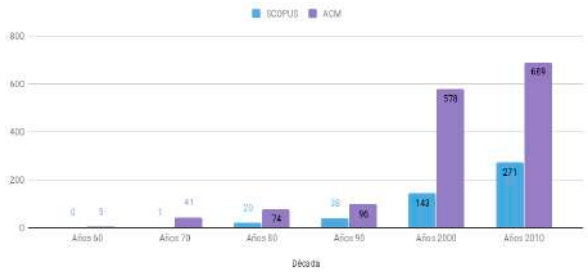

Fig. 8. Resultados de búsqueda en SCOPUS y ACM para "programming AND learn AND teach"

En general, la literatura sobre la formación en Programación de Computadores es abundante y ofrece diversos tópicos. Algunos autores se limitan a estudiar el problema sin especificar soluciones y otros abordan en detalle propuestas de solución. Para efectos de este documento es conveniente destacar los siguientes tópicos generales:

- Delimitación conceptual de qué es programación de computadores, estableciendo diferencias y relaciones con la resolución de problemas (*problem solving*), el pensamiento computacional (*computational thinking*), la codificación (*coding*) y la gerencia de proyectos de software (*software engineering*).
- Aspectos educativos diversos, como estilos de aprendizaje y docencia, aprender haciendo, conocimiento tácito y explícito, leer y escribir código, pedagogía y didáctica y la granularidad del currículo.
- Currículo del primer curso (*first course*) de programación de computadores, contenidos, actividades, evaluación, etc.
- Integración y articulación de la programación de computadores con otras áreas, como matemáticas, interfaces gráficas de usuario, juegos, redes, ingeniería del software, etc.
- Aspectos psicológicos y sociológicos como la motivación y la ansiedad o las preferencias y dificultades (del profesor y el estudiante) o el trabajo individual y grupal.
- El lenguaje y paradigma de programación usado para aprender a programar.
- El entorno de desarrollo IDE (*integrated development environment*) y sus características, como las facilidades de depuración (*debug, step-by-step*) y sugerencias de código (*code suggestion/completion*).
- El uso de herramientas tecnológicas de apoyo, como herramientas para visualización, diseño y automatización de tareas.
- El uso de herramientas para pruebas automáticas de software y evaluación automática de tareas de programación, que se deriva del tópico anterior.

Para la búsqueda sobre la idea específica de usar pruebas automáticas, se limitaron los resultados al término evaluación formativa (assessment) y se omitieron otros términos (test, evaluation, appraisal), menos relacionados con el problema de interés. De manera consistente con la búsqueda anterior, en SCOPUS se encontraron menos registros (151) que en ACM (216), aunque en proporción diferente (Figura 9).

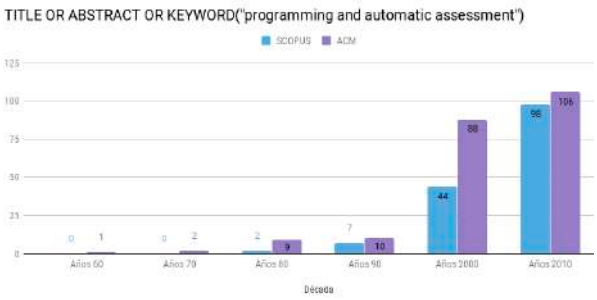

Fig. 9. Resultados de búsqueda en SCOPUS y ACM para "programming AND automatic assessment"

Puesto que el objetivo de la revisión de literatura era identificar soluciones y experiencias exitosas para abordar los problemas de la formación en programación de computadores, se realizó una lectura de los títulos y resúmenes de las publicaciones usando como criterio de ordenamiento la cantidad de citaciones, a fin de enfocarse en el impacto, y se profundizó en el detalle de las publicaciones que aportaban soluciones al contexto específico de la UFPS y que se relacionaban con los tópicos *ad hoc* que inspiraron la propuesta inicial, lo cual se expone a continuación.

A. Programación Competitiva

La programación competitiva es una actividad en la que equipos de tres estudiantes se enfrentan a resolver problemas complejos relacionados con matemáticas y ciencias de la computación, para lo cual deben escribir la mayor cantidad de programas de computador en el menor tiempo posible [20]. Este tipo de competencias iniciaron en 1974, y en la actualidad agremian mundialmente a muchos profesores y estudiantes en lo que se conoce como la Internacional Collegiate

Programming Contest (ICPC) [21]. En Colombia, la Asociación Colombiana de Ingenieros de Sistemas (ACIS) realiza desde 1986 la Maratón Nacional de Programación, que permite clasificar a la competencia latinoamericana, la cual otorga uno o dos cupos para la competencia mundial ICPC [22]. Asimismo, en Colombia existen dos ligas que mensualmente organizan competencias de entrenamiento: la Red de Programación Competitiva (RPC), que organiza competencias para toda Latinoamérica, desde México hasta Argentina [23], y la Colombian Collegiate Programming League (CCPL), que realiza competencias para estudiantes universitarios de Colombia [24].

El Programa de Ingeniería de Sistemas de la UFPS se vinculó y se ha mantenido vigente en este tipo de competencias desde hace cinco años, bajo el liderazgo de los mismos estudiantes, y ha logrado buenos resultados, en especial la motivación y el autoaprendizaje en tópicos que no son el eje del currículo del programa [14]. Por el éxito de esta experiencia, los estudiantes líderes del grupo desarrollaron un marco de trabajo y una plataforma de entrenamiento (*Training Center*), disponible en el sitio http://trainingcenter.cloud.ufps.edu.co [25]. En dicha plataforma se ha conformado un banco de problemas y de material de estudio que ha permitido a los estudiantes aprender y competir exitosamente a nivel nacional y latinoamericano.

Para el proyecto de Training Center se estudiaron diversas plataformas de programación competitiva, y se destacan las siguientes:

URI (https://www.urionlinejudge.com.br/)
UVA (https://uva.onlinejudge.org/)
Codeforces (https://codeforces.com/)
A20J (https://a2oj.com/)
Codechef (https://www.codechef.com/)
Boca (http://bombonera.org/)
Acepta el Reto (https://www.aceptaelreto.com/).

Todas estas plataformas permiten subir un programa de computador en varios lenguajes de programación (Java, C y C++, Python, C#, entre otros) y ofrecen retroalimentación inmediata, informando si el programa resuelve un problema determinado y cumple con los requerimientos no funcionales de rendimiento en procesador y memoria RAM. Por ser un contexto de competencia, estas plataformas solo ofrecen retroalimentación total: indican si el programa cumple o no cumple, pero no hay pistas o indicaciones parciales para guiar al programador hacia una solución. No obstante, en el contexto del aprendizaje de programación de computadores, es deseable la retroalimentación parcial. En consecuencia, inspirados en el éxito de la programación competitiva, se propuso como una posible solución el uso de plataformas similares, pero para apoyar el aprendizaje de cursos de programación de computadores.

B. Pruebas Automáticas y VPL

El uso de pruebas automáticas en cursos de programación de computadores aparece en la literatura como evaluación automática (*automatic assessment*) y es algo que se ha probado con éxito desde hace cuatro décadas [15], [18], [26], [27]. Este tipo de plataformas abordan varios aspectos, y se destacan tres que aplican para el presente trabajo: En primer lugar, enfrentan el problema de la retroalimentación que necesita el estudiante cuando estudia sin presencia del profesor o tutor, con lo cual se propicia el entrenamiento autónomo e independiente, el autoaprendizaje y la motivación individual [12], [16], [18]. En segundo lugar, reducen la sobrecarga del profesor, puesto que para la evaluación de cada estudiante se requiere mucho tiempo [12], [15], [16]. Y tercero, permiten enfrentar el problema del plagio, puesto que se tiene un repositorio de problemas y soluciones que se pueden comparar de manera mucho más sencilla y rápida [28].

También, las plataformas de pruebas automáticas permiten usar varios lenguajes de programación, con lo cual los cursos se pueden orientar al desarrollo de habilidades, solución de problemas y conceptos fundamentales, en lugar de centrarse en aprender un lenguaje específico. Adicionalmente, el uso de pruebas automáticas permite introducir la técnica de Desarrollo de Software Dirigido por Pruebas (*Test Driven Software Development*), usada en el

contexto del software libre y los modelos ágiles, como los propuestos por Kent Beck [29] y Rober Martin [30], usados ampliamente en el mundo profesional de la Industria 4.0.

Para el presente trabajo se implementó la plataforma VPL (*Virtual Programming Lab*), una iniciativa de la Universidad Las Palmas de Gran Canaria (España) que se integra con el Sistema de Gestión de Aprendizaje Moodle, permite usar varios lenguajes de programación y ofrece verificación de plagio [18], [28]. VPL tiene licencia libre y abierta GNU/GPL, cuenta con documentación detallada, hay más de veinte publicaciones relacionadas en su sitio web [31] y las estadísticas de uso indican que 824 sitios Moodle usan VPL (Figura 10).

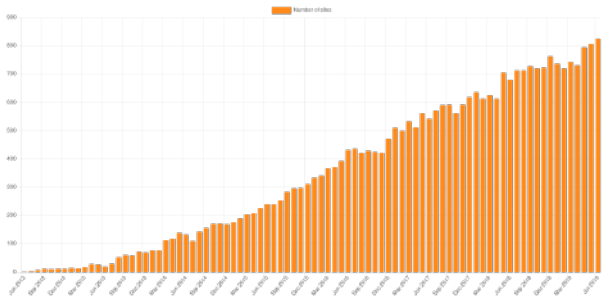

Fig. 10. Estadística de uso VPL en Moodle [31]

La arquitectura de VPL (Figura 11) oculta la complejidad del proceso de calificación, seguridad y lenguajes en una máquina virtual llamada "jaula de ejecución". Dicha jaula recibe peticiones desde el servidor Moodle, lo cual se consigue mediante un componente (plugin). VPL ofrece además un editor de código fuente enriquecida que se ejecuta en el navegador (*browser*) usando HTML y JavaScript, en el cual el estudiante puede escribir su programa sin requerir un entorno de desarrollo (IDE). Este editor permite realizar exámenes presenciales y reduce la complejidad que involucra el aprendizaje de un IDE.

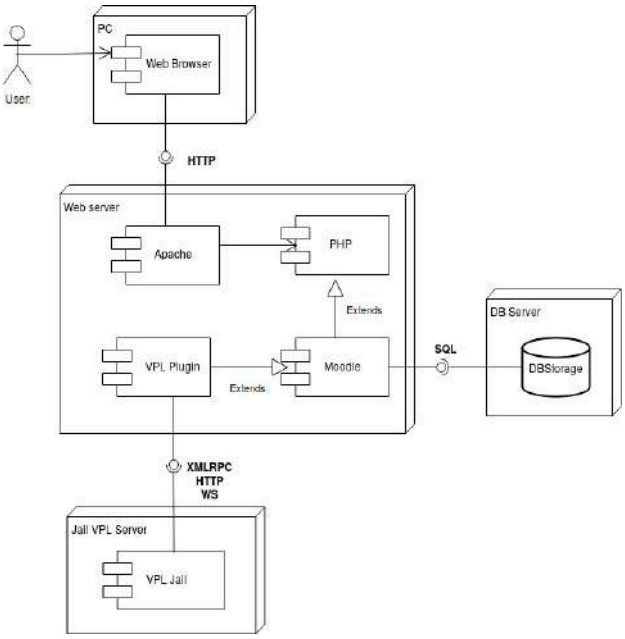

Fig. 11. Arquitectura de VPL y Moodle

Cada ejercicio de programación en VPL y Moodle tiene un enunciado y las pruebas. Existen dos alternativas para las pruebas: La primera son las pruebas de caja negra (Figura 12), que consisten en una lista de datos de entrada y salida llamados casos de prueba (*test cases*). VPL ejecuta el programa del estudiante, le suministra las entradas de prueba y luego compara las salidas del programa con las salidas de prueba. En cada caso de prueba se puede configurar que la calificación aumente o disminuya según el resultado y se pueden agregar comentarios de retroalimentación. Toda la información se almacena en un repositorio que puede ser consultado por el profesor y el estudiante, según corresponda. Este tipo de pruebas son ideales para cursos de primer semestre, donde se aprende el paradigma de programación estructurado/procedural.

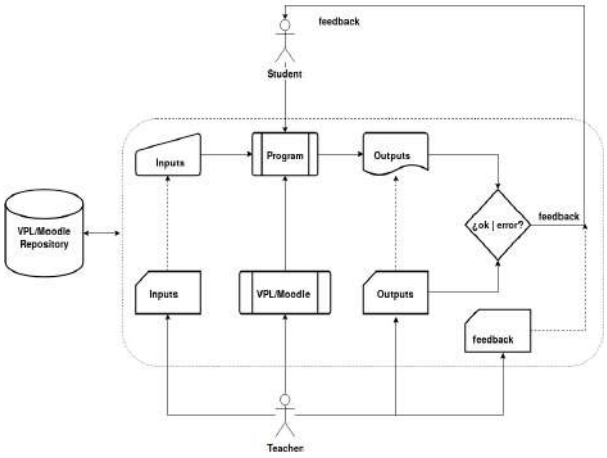

Fig. 12. Arquitectura de VPL y Moodle de pruebas de caja negra

La segunda alternativa de pruebas en VPL son las pruebas unitarias (Figura 12), usando librerías especializadas como JUnit.

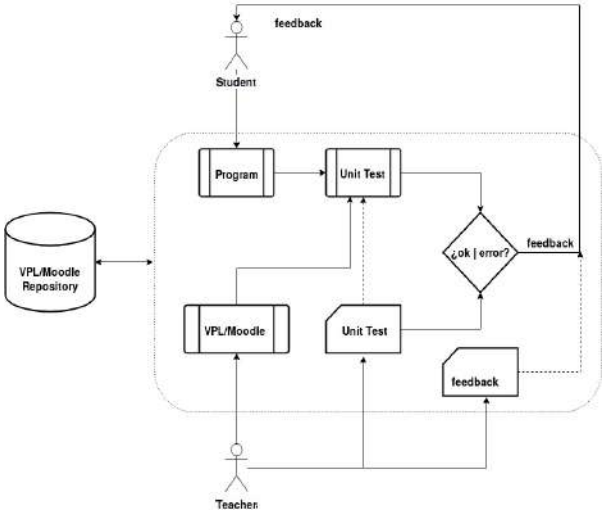

Fig. 13. Arquitectura de VPL y Moodle de pruebas unitarias

VPL ejecuta las pruebas unitarias usando como entrada el programa del estudiante. Las pruebas unitarias deben configurarse para que generen retroalimentación y calificación, para lo cual deben seguirse la documentación y los ejemplos en el sitio web de VPL. Este tipo de pruebas son ideales para cursos de segundo semestre, en los que se aprende el paradigma de programación orientado a objetos.

Para la calificación, VPL y Moodle permiten una escala porcentual de 0 a 100 o una escala específica seleccionada por el profesor. Además, tienen opciones de trabajo individual o en grupo y opciones para permitir varios intentos. Como cada ejercicio es una actividad de evaluación dentro del curso en Moodle, el profesor puede llevar un registro detallado de la dedicación y el desempeño del estudiante, y el estudiante puede intentar resolver el ejercicio tantas veces como lo permita el profesor.

Puesto que VPL es un plugin de Moodle, y junto a la jaula de ejecución se dispone del código fuente y la documentación con licencia GNU/GPL, los usuarios pueden personalizar el plugin y la jaula para que se adapten según las necesidades específicas. No obstante, las opciones que ofrece VPL para pruebas caja negra y unitarias son bastante sencillas de usar y hacen a VPL muy útil para implementar fácilmente la idea de pruebas automáticas en cursos de Programación de Computadores. Además, la integración con el Sistema de Gestión de Aprendizaje Moodle facilita su uso en contextos educativos virtuales, presenciales o mixtos, lo que permite aprovechar las ventajas tecnológicas para introducir estrategias educativas como el aprendizaje basado en retos.

C. Aprendizaje Basado en Retos

El aprendizaje basado en retos es un enfoque pedagógico centrado en el estudiante, que privilegia el hacer como medio para llegar al saber, con la particularidad de que el hacer se promueve estimulando al estudiante mediante un reto [19]. Un reto es una actividad de aprendizaje, tarea o evaluación, que consiste en una situación real o hipotética con la cual se desafía al estudiante a buscar el saber y a ponerlo en práctica, algo que se requiere diariamente en profesiones de ingeniería y ciencia [19], [32]. En el aprendizaje basado en retos se enfatiza la importancia de la creatividad y el ingenio, por eso el centro es el estudiante y no el contenido ni las actividades ni el profesor, el ser humano es el centro del aprendizaje. Esta idea se relaciona con la escuela de pensamiento de la psicología de

la programación de computadores, que plantea la importancia de abordar la programación como una actividad humana [7], [8], [33], [34]. Según la psicología de la programación de computadores cualquier persona puede aprender a programar y puede llegar a ser un experto programador, siempre y cuando se reciba una formación inicial apropiada, se tenga un entrenamiento permanente y se consiga una motivación apropiada y oportuna [8].

Una dificultad al implementar el aprendizaje basado en retos es la formulación de los retos y su posterior evaluación. Una vez un estudiante logra resolver un reto es fundamental suministrarle retos adicionales, y esto demanda una carga de trabajo para el profesor que es proporcional a la cantidad de estudiantes y que crece con el tiempo y de manera proporcional al aprendizaje [12], [35], [36]. En ese sentido, resulta de gran ayuda el uso de herramientas tecnológicas, como el enfoque descrito de pruebas automáticas con VPL, que permite disponer de un repositorio de retos y retroalimentación automática, y logra que el estudiante mantenga la motivación y además lidere su propio aprendizaje [15], [16], [18].

Como lo indica la literatura sobre los problemas y soluciones de la formación en programación de computadores, influyen mucho las prácticas educativas de ambos, profesores y estudiantes [6], [37], [38]. Asimismo, las experiencias exitosas se derivan de cambios innovadores en estas prácticas [11], [18], [39]. Es así que la literatura respalda la idea de introducir el aprendizaje basado en retos en combinación con pruebas automáticas, partiendo de la experiencia exitosa de la programación competitiva, como se explica a continuación.

IV. ESTRATEGIA, ACTIVIDADES DE APRENDIZAJE Y HERRAMIENTAS TECNOLÓGICAS

A. Curso de Primer Semestre – Fundamentos de Programación

La Figura 14 resume la estrategia implementada para primer semestre, cuyo objetivo es el aprendizaje de los Fundamentos de Programación. En primer lugar se tienen dos momentos: el trabajo en la clase presencial, con el acompañamiento y supervisión del profesor y el trabajo independiente, ambos usando recursos tecnológicos. Siguiendo las ideas de la psicología de la programación, el aprendizaje de la programación de computadores se relaciona con leer y escribir código fuente [8]. En consecuencia, la estrategia prioriza leer código fuente en clase presencial con acompañamiento del profesor, y se deja como trabajo independiente escribir código, para lo cual el estudiante puede apoyarse en la retroalimentación de las pruebas automáticas.

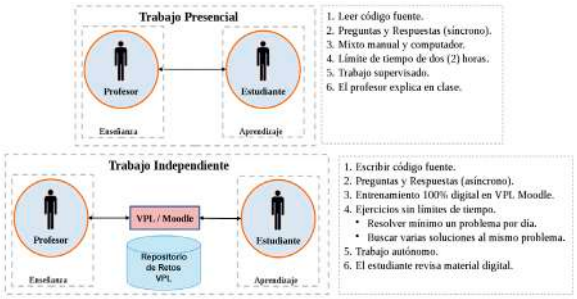

Fig. 14. Estrategia

En clase presencial se explican ejercicios de ejemplo y se resuelven dudas, siempre bajo la supervisión del profesor y con un tiempo límite de dos horas, que es el tiempo programado para cada una de las dos clases semanales. Este límite de tiempo implica que no todos tienen tiempo para preguntar y no todas las dudas se logran resolver. Hasta aquí no hay diferencias con un modelo clásico de clase de programación de computadores. La diferencia aparece al involucrar las pruebas automáticas, pues el estudiante, en su trabajo independiente, tendrá retroalimentación inmediata y sin límites de tiempo ni de intentos. La regla establecida es resolver como mínimo un problema por día, de lunes a viernes, e intentar encontrar varias soluciones diferentes al mismo problema.

Surge entonces un desafío para el profesor: lograr acopiar un banco de ejercicios suficiente para satisfacer la regla de resolver un problema diario. En la medida en que un profesor o un equipo de profesores diseñen varios ejercicios y sus pruebas, se consigue con el tiempo este banco de ejercicios.

La estrategia propicia el entrenamiento autónomo de los estudiantes y la retroalimentación inmediata sin dependencia del profesor. Ahora se cambia la dedicación del tiempo del profesor: más tiempo diseñando ejercicios y pruebas y muy poco tiempo calificando y, como ya se dijo, se elimina el sesgo en la evaluación, pues es una máquina la que evalúa según criterios objetivos planeados al diseñar las pruebas.

De acuerdo con esta estrategia, se implementó en Moodle el curso VPL (Virtual Programming Lab) Training, disponible bajo licencia Creative Commons (CC BY-NC-SA 4.0), con más de setenta problemas (https://uvirtual.cloud.ufps.edu.co/course/view.php?name=VPLTraining). El curso organiza los ejercicios en cinco unidades que abordan los tópicos principales de Fundamentos de Programación (Figura 15). El estudiante recibe en clase la explicación de cada tópico y debe desarrollar como trabajo independiente cada uno de los ejercicios, usando VPL y Moodle.

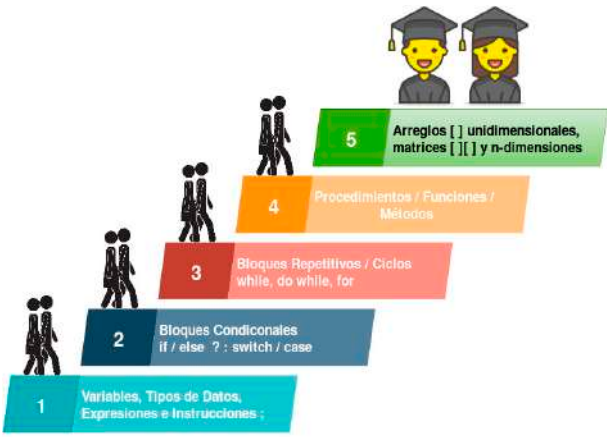

Fig. 15. Actividades de aprendizaje primer semestre

El estudiante puede hacer consultas por medios digitales (foro, chat y correo electrónico), las cuales se resuelven de manera personalizada. Asimismo, a través del foro y otros medios el estudiante puede compartir preguntas y soluciones con sus compañeros. En clase presencial el profesor enfoca la explicación en los problemas más complejos, en las dudas que los estudiantes informaron y en los demás aspectos derivados de la revisión, que manualmente realiza el profesor como parte del acompañamiento y asesoría al curso. De este modo las actividades de aprendizaje se resumen en la resolución de problemas, la lectura y escritura de código fuente y una conversación permanente entre estudiantes y profesor sobre los tópicos y sus ejercicios, lo cual coincide mucho con el escenario real al desarrollar software en ámbitos profesionales.

En cuanto a las herramientas tecnológicas, además de VPL Moodle, se usa el entorno de desarrollo (IDE) BlueJ, por sus facilidades para visualización de conceptos y depuración [39], [40]. Los estudiantes pueden usar BlueJ en clase o desconectados, y en línea VPL les ofrece un entorno para programar directamente en el navegador y allí mismo conocer la retroalimentación (Figura 16). Por su parte, el profesor puede observar en VPL el detalle de todas las entregas e intentos de los estudiantes, hacer comentarios y evaluar el nivel de similitud o plagio (Figura 17).

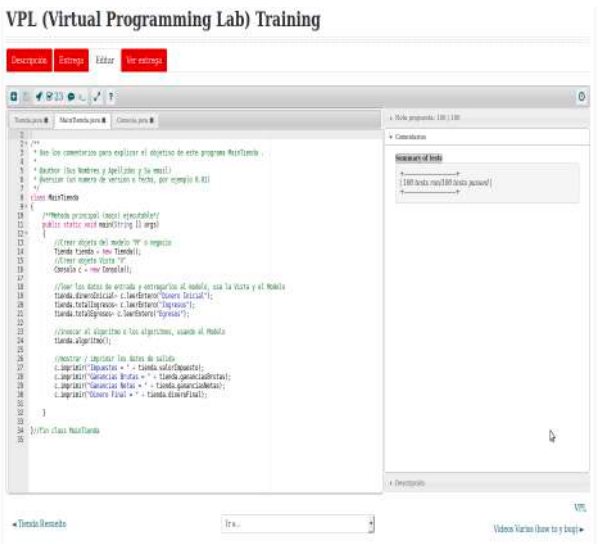

Fig. 16. Herramientas tecnológicas, entorno IDE de VPL Moodle para el estudiante

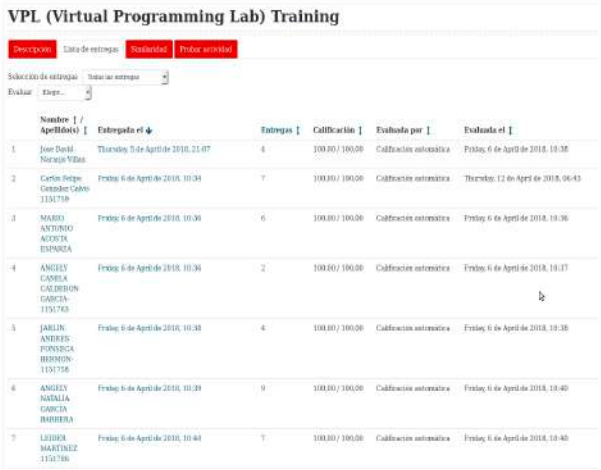

Fig. 17. Herramientas tecnológicas, entorno de VPL Moodle para el profesor

La implementación de esta estrategia, las actividades de aprendizaje y herramientas inició en el primer semestre del año 2017 y se ha mantenido desde entonces. En el segundo semestre del mismo año se avanzó a segundo semestre para darle continuidad al proceso, como se explica seguidamente.

B. Curso de Segundo Semestre – Programación Orientada a Objetos I

Para el segundo semestre se mantuvo la misma estrategia, ajustando las actividades de aprendizaje al Paradigma Orientado a Objetos y considerando los tópicos establecidos por el currículo del programa de Ingeniería de Sistemas (Figura 18).

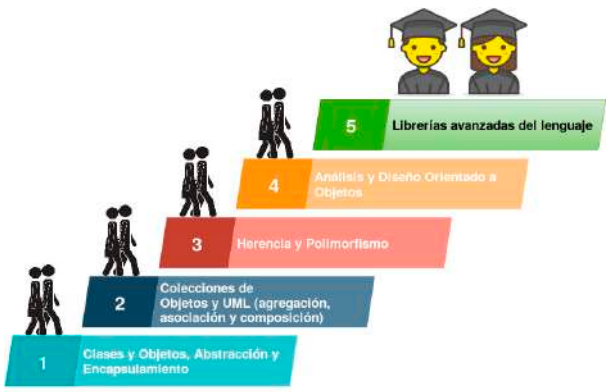

Fig. 18. Actividades de aprendizaje segundo semestre

A diferencia de primer semestre, en segundo semestre las pruebas automáticas dejaron de ser caja negra y pasaron a ser pruebas unitarias, usando JUnit. En consecuencia, el estudiante puede ejecutar las pruebas unitarias sin conexión a VPL, usando únicamente BlueJ y publicando en VPL sólo la versión definitiva del ejercicio, lo cual dificulta hacer seguimiento al trabajo del estudiante. Una dificultad en segundo semestre corresponde a los estudiantes que no pasaron por el mismo proceso el semestre anterior, por lo que es necesario remitirlos al curso de VPL Training para nivelar. Algo que también conviene mencionar es que la cantidad de problemas se redujo a un promedio de veinte, considerando la densidad y complejidad del paradigma orientado a objetos, y a la fecha aún no se han acopiado en un único curso de entrenamiento como se hizo para el primer semestre.

Como se muestra a continuación, en la sección de resultados, durante dos años se ha mantenido en ambos semestres la estrategia, actividades de aprendizaje y herramientas tecnológicas, lógicamente con algunas pequeñas adaptaciones según la retroalimentación de los mismos estudiantes y la evaluación del profesor a cargo de la iniciativa.

V. RESULTADOS Y DISCUSIÓN (AFTER TEST)

En general los resultados se pueden considerar satisfactorios. Esta afirmación se fundamenta en las calificaciones finales obtenidas por los estudiantes, la tasa promedio de aprobación y la evaluación que los estudiantes realizan al profesor semestralmente. Las Figuras 19 y 20 permiten apreciar el porcentaje de estudiantes que aprobaron y reprobaron los cursos de primero y segundo semestres en los cuales se implementaron pruebas automáticas, con un promedio de 35 estudiantes en primer semestre y de 30 en segundo semestre.

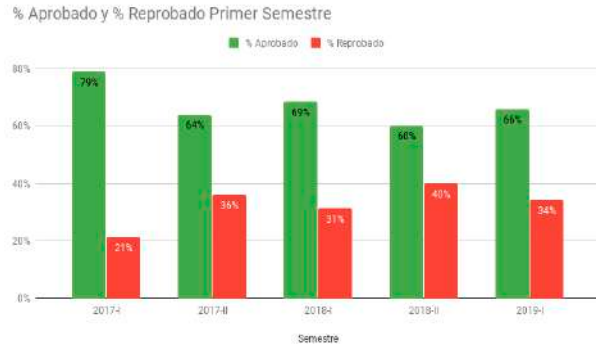

Fig. 19. Porcentaje de aprobación y reprobación primer semestre, usando pruebas automáticas

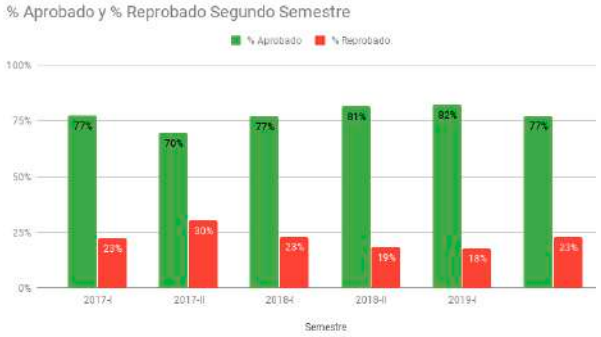

Fig. 20. Porcentaje de aprobación y reprobación segundo semestre, usando pruebas automáticas

Se realizó un análisis de los estudiantes que reprobaron en primer semestre y, en su mayoría, repiten una sola vez la asignatura. Durante los semestres analizados se ha conseguido mantener un nivel de reprobación aceptable y se observa una mejora en las calificaciones, con lo cual se ha minimizado la cantidad de estudiantes con calificaciones inferiores a 2,0, como se muestra en las Figuras 21 y 22.

	Aprobado		Reprobado		Total	
Semestre	Promedio	Desviación	Promedio	Desviación	Promedio	Desviación
2017-I	3,43	0,38	2,46	0,46	3,22	0,54
2017-II	3,39	0,36	2,59	0,51	2,94	0,61
2018-I	3,21	0,41	2,40	0,38	2,89	0,53
2018-II	3,33	0,43	2,12	0,38	2,76	0,67
2019-I	3,59	0,54	2,68	0,62	3,23	0,73
Total general	3,39	0,42	2,45	0,47	3,01	0,62

Fig. 21. Promedio y desviación de calificaciones en primer semestre, usando pruebas automáticas

	Aprobado		Reprobado		Total	
Semestre	Promedio	Desviación	Promedio	Desviación	Promedio	Desviación
2017-I	3,40	0,57	2,43	0,08	3,18	0,63
2017-II	3,29	0,37	2,67	0,15	2,97	0,42
2018-I	3,27	0,51	2,53	0,23	3,10	0,46
2018-II	3,09	0,48	2,42	0,32	2,96	0,47
2019-I	3,36	0,62	2,72	0,15	3,24	0,41
Total general	3,28	0,51	2,55	0,19	3,09	0,48

Fig. 22. Promedio y desviación de calificaciones en segundo semestre, usando pruebas automáticas

Por otra parte, en la evaluación docente que realizan los estudiantes cada semestre, el profesor a cargo del proyecto ha obtenido calificaciones y observaciones muy buenas (Figura 23), lo que indica que los estudiantes se sienten satisfechos con la metodología implementada.

Semestre	Primer Semestre	Segundo Semestre
2017-I	4,81	4,75
2017-II	4,85	4,75
2018-I	4,82	4,76
2018-II	4,81	4,68
2019-I	4,74	4,82
Promedio	4,81	4,75

Fig. 23. Calificaciones del profesor en la evaluación semestral que le hacen sus estudiantes

De acuerdo con las observaciones del profesor a cargo del proyecto, los estudiantes se mantienen motivados y en actividad permanente durante todo el semestre y se genera una dinámica de colaboración e interés por resolver los problemas en VPL Moodle. Una situación interesante es que algunos estudiantes tienden a cuestionar los resultados y argumentan que la plataforma o las pruebas están equivocadas. No obstante, una vez en clase se resuelven las inquietudes, los estudiantes aceptan los errores y retoman el ejercicio hasta resolverlo por completo. Asimismo, se experimentaron algunos inconvenientes de rendimiento, lo que generó la necesidad de migrar tanto Moodle como VPL a un modelo de Computación en Nube (*Cloud Computing*) usando una arquitectura de contenedores (*docker*), a fin de mejorar el rendimiento y facilitar el crecimiento y decrecimiento de recursos computacionales.

Finalmente, hay dos resultados adicionales que se han identificado durante el proyecto: en primer lugar, algunos estudiantes han comenzado desde primer semestre a utilizar plataformas de programación competitiva y a participar en el semillero, pues allí profundizan en tópicos que no se abordan en las clases y que les interesan y los motivan. En segundo lugar, en el curso de Programación Orientada a Objetos II, de tercer semestre, se realizó un experimento de introducir Programación Web y uso de Bases de Datos, considerando que la mayoría de los estudiantes provenían de los cursos donde se usaron pruebas automáticas. El experimento fue exitoso, y los estudiantes desarrollaron proyectos de aplicaciones web interesantes bajo un enfoque de innovación y emprendimiento.

VI. CONCLUSIONES Y TRABAJO FUTURO

Si bien falta mucho trabajo para generar conclusiones definitivas, los resultados obtenidos a la fecha indican que hay un nivel importante de satisfacción de los estudiantes y conviene continuar abordando el problema y mejorando el enfoque propuesto. Básicamente se mejoró la situación inicial, pero aún hay un volumen importante de estudiantes que reprueban y los niveles de calificación siguen siendo aceptables.

Es importante pasar ahora de la acción a la investigación, para identificar con mayor claridad las dificultades en el proceso de aprendizaje y las prácticas más apropiadas tanto de profesores como de estudiantes, a fin de aprobar exitosamente los cursos de Programación de Computadores y mejorar el nivel de calificaciones.

En el análisis presentado en este documento no se estudiaron en detalle los estudiantes que reprobaron, lo cual es algo fundamental para entender por qué es difícil aprender Programación de Computadores y cómo superar esa dificultad. De alguna manera, los estudiantes que reprueban demandan mayor atención por parte del profesor, y esto es necesario incorporarlo al enfoque propuesto a fin de reducir aún más la tasa de reprobación.

Aunque las calificaciones obtenidas por el profesor en la evaluación que le hacen sus estudiantes son buenas, no se puede afirmar que esto derive de la estrategia propuesta, las actividades de aprendizaje elaboradas o de la herramienta de pruebas automáticas empleada. Es necesario avanzar a una segunda etapa donde todos los profesores sigan el mismo enfoque y usen el mismo material educativo y tecnológico, a fin de identificar la influencia de la práctica docente específica.

Después de cinco semestres de trabajo, se tiene un curso de entrenamiento para estudiantes de primer semestre, con más de setenta problemas originales sobre tópicos de Fundamentos de Programación. El curso se liberó bajo licencia abierta Creative Commons (CC BY-NC-SA 4.0) y se puede acceder en la plataforma del programa de Ingeniería de Sistemas de la UFPS (https://uvirtual.cloud.ufps.edu.co/course/view.php?name=VPLTraining). Este curso es ahora un repositorio de problemas y soluciones que se planea comenzar a investigar usando técnicas de análisis de código fuente.

Por ahora se puede afirmar que el uso de pruebas automáticas para evaluar cursos de Programación de Computadores ha resultado una estrategia efectiva en el contexto de Ingeniería de Sistemas de la UFPS, que ayuda a reducir la tasa de reprobación y a mejorar las calificaciones, lo cual es consecuencia de la motivación que genera en los estudiantes el abordar los problemas como un reto de aprendizaje y no como una obligación, lo que se corresponde con el enfoque educativo de aprender haciendo y de aprendizaje basado en retos.

REFERENCIAS

[1] M. Andreessen, “Why Software Is Eating The World”, *The Wall Street Journal,* 20 agosto 2011.

[2] Ministerio de Tecnologías de la Información y las Comunicaciones de Colombia (Mintic), *Estudio Exploratorio Prospectiva de la Industria TI en Colombia.* Mintic, 2015. Disponible en: http://fedesoft.org/estudios/EstudiodeProspectiva2015.pdf

[3] Federación Colombiana de la Industria del Software y Tecnologías Informáticas Relacionadas (Fedesoft), "Informe de Caracterización del sector de Software y Tecnologías de la Información en Colombia", 2015.

[4] Ministerio de Tecnologías de la Información y las Comunicaciones de Colombia (Mintic), "Plan Nacional de Ciencia, Tecnología e Innovación 2017-2022", 2016.

[5] Mintic y Fedesoft, "Estudio de Salarios del Sector de Software y TI de Colombia". 2015.

[6] Y. Bosse and M. A. Gerosa, "Why is Programming so difficult to learn?: Patterns of difficulties related to Programming learning mid-stage", *SIGSOFT Softw Eng Notes*, vol. 41, No. 6, pp. 1-6, 2016.

[7] R. Pea and D.-M. Kurland, *On the Cognitive Prerequisites of Learning Computer Programming.* Washington, DC: National Inst. of Education (ED), 1983.

[8] G. M. Weinberg, *The Psychology of Computer Programming*. New York: John Wiley & Sons, Inc., 1985.

[9] M. Kölling, "The problem of teaching object-oriented programming Part I: Environments", *J. Object-Oriented Program.,* vol. 11, No. 9, pp. 6-12, 1999.

[10] J. Villalobos y R. Casallas, "Looking for a new approach to teach/learn a first computer-programming course", *International Conference on Engineering and Computer Education (ICECE),* Madrid, España, 2005.

[11] J. Villalobos y R. Casallas, "Proyecto CUPI2: Un enfoque multidimensional frente al problema de enseñar y aprender a programar", *Rev. Investig. UNAD,* vol. 8, N.º 2, 2009.

[12] R. Queirós y J. P. Leal, "Programming Exercises Evaluation Systems: An Interoperability Survey", in *Proceedings of the 4th International Conference on Computer Supported Education (CSEDU-2012),* pp. 83-90.

[13] F. B. Flórez, R. Casallas, M. Hernández, A. Reyes, S. Restrepo and G. Danies, "Changing a generation's way of thinking: Teaching computational thinking through programming", *Rev. Educ. Res.,* vol. 87, No. 4, pp. 834-860, 2017.

[14] G. Y. Lázaro, A. M. Delgado y F. H. Vera, "Desarrollo e implementación de un marco de trabajo para el entrenamiento en Programación Competitiva", *Rev. Univ., Cienc. Tecnol.,* vol. 20, N.º 79, pp. 69-74, 2016.

[15] A. Chan, J. Cao, C.-K. Liu and W. Cao, "Design and Implementation of VPL: A Virtual Programming Laboratory for Online Distance Learning", in *Advances in Web-Based Learning - ICWL 2003,* vol. 2783, pp. 509-519.

[16] C. Douce, D. Livingstone and J. Orwell, "Automatic Test-based Assessment of Programming: A review", *J. Educ. Resour. Comput.,* vol. 5, No. 3, sep. 2005.

[17] P. Ihantola, T. Ahoniemi, V. Karavirta and O. Seppälä, "Review of recent systems for Automatic Assessment of Programming Assignments", in *Proceedings of the 10th Koli Calling International Conference on Computing Education Research,* New York, USA, 2010, pp. 86-93.

[18] J. C. Rodríguez, E. R. Royo y Z. J. Hernández, "VPL: Laboratorio Virtual de Programación para Moodle", en *Jornadas de Enseñanza Universitaria de la Informática,* Santiago de Compostela, 2010.

[19] Instituto Tecnológico y de Estudios Superiores de Monterrey (ITESM), Edu Trends - Aprendizaje Basado en Retos. Observatorio de Innovación Educativa, Tecnológico de Monterrey, 2016. [En línea]. Disponible en: https://observatorio.tec.mx/edutrendsabr

[20] S. Halim and F. Halim, *Competitive Programming,* 3.ª ed., Singapore, 2013.

[21] ICPC, *ICPC History.* 2017.

[22] ACIS, Consolidado Maratones de Programación Colombia. 2017. [En línea]. Disponible en: https://acis.org.co/portal/content/consolidado-de-maratones-de-programacion

[23] Red de Programación Competitiva (RPC), Sitio Red de Programación Competitiva. 2017. [En línea]. Disponible en: http://redprogramacioncompetitiva.com/

[24] Colombian Collegiate Programming League (CCPL). 2017. [En línea]. Disponible en: https://www.programmingleague.org/

[25] A. M. Delgado y G. Y. Lázaro, Desarrollo e implementación de un marco de trabajo para el grupo de estudio en programación competitiva del programa de ingeniería de sistemas de la Universidad Francisco de Paula Santander (archivo electrónico). San José de Cúcuta: Universidad Francisco de Paula Santander, 2018.

[26] S. K. Robinson and I. S. Torsun, "The Automatic Measurement of the Relative Merits of Student Programs", *SIGPLAN Not,* vol. 12, No. 4, pp. 80-93, abr. 1977.

[27] M. J. Rees, "Automatic Assessment Aids for Pascal Programs", *SIGPLAN Not,* vol. 17, No. 10, pp. 33-42, oct. 1982.

[28] J. C. Rodríguez, E. R. Royo and Z. J. Hernández, "Fighting plagiarism: Metrics and methods to measure and find similarities among source code of computer programs in VPL", in *Proc. EDULEARN11*, 2011, pp. 4339-4346.

[29] K. Beck, *Test Driven Development: By Example.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[30] R. C. Martin, "Professionalism and Test-Driven Development", *IEEE Softw,* vol. 24, No. 3, pp. 32-36, may 2007.

[31] J. C. Rodríguez del Pino, "Sitio Web VP", 2017. [En línea]. Disponible en: https://moodle.org/plugins/stats.php?plugin=mod_vpl

[32] A. C. Vélez y A. Vélez, *Creatividad e inventiva: Retos del siglo XXI.* Medellín: Universidad de Antioquia, 2014.

[33] J. Rogalski and R. Samurçay, "Acquisition of Programming Knowledge and Skills", in *Psychology of Programming,* J. M. Hoc, T. R. G. Green, R. Samurçay and D. J. Gilmore, Eds., London: Academic Press, 1990, pp. 157-174.

[34] R. Scherer, "Learning from the Past–The Need for Empirical Evidence on the Transfer Effects of Computer Programming Skills", *Front. Psychol.,* vol. 7, p. 1390, 2016.

[35] P. J. Clarke, D. Davis, T. M. King, J. Pava and E. L. Jones, "Integrating Testing into Software Engineering Courses Supported by a Collaborative Learning Environment", *Trans. Comput. Educ.,* vol. 14, No. 3, pp. 18:1-18:33, oct. 2014.

[36] S. A. Brian, R. N. Thomas, J. M. Hogan and C. Fidge, "Planting Bugs: A System for Testing Students' Unit Tests", en *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education,* New York, USA, 2015, pp. 45-50.

[37] Y. Bosse and M. A. Gerosa, "Difficulties of Programming Learning from the point of view of students and instructors", *IEEE Lat. Am. Trans.,* vol. 15, No. 11, pp. 2191-2199, 2017.

[38] Y. Bosse, D. Redmiles and M. A. Gerosa, "Pedagogical Content for Professors of Introductory Programming Courses", in *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education,* 2019 pp. 429-435.

[39] M. Kölling, B. Quig, A. Patterson and J. Rosenberg, "The BlueJ system and its pedagogy", *J. Comput. Sci. Educ. Spec. Issue Learn. Teach. Object Technol.,* vol. 13, No. 4, 2003.

[40] D. Barners and M. Kölling, *Object First with Java, A practical Introduction using BlueJ.* 6th ed., Pearson, 2016.

Capítulo IV
Criterios y Métricas para Evaluar la Seguridad en Aplicaciones Web: Metodología MESW

Alexánder Barinas López, abarinas@jdc.edu.co
Docente-investigador, Fundación Universitaria Juan de Castellanos, Tunja-Colombia
John A. Bohada, jbohada@jdc.edu.co
Docente-investigador, Fundación Universitaria Juan de Castellanos, Tunja-Colombia
Iván Andrés Delgado, idelgado@jdc.edu.co
Docente-investigador, Fundación Universitaria Juan de Castellanos, Tunja-Colombia

I. INTRODUCCIÓN

En la actualidad, la información es el activo más importante para el desarrollo de las actividades en una organización. Con la masificación de internet las aplicaciones web son cada vez más populares y complejas, y brindan abundantes beneficios a las empresas en cuanto a posicionamiento en el mercado, publicidad, e información sensible para llevar a cabo la ejecución de sus procesos.

Además, es innegable que cada día las organizaciones dependen en mayor medida de la información y la tecnología, y que ambas, actualmente, soportan los sistemas de información. Pero no siempre ha sido así. En años anteriores, la protección de la información era más sencilla, ya que las grandes plataformas de datos actuaban de forma independiente sin conectividad alguna, las arquitecturas existentes eran totalmente centralizadas y las terminales tenían capacidades de procesamiento limitadas; en cambio, en la actualidad, esa posibilidad se ha incrementado notablemente con la diversidad de redes y plataformas tanto internas como externas, y la información está cada vez más expuesta a las diferentes amenazas y ataques informáticos [1] [2], cualquier dispositivo conectado a la internet corre el riesgo de ser atacado desde computadoras, servidores, teléfonos celulares, entre otros; asimismo, la mayoría de las aplicaciones y servicios web son susceptibles a un conjunto de ataques debido a las vulnerabilidades originadas en los defectos del diseño y la implementación de las aplicaciones, en la programación descuidada de las rutinas, en la escasa implementación de medidas de control de acceso o en la falta de validación de los datos de entrada o incluso en ataques originados por ingeniería social [3].

Por ello, toda organización debe contar con políticas e instrumentos que garanticen la valoración de los riesgos a los cuales está expuesta su información, implementar procedimientos de solución que minimicen la afectación que estos riesgos puedan ocasionar, y crear o fortalecer las áreas de TI que permitan hacer seguimiento y control para garantizar la integridad, confidencialidad y protección de su activo más valioso: la información [4].

Por tanto, el objetivo de esta investigación es presentar el diseño de una metodología de evaluación con base en criterios y métricas de acuerdo con normas y estándares de seguridad, que permita estimar el nivel de seguridad de una aplicación web, de manera que les sirva a los desarrolladores para tomar decisiones y medidas preventivas en el desarrollo de sus aplicaciones.

II. MATERIALES Y MÉTODOS

Para el desarrollo de esta investigación, en primer lugar, se realizó una indagación bibliográfica que permitiese una contextualización de este artículo orientada a la seguridad web a partir de su descripción y de los proyectos, aplicaciones y trabajos desarrollados. En segundo lugar, y basado en dicha contextualización, se propone una novedosa metodología que permite evaluar o

estimar el nivel de seguridad de una determinada aplicación web. Dicha metodología se describe y, además, muestra el resultado de su ejecución en la aplicación, fase por fase.

III. RESULTADOS

A. *Fundamento Teórico*

La seguridad es un aspecto primordial que todo sistema debe incorporar con el fin de proteger los activos de cualquier amenaza. Es importante conocer los rasgos inherentes a la seguridad de la información y la protección de datos. Sin embargo, antes de abordar la problemática existente en las aplicaciones web, es necesario describir los fundamentos teóricos más relevantes que servirán como base para el desarrollo de la metodología propuesta, los cuales se presentan a continuación:

1. *Seguridad Web:* Actualmente, internet está integrado por un conjunto de servicios y aplicaciones que funcionan en la red, desde comercio electrónico, transferencia de archivos, transacciones que mueven enormes cantidades de dinero, búsquedas en la web, correo electrónico, hasta sitios de redes sociales que contienen información importante y confidencial de sus miembros [5].

Mientras aumenta de forma considerable el número de conexiones a la red, así como la migración de aplicaciones y de usuarios a los servicios de internet, crecen las amenazas y sus consecuencias y las medidas tomadas para contrarrestarlas y por tanto la demanda de servicios seguros en la web (ver Tabla 1). Sin lugar a dudas, el aspecto más crítico de la seguridad en internet está en los actores que intervienen directamente en su uso y en la prestación de los servicios, es decir, los usuarios y los servidores web.

Tabla 1. Amenazas, consecuencias y contramedidas a la seguridad web

	Amenazas	Consecuencias	Contramedidas
Integridad	- Virus, puertas traseras y caballos de Troya - Modificación de datos de usuario - Codificación descuidada - Reemplazos o borrados accidentales - Modificación de tráfico de mensajes en tránsito	- Pérdida de información - Compromiso de la máquina - Vulnerabilidad ante todas las otras amenazas	- Cifrado de datos - Sistemas de detección de intrusos - Verificaciones de resúmenes
Confidencialidad	- Monitoreo de la red - Captura de información acerca de la configuración de la red - Acceso no autorizado a archivos - Acceso remoto no autorizado a bases de datos	- Pérdida de información - Pérdida de confidencialidad	- Cifrado de datos - Clasificación de datos - Proxies Web
Disponibilidad	- Destrucción de procesos de usuario - Ataques de denegación de servicio (Denial Of Service o DoS) - Inhabilitar disco o memoria - Cortes en las líneas de comunicaciones	- Disrupción - Anonimidad - Evita que los usuarios hagan su trabajo	- Implantación de sólidos mecanismos de control de acceso - Implantación de sistemas redundantes en aplicaciones críticas - Copias de respaldo de la información vital
Autenticación	- Suplantación de usuarios legítimos - Falsificación de datos	- Representación errónea del usuario - Creencia en la validez de la información falsa	- Técnicas criptográficas - Autenticación de usuarios - Autorización de usuarios
No repudio	- Posibilidad de repudio - Falsificación del origen del envío	- Robos o sabotaje - Fraude electrónico	- Cifrado - Firma digital o certificado digital - Control de acceso

Fuente: Tomado de [6].

Como es sabido, las fallas de seguridad en los servidores web en muchos casos son ocasionadas por descuidos en los mecanismos de protección, fallas en los accesos de control y protección a las aplicaciones y a los datos [7], sin embargo, la razón de la mayoría de fallas de seguridad son las aplicaciones que no se desarrollan de manera profesional, y en varios casos sus autores no tienen conocimiento profundo de los problemas asociados con la seguridad.

Asimismo, los tres propósitos que con mayor frecuencia persiguen los atacantes a sitios web son los siguientes [8], [9]:

- Dejar el sitio fuera de servicio, lo que se conoce como Denegación de Servicio (DoS, por sus siglas en inglés). Los motivos que llevan a este tipo de ataque pueden ser: dejar sin acceso a los usuarios legítimos e imposibilitar el acceso a los servicios y recursos de una organización, o simplemente para atentar contra la disponibilidad del sistema.

- Atentar contra la integridad del sistema modificando o alterando sin autorización los programas, lo que conduce a provocar fallos en el sistema que afectan los procesos y la imagen de una organización.

- Obtener información no autorizada con el objetivo de captar información sensible de las personas, y realizar espionaje de interés para la competencia, como archivos de cuentas bancarias, lo que afecta la confidencialidad de la información.

Teniendo en cuenta que las causas de un ataque son diversas, desde un pirata informático que busca poner a prueba sus destrezas técnicas y lograr captar información sensible, hasta los actos de ciberterroristas que pueden llegar a impedir el normal funcionamiento de una organización. De esta manera, el éxito de un ataque va a depender del minucioso análisis al comportamiento del sitio web, de las debilidades del sitio o de un pobre control de acceso.

2. Iniciativas para la seguridad web: Actualmente hay varias iniciativas que velan por prevenir y ayudar a la seguridad web de las organizaciones [4], ejemplo de ello son OWASP, NIST 800-115, ISSAF y MAGERIT. El proyecto Abierto de Seguridad de Aplicaciones Web (OWASP, por sus siglas en inglés) [10], [11], es uno de los que más relevancia tienen y es una comunidad de código abierto dedicada a determinar y combatir las causas que hacen que el software sea inseguro. Este organismo sin ánimo de lucro conforma una serie de guías y proyectos relacionados con la implementación de la seguridad principalmente en entornos web, además concentra sus esfuerzos en la gestión de metodologías de apoyo a las organizaciones para que desarrollen, adquieran y mantengan aplicaciones en las que se pueda confiar. Para el caso de las aplicaciones web, la iniciativa OWASP presenta una guía importante a tener en cuenta para la seguridad de este tipo de aplicaciones, ya que este proyecto continuamente realiza estudios de las vulnerabilidades y ataques. En la Tabla 2 se describen las principales vulnerabilidades publicadas por este proyecto [10] [11].

Tabla 2. Principales vulnerabilidades en aplicaciones web

Vulnerabilidad	Descripción
A1 - Inyección	Corresponde a las inyecciones de código, siendo las inyecciones SQL unas de las más comunes.
A2 – Pérdida de autenticación	Corresponde al mal manejo de las sesiones en aquellas aplicaciones que utilizan autenticación.
A3 – Exposición de datos sensibles	Se refiere a la protección incorrecta de datos críticos, por ejemplo, números de tarjetas de crédito, contraseñas, etc.
A4 – XML External Entities (XXE)	Sucede cuando las entidades externas se pueden utilizar para divulgar archivos internos, posibilitando la ejecución remota de código y ataques de denegación de servicio.

Vulnerabilidad	Descripción
A5 – Pérdida de control de acceso	Puede derivar en un acceso no autorizado a información crítica debido a fallos en la restricción a los usuarios autenticados.
A6 – Configuración de seguridad incorrecta	Corresponde a configuraciones no adecuadas que pueden impactar en la seguridad de la propia aplicación.
A7 – Secuencia de comandos en sitios cruzados (XSS)	Ocurre cuando existe validación pobre de la información ingresada por el atacante.
A8 – Deserialización insegura	Esta vulnerabilidad puede permitir la ejecución remota de código en servicios web, como ataques de repetición, de inyección y de escalamiento de privilegios.
A9 – Uso de componentes con vulnerabilidades conocidas	Corresponde a la explotación de librerías, *frameworks* y otros componentes vulnerables por parte de un atacante, con el fin de obtener acceso o combinar con otros ataques.
A10 – Registro y monitorización insuficientes	Permite a los atacantes mantener la persistencia, pivotar hacia más sistemas y manipular, extraer o destruir datos.

Fuente: Tomado de [10].

Por su parte, la guía técnica para la evaluación y pruebas de seguridad de la información (NIST SP 800-115, por sus siglas en inglés), presenta cuatro fases: planificación, descubrimiento, ejecución y reportes. En la fase de planificación se realiza una aprobación de la prueba por parte de la gerencia y se identifican su alcance y objetivos; en la fase de descubrimiento se recopila la información y se analizan las vulnerabilidades que se presentaron; en la fase de ejecución del ataque se comprueban las vulnerabilidades previamente detectadas y, si es necesario, se hace un proceso de mitigación de las mismas; finalmente, en la fase de reportes o presentación del informe se documentan todas las fases y las acciones o resultados que se van obteniendo de cada una de ellas y se hacen las posibles recomendaciones. Cada una de las fases se hace en una secuencia, al llegar a la segunda y tercera fases (ataque y descubrimiento) se presenta un bucle de retroalimentación, momento en el cual se deben realizar las pruebas y el análisis sobre múltiples sistemas para determinar el nivel de acceso que puede tener un atacante.

De igual forma, la plataforma para la evaluación de la seguridad de los sistemas de información (ISSAF, por sus siglas en inglés), se compone de tres fases y nueve pasos cíclicos para el desarrollo de su proceso, las fases corresponden a: planificación y preparación, evaluación, y reportes y limpieza y destrucción de información. En la fase de planificación y preparación se deben planear los aspectos más importantes de las pruebas que se llevarán a cabo antes de ejecutarse y firmar un acuerdo; en la fase de evaluación se realiza un enfoque por capas en el que cada una de ellas presenta un mayor nivel de acceso a los activos de información, y en la fase final se debe generar un informe con los resultados obtenidos y las recomendaciones pertinentes. Adicionalmente, la información que se almacenó en los sistemas de prueba debe ser eliminada o, en su defecto, mencionada en el informe técnico para su posterior eliminación.

Finalmente, la Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información (Magerit) fue desarrollada por un ente gubernamental como respuesta a la percepción de que la administración, y en general toda la sociedad, depende de forma creciente de las tecnologías de la información para el cumplimiento de su misión [12]; además, se afirma que el análisis del riesgo no es más que un proceso sistemático para estimar la magnitud de los riesgos a los que está expuesta una organización, y este análisis permite determinar cómo es, cuánto vale y cuán protegidos se encuentran los activos informáticos. Los objetivos que Magerit persigue son: concienciar a los gerentes de las organizaciones sobre los riesgos que tiene la información y la importancia de gestionarlos, ofrecer un método sistemático para analizar los riesgos derivados del uso de tecnologías de la información y la comunicación, descubrir los riesgos y mantenerlos bajo control, y preparar la organización para

procesos de evaluación, auditoría, certificación o acreditación, según sea el caso.

B. *Referentes en Seguridad Web*

Para llevar a cabo esta investigación, se referenciaron algunos documentos y artículos relacionados con el tema de seguridad, que corresponden a estudios adelantados en torno a la problemática y que permitieron dar soporte a la presente investigación.

En [13] se describen los pasos a seguir para establecer un modelo SGSI en las instituciones de educación superior, teniendo como base el estándar ISO 27001.

En [14] los autores proponen el diseño y la construcción de un modelo para evaluar la seguridad en ambientes virtuales de aprendizaje (AVA), a partir de la identificación de criterios de seguridad propuestos en guías, normas y estándares. Una vez establecidos los criterios, se analizaron métricas que permiten cuantificar dicha relación para facilitar la validación del modelo propuesto, el cual fue aplicado a los ambientes Moodle y Dokeos, dos plataformas de código abierto y distribución libre, lo que permite llevar a cabo un juicio de valor confiable tanto cuantitativo como cualitativo sobre el estado de seguridad de los AVA [14].

En [15] se analizan los principales ataques que se han producido en los últimos años y que han comprometido las aplicaciones web, sus bases de datos, los usuarios, entre otros; asimismo, los autores investigan la tendencia que tienen los ataques a las aplicaciones web en las diferentes categorías, con el objetivo de saber qué vulnerabilidades están comúnmente en riesgo y utilizar ese análisis en la creación de aplicaciones web más seguras en el futuro y poder guiar a los desarrolladores web a tomar medidas preventivas [15].

En [16] se presentan estudios sobre la detección de vulnerabilidades Cross Site Scripting (XSS), pero muy pocos se han centrado en su eliminación, por lo cual en este documento los autores enfatizan tanto en los métodos como en las herramientas para eliminar vulnerabilidades XSS; los autores dentro del estudio también muestran una descripción de las diferentes técnicas o modelos capaces de disminuir o filtrar en el momento vulnerabilidades en aplicaciones web, e invitan a los investigadores a centrarse más en su identificación y eliminación antes de hacer una implementación en las web [16].

C. *Metodología para Evaluar la Seguridad Web (MESW)*

Antes de describir la metodología y cada una de sus fases, se indica qué aplicación se tomó como base de prueba para generar los resultados presentados en esta sección.

1. Aplicación web para realizar evaluación de seguridad: Para realizar las pruebas de seguridad, se tomó como referente una aplicación web desarrollada en el grupo de investigación Ciencia, Innovación y Tecnología (CIyT), de la Facultad de Ingeniería y Ciencias Básicas de la Fundación Universitaria Juan de Castellanos. Por seguridad, el dominio de dicha aplicación no será referenciado.

La aplicación trata de una solución web para una tienda de venta de artesanías en línea. La aplicación cuenta con dos tipos de usuario: usuario general y usuario administrador del sitio. El usuario general podrá registrarse e iniciar sesión en el sitio, consultar los artículos más recientes, los artículos por su tipo, artículos por precio (menor a mayor), buscar artículos, ver el detalle de cada artículo, agregar un artículo al carrito de compras y realizar la compra. Por otro lado, el usuario administrador cuenta con privilegios tales como: ingresar al sitio, consultar los artículos más recientes, artículos por su tipo, artículos por precio (menor a mayor), buscar artículos, ver el detalle de cada artículo, tener un menú de administración, registrar un nuevo artículo, ver el listado de artículos, de clientes y de ventas, entre otras funcionalidades.

Desde el punto de vista técnico, la aplicación fue desarrollada bajo un servidor ApacheServer, lenguaje web PHP5, base de datos MySQL Server.

2. Descripción y uso de la metodología MESW: La metodología para evaluar la seguridad de aplicativos web (MESW) está compuesta por cinco fases: estado del sistema, diseño de pruebas, identificación de vulnerabilidades, evaluación de la seguridad y análisis de resultados (Figura 1).

Las fases de MESW son consecutivas y están compuestas por diferentes actividades que se llevan a cabo para medir o estimar el nivel de seguridad de una determinada aplicación web, de manera que sirva a los desarrolladores para tomar decisiones y medidas preventivas en el desarrollo de sus aplicaciones.

Fig. 1. Diseño metodológico de evaluación de seguridad

Fase 1. Estado del sistema: En esta fase se realiza la identificación de la aplicación que se va a evaluar, asimismo, se lleva a cabo el reconocimiento y registro de las características del equipo donde se encuentra alojada la aplicación web, es decir, el servidor web. Las características del servidor web donde se encuentra alojada la aplicación objeto de evaluación son las que se muestran en la Tabla 3.

Tabla 3. Características del servidor web

Hardware/ Software	Descripción
CPU	Intel Core i5 1.7 GHz - 2.4 GHz
RAM	8 Gb
Sistema operativo	Windows 10 Home
Stack	WAMP
Servidor web	Apache Server
Base de datos	MySQL Server
Lenguaje web	PHP 5
Aplicación web	

Fuente: Elaboración propia.

Posteriormente, se llevan a cabo las pruebas de seguridad, las cuales se hacen desde un equipo local con sistema operativo Windows y ejecutando las herramientas de seguridad propuestas y descritas en la siguiente fase.

Fase 2. Diseño de pruebas: En esta fase se deben realizar tres actividades: 1: Identificar herramientas de seguridad que permitan evaluar, inicialmente, la seguridad de la aplicación web. 2: Diseño de métricas y criterios de seguridad de acuerdo con normas y estándares de seguridad. 3: Con base en las métricas propuestas, se establecen valores de medición según las normas o estándares usados en la actividad 2.

En este sentido, el desarrollo de las tres actividades correspondió a lo siguiente:

Actividad 1: Se realizó una indagación de herramientas que permitiesen evaluar la seguridad web del aplicativo objeto de estudio. En este sentido, se identificaron las siguientes tres herramientas: SQLMap (herramienta automática de inyección SQL) [17], W3af (*frame* de testeo de auditoría y ataques a servicios web) [18], IronWASP (plataforma avanzada de testeo de seguridad web) [19].