Teaching Model of Object Oriented Programming based on Video Game Development and applying Software Project Management and Scrum

Marco Aedo López, Mg.[1], Eveling Castro Gutiérrez, Mg.[1]
[1]Universidad Nacional de San Agustín de Arequipa, Perú, maedol@unsa.edu.pe, ecastro@unsa.edu.pe

***Abstract*– This article presents the proposal of an object-oriented programming teaching model based on the development of video games and applying software project management techniques and Scrum, generating a motivating environment for learning. Teaching the object-oriented paradigm creates significant challenges that are not easy to deal with, even with a basic understanding of computer programming. It is shown that the teaching-learning process applying the proposed model to teach object-oriented programming concepts is effective and motivating, validating the model in video game development projects, validation carried out at the National University of San Agustín de Arequipa - Peru in the Professional School of Systems Engineering, in the Fundamentals of Programming 2 course and takes into account the learning styles of post-millennials or Generation Z.**

***Keywords*-- Object Oriented Programming, Teaching Model, Teaching Motivation, Software Project Management, Scrum.**

Digital Object Identifier (DOI):
http://dx.doi.org/10.18687/LACCEI2022.1.1.229
ISBN: 978-628-95207-0-5 **ISSN:** 2414-6390

Modelo de Enseñanza de la Programación Orientada a Objetos basado en el Desarrollo de Videojuegos y aplicando Gestión de Proyectos de Software y Scrum

Marco Aedo López, Mg.[1], Eveling Castro Gutiérrez, Mg.[1]
[1]Universidad Nacional de San Agustín de Arequipa, Perú, maedol@unsa.edu.pe, ecastro@unsa.edu.pe

Abstract*– Este artículo presenta la propuesta de un modelo de enseñanza de la programación orientada a objetos basado en el desarrollo de videojuegos y aplicando técnicas de gestión de proyectos de software y Scrum, generando un ambiente motivador para su aprendizaje. La enseñanza del paradigma de la orientación a objetos genera retos importantes que no son sencillos de tratar, incluso teniendo un conocimiento básico de la programación de computadoras. Se demuestra que el proceso enseñanza-aprendizaje aplicando el modelo propuesto para enseñar conceptos de programación orientada a objetos es efectiva y motivadora, realizando la validación del modelo en proyectos de desarrollo de videojuegos, validación realizada en la Universidad Nacional de San Agustín de Arequipa – Perú en la Escuela Profesional de Ingeniería de Sistemas, en el curso de Fundamentos de Programación 2 y toma en consideración los estilos de aprendizaje de los post-millennials o Generación Z.

***Keywords*-- Programación Orientada a Objetos, Modelo de Enseñanza, Motivación en la Enseñanza, Gestión de Proyectos de Software, Scrum.**

I. Introducción

Actualmente hay numerosas experiencias que muestran que la naturaleza lúdica y visual de los videojuegos aporta en la mejora del proceso de enseñanza-aprendizaje de la programación básica de computadoras, coadyuvando a los estudiantes a comprender conceptos abstractos que de lo contrario serían más difíciles de dominar [1],[2],[3],[4].

Sin embargo, en la enseñanza de los conceptos de la programación orientada a objetos y otros temas avanzados de programación siempre existe un alto grado de complejidad, que genera un índice considerable de estudiantes desaprobados en el curso donde se imparten tales tópicos.

Para enfrentar tal problemática, a lo largo de los años se empezaron a utilizar técnicas basadas en entornos lúdicos para la enseñanza de algunos conceptos básicos de la programación orientada a objetos [5], mejorando los resultados, creando un ambiente motivador para los estudiantes y despertando en ellos el deseo de dominar y profundizar en dichos conocimientos.

También se debe destacar que en la actualidad, los estudiantes universitarios pertenecen a la llamada Generación Z o post-millenials, que son nativos digitales puros, que utilizan la tecnología y los gadgets como parte fundamental de sus actividades cotidianas desde sus primeros años de vida.

Digital Object Identifier (DOI):
http://dx.doi.org/10.18687/LACCEI2022.1.1.229
ISBN: 978-628-95207-0-5 **ISSN:** 2414-6390

De tal forma que las estrategias de enseñanza a utilizar con estos estudiantes deben ser adecuadas a su estilo de aprendizaje y es por eso que se plantea un modelo de enseñanza basado en el desarrollo de videojuegos y aplicando técnicas de gestión de proyectos de software y Scrum.

Así, para el 2019, se planteó la aplicación de un proyecto de desarrollo de software de videojuego para lograr un proceso enseñanza-aprendizaje más efectivo [5], experiencia que fue mejorando con actividades de aprendizaje más completas y complementadas con actividades de gestión de proyectos de software y Scrum para el 2020 y el 2021.

La elección de que sea un proyecto de desarrollo de videojuegos no es arbitraria, se basa en el hecho de que parte de la tecnología que los estudiantes utilizan diariamente la constituyen los videojuegos, situación confirmada por una encuesta realizada el primer día de clases a los estudiantes ingresantes desde hace 9 años, la pregunta es "¿quiénes utilizan videojuegos?" y se encuentra que casi todos son usuarios de dichas aplicaciones. Tabla I.

TABLA I
RESPUESTAS A UTILIZACIÓN DE VIDEOJUEGOS

Año	Si	No
2013	90%	10%
2014	92%	8%
2015	95%	5%
2016	97%	3%
2017	97%	3%
2018	96%	4%
2019	98%	2%
2020	97%	3%
2021	98%	2%

Como parte de la justificación también se considera que, de acuerdo al plan de estudios, el perfil del egresado de nuestra escuela profesional indica que el egresado estará preparado para el desarrollo de software de calidad, considerando 3 especializaciones: Sistemas de Información Empresariales, Sistemas para Plataformas Móviles, y Videojuegos y Sistemas de Entretenimiento. Esta especialización se realiza durante los 3 últimos semestres del plan de estudios vigente.

Así, durante los últimos años, una pregunta de la misma encuesta es "¿qué tipo de software está más interesado en desarrollar en su vida profesional?" Tabla II.

TABLA II
RESPUESTAS A PREFERENCIA DE TIPO DE SOFTWARE A DESARROLLAR

Año	Sistemas de Información Empresariales	Sistemas para Plataformas Móviles	Videojuegos y Sistemas de Entretenimiento
2016	20%	25%	55%
2017	20%	26%	54%
2018	21%	24%	55%
2019	20%	24%	56%
2020	18%	25%	57%
2021	15%	29%	56%

Se muestra que el interés está guiado preferentemente al desarrollo de videojuegos, superando a las otras 2 especialidades. De tal forma, se puede observar que los estudiantes, casi en su totalidad, son usuarios de videojuegos y que la mayoría tienen la intención de guiar su formación profesional al desarrollo de los mismos. Esta situación genera una motivación intrínseca que contribuye al dominio de los conocimientos a aprender y aplicar [6],[7].

En este artículo se presenta un modelo de enseñanza basado en el desarrollo de videojuegos y aplicando la gestión de proyectos de software y Scrum para la enseñanza de los conceptos de la programación orientada a objetos, modelo con el que se aplicaron y consolidaron los conocimientos adquiridos en la parte teórica del curso y se alcanzó un aprendizaje más efectivo. Se debe recalcar que este artículo es la extensión de la tesis de maestría en ciencias [8].

El resto del artículo está organizado de la siguiente manera. En la sección II se muestran los estilos de aprendizaje de los estudiantes universitarios de la actualidad en nuestra realidad. En la sección III se presenta el marco de trabajo general del modelo de enseñanza. En la sección IV se realiza la descripción del modelo de enseñanza. En la sección V se muestra la evaluación y análisis de resultados, donde se indican los resultados obtenidos del rendimiento académico de los estudiantes en el curso, donde se compara el rendimiento cuando utilizamos el enfoque tradicional de enseñanza aplicado en años anteriores (2013 – 2017), comparándolo con un enfoque lúdico de enseñanza aplicado durante el 2018, luego en comparación con un enfoque lúdico y aplicando un proyecto de desarrollo de software de videojuego para el 2019, y al final se compara con proyectos de desarrollo de software de videojuego aplicándoles gestión de proyectos y Scrum para el 2020 y el 2021. Finalmente se presentan nuestras conclusiones.

II. ESTILOS DE APRENDIZAJE DE LOS ESTUDIANTES

Es fundamental considerar que los estudiantes universitarios actuales pertenecen a la llamada Generación Z o post-millenials, una generación conformada por nativos digitales puros que utilizan los dispositivos electrónicos y la tecnología como parte fundamental de todas sus actividades desde sus primeros años de vida, y cuya forma de aprender es diferente a la de generaciones anteriores [6].

En consecuencia, aplicar estrategias de enseñanza del siglo pasado con estos estudiantes constituye un grave error, ya que ellos no toleran 2 horas de una clase magistral centrada netamente en el docente, lo auditivo ya no se constituye en su estilo de aprendizaje principal, en cambio predominan los estilos de aprendizaje visual y kinestésico.

Se corrobora la afirmación anterior considerando que los estudiantes pueden tener diferentes estilos de aprendizaje y aplicando el Test desarrollado por Lynn O'Brien, a un universo de 70 estudiantes, ver Fig. 1. Confirmando el resultado de los estudios de [6], donde se comprueba que los estudiantes en nuestro entorno particular han desarrollado un estilo de aprendizaje más visual y kinestésico, superando al aprendizaje auditivo que predominaba en generaciones de estudiantes anteriores.

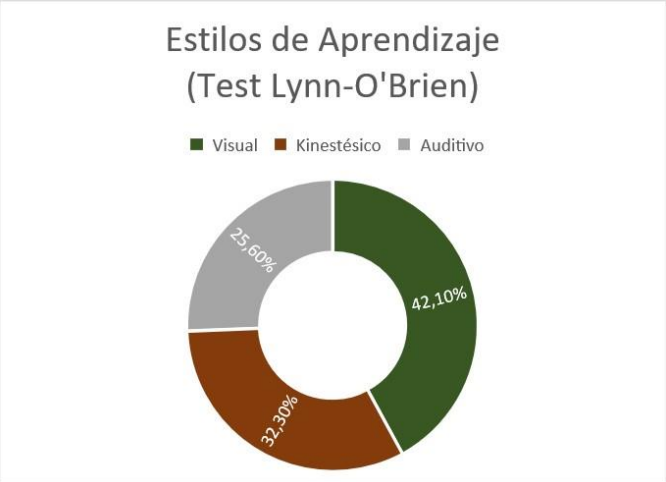


Fig. 1. Estilos de Aprendizaje – Test-Lynn O'Brien, aplicado a 70 estudiantes en la Escuela Profesional de Ingeniería de Sistemas

Los estudiantes actuales absorben rápidamente la información de imágenes y videos, construyen sus conceptos en base a objetos digitales, tienen un comportamiento multitarea ya que adquieren información en simultáneo de diversas fuentes, los estimula el trabajo colaborativo, esperan retroalimentación inmediata, tienen periodos de atención cortos y valoran más las actividades prácticas que aquellas actividades netamente teóricas [7].

Como educadores debemos adaptarnos a estas circunstancias y las estrategias de enseñanza a utilizar con los estudiantes deben adecuarse a su estilo de aprendizaje y a sus intereses. Basados en esto es que se planteó, para el 2019, la aplicación de un proyecto de desarrollo de software de videojuego para lograr un proceso enseñanza-aprendizaje más efectivo, experiencia que fue enriquecida con actividades más completas para el 2020 y el 2021 incluyendo actividades de la gestión de proyectos de software y Scrum para que la enseñanza sea más efectiva.

La aplicación de esta estrategia promueve la idea de aprender haciendo [9], algo que en programación de

computadoras es fundamental, ya que se aprende a programar programando, practicando, poniendo manos a la obra.

Al aplicar este modelo de enseñanza, los estudiantes van construyendo su propio conocimiento [10] y desarrollando el sentido del logro. El hecho de considerar como punto de partida los intereses, necesidades y el estilo de aprendizaje de los estudiantes es fundamental para lograr desarrollar las competencias propuestas y lograr un aprendizaje efectivo.

III. Marco de Trabajo General del Modelo de Enseñanza

A. *Competencias Previas*

Las competencias previas que los estudiantes deben poseer antes de aplicar el modelo de enseñanza, se muestran en la Tabla III.

TABLA III
COMPETENCIAS PREVIAS

1. Identifica, establece e integra los diferentes conceptos de programación reconociendo los componentes y características de un algoritmo.
2. Elabora, crea y codifica algoritmos para la solución de problemas reales en un lenguaje de programación.
3. Aplica, codifica y ejecuta sentencias de selección y repetición apropiadas para la elaboración de programas.
4. Introduce, analiza, y utiliza el concepto de métodos reconociendo su importancia en la programación.

B. *Competencias del Curso*

El modelo de enseñanza propuesto se aplica a un curso que considere las competencias generales y específicas de la Tabla IV y Tabla V.

TABLA IV
COMPETENCIAS GENERALES DEL CURSO

C.c. Diseña responsablemente sistemas, componentes o procesos para satisfacer necesidades dentro de restricciones realistas: económicas, medio ambientales, sociales, políticas, éticas, de salud, de seguridad, manufacturación y sostenibilidad.
C.p. Aplica de forma flexible técnicas, métodos, principios, normas, estándares y herramientas de ingeniería necesarias para la construcción de software e implementación de sistemas de información.

TABLA V
COMPETENCIAS ESPECÍFICAS DEL CURSO

1. Identifica, establece e integra los diferentes conceptos de Arreglos, ArrayList y HashMap, y describe los algoritmos de búsqueda y ordenamiento valorando su importancia
2. Elabora, crea y codifica algoritmos para la solución de problemas reales aplicando los conceptos fundamentales de la Orientación a Objetos, tales como clases propias, objetos, atributos y métodos valorando la importancia del paradigma
3. Aplica, codifica y ejecuta los conceptos avanzados de la Orientación a Objetos, tales como referencias, métodos sobrecargados y constructores valorando su utilización
4. Aplica, codifica y ejecuta otros conceptos de la Orientación a Objetos, tales como atributos y métodos de clase y de instancia
5. Analiza y aplica los mecanismos de agregación, composición, herencia, polimorfismo, clases abstractas e interfaces de la Orientación a Objetos valorando la potencia de su utilización
6. Aplica la programación orientada a eventos y analiza los diferentes componentes gráficos que brinda Java valorando dicho paradigma
7. Concibe el concepto de archivos como un medio de almacenamiento permanente, valorando su importancia

C. *Requerimientos de Tiempo*

En relación a la cantidad de horas de un curso que aplique el modelo propuesto, se recomienda de 2 a 4 horas semanales de teoría, donde se enseñe la base teórica del curso aplicando aprendizaje basado en problemas. En la parte de laboratorio, que es donde se realizarán las actividades de los proyectos de desarrollo de videojuegos, debería contar con 4 horas semanales, preferentemente divididas en 2 bloques de 2 horas.

Sobre la duración de un curso que implemente este modelo de enseñanza, se recomienda la duración de un semestre/ciclo de 16 semanas (adaptable a 15 o 17 semanas).

D. *Requerimientos Tecnológicos*

Para aplicar el modelo de enseñanza se requiere un laboratorio con computadoras (pcs, portátiles, tablets o smartphones) con capacidad para realizar programas en lenguajes de programación que permitan aplicar el paradigma de programación de la orientación a objetos (Java, Python, C#, etc.).

Es importante tener conectividad con internet para realizar investigación y realizar el trabajo colaborativo, así como para cumplir con la presentación de entregables del proyecto.

Es deseable implementar un aula virtual para mantener los entregables de cada sesión de forma ordenada.

IV. Modelo de Enseñanza

A. *Definición general y planificación del Proyecto Guiado (PG) y Proyecto Final (PF)*

El modelo de enseñanza propuesto consta del desarrollo de 2 proyectos. El primero es un proyecto guiado por el docente en el que se desarrollará un videojuego, con una base común para todos los estudiantes y es individual, se denominará PG. Constará de 2 fases que se describirán posteriormente. Se cronograma para las primeras 15 semanas del semestre/ciclo y sigue un proceso iterativo e incremental.

El segundo es un proyecto libre de desarrollo de videojuego, pero enmarcado en la misma temática que el PG, se recomienda formar equipos de 2 estudiantes y donde se debe aplicar la creatividad para el desarrollo de su producto de software, constituye el proyecto final del curso y se denominará PF. Se cronograma para las últimas 4 semanas del semestre/ciclo y también sigue un proceso iterativo e incremental.

Tanto en el PG como en el PF se aplicarán actividades de gestión de proyectos de software y de Scrum.

Las últimas semanas del PG tienen actividades en paralelo con el PF. En la Fig. 2 se puede observar la planificación general de ambos proyectos (PG y PF).

Fig. 2. Planificación del Proyecto Guiado (PG) y del Proyecto Final (PF)

B. Definición del Videojuego del Proyecto Guiado

Inicialmente se explica a los estudiantes que aprenderán los conceptos de la programación orientada a objetos mediante el desarrollo de un videojuego. El objetivo es captar su atención, la motivación es automática dada la sorpresa generada y la expectativa de crear un videojuego propio cuando apenas se encuentran en primer año.

Se les plantea un marco general de trabajo y la planificación de actividades correspondiente, y se explica acerca de la trama del videojuego, así ellos empiezan a relacionarlo con algunos videojuegos que utilizan actualmente o utilizaron alguna vez, despertando su imaginación y creatividad.

El videojuego propuesto puede ser de cualquier género, pero que contenga una variedad de personajes (objetos), con varias características (atributos) y con comportamiento amplio (métodos). Se recomienda el género de estrategia por la riqueza de sus variantes y la aplicación natural de los conceptos de la orientación orientada a objetos.

C. Determinación de las actividades de desarrollo del videojuego (PG)

Las actividades se realizan en el laboratorio semanalmente (2 sesiones de 2 horas para el desarrollo de las actividades propuestas y se realiza una revisión continua). Tienen la opción de consultar con otros estudiantes del grupo de laboratorio, consultar con el docente, se les permite utilizar internet, smartphones y otros dispositivos, consultar con todas las fuentes que deseen, tales como libros físicos y digitales, apuntes de clase, etc. (todo enmarcado en un ambiente de respeto y disciplina).

Por cada sesión se presentan 2 entregables: el avance de la actividad (plazo de entrega al finalizar la sesión) y la actividad completa o versión final (plazo de entrega al día siguiente a medianoche).

Para el avance de la actividad se recomienda que cada estudiante vaya a su ritmo, aunque avance poco, pero que demuestre algún avance. Se realiza en el aula y se recomienda evitar el stress y la deserción, y en cambio, fomentar la motivación y el disfrute de la actividad. El entregable se realiza por medio del aula virtual implementada para el curso (en caso no haya aula virtual, utilizar el correo electrónico). Ver Fig. 3.

Fig. 3. Ejemplo de Actividades Tarea en el Aula Virtual

Para la entrega de la versión final se tiene un plazo considerable (mayor a 24 horas). Se realiza en horario fuera de clase y se procura culminar la actividad propuesta. El entregable se realiza por medio del aula virtual implementada para el curso (en caso no haya aula virtual, utilizar el correo electrónico). Ver Fig. 3.

El PG se divide en 2 fases, la Fase 1 se desarrolla las primeras 12 semanas y un ejemplo de programación para cada semana se puede ver en la Fig. 4, donde se puede observar la dependencia entre las actividade.

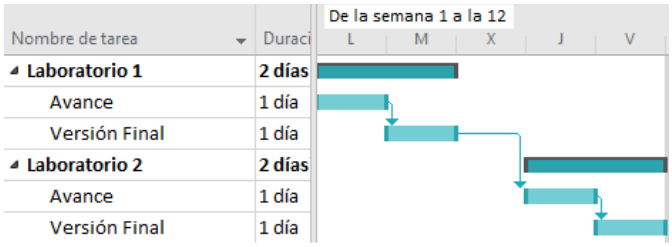


Fig. 4. Diagrama de Gantt del Proyecto Guiado para la Fase 1 (para semanas 1 a 12)

La Fase 2 se desarrolla de la semana 13 a la 15, pero sólo durante la primera sesión semanal. La otra sesión será parte del PF. Ver Fig. 5.

Fig. 5. Diagrama de Gantt del Proyecto Guiado para Fase 2 (para semanas 13 a 15)

Para cada sesión se utiliza una rúbrica de evaluación, un ejemplo sugerido de rúbrica se muestra en la Tabla VI.

TABLA VI
EJEMPLO DE RÚBRICA DE EVALUACIÓN DE SESIÓN DE LABORATORIO

Nivel de Desempeño Insatisfactorio 25%	Nivel de Desempeño en Proceso 50%	Nivel de Desempeño Satisfactorio 75%	Nivel de Desempeño Sobresaliente 100%
No comprende creación y utilización de clases y objetos	Reconoce la creación y utilización de clases y objetos	Aplica la creación y utilización de clases y objetos para resolver problemas	Aplica y propone la mejor solución a un problema aplicando la creación y utilización de clases y objetos

D. Elaboración de Laboratorios (PG)

Los laboratorios deberán cumplir la planificación y temática mostrada en la Tabla VII. La metodología de enseñanza empleada en el curso se basa en que primero se aprenden a usar clases y objetos, luego se aprende a crear clases personalizadas, posteriormente se enseñan conceptos avanzados y finalmente todos los conocimientos adquiridos se aplican a temas más complejos.

Se muestran también los entregables sugeridos por cada sesión de laboratorio.

TABLA VII
PLANIFICACIÓN Y TEMÁTICA DE LOS LABORATORIOS PARA PG

Semana	Nº Lab.	Tema	Objetivo	Entregables
1	1	Arreglos Estándar (Arrays)	Comprender y valorar el uso de arreglos estándar en la solución de problemas. Problemas simples.	Código Fuente
1	2	Arreglos Estándar (Arrays)	Comprender y valorar el uso de arreglos estándar en la solución de problemas. Problemas complejos.	Código Fuente
2	3	Arreglos de Objetos	Comprender y valorar el uso de arreglos de objetos en la solución de problemas.	Código Fuente
2	4	Arreglos de Objetos, búsquedas y ordenamientos	Comprender y valorar las operaciones de búsqueda y ordenamiento en un arreglo de objetos.	Código Fuente
3	5	Arreglos Bidimensionales de Objetos	Comprender y valorar el uso de arreglos bidimensionales de objetos en la solución de problemas.	Código Fuente
3	6	ArrayList	Comprender y valorar el uso de ArrayList en la solución de problemas como una alternativa más flexible a la aplicación de los arreglos estándar.	Código Fuente
4	7	HashMap	Comprender y valorar el uso de HashMap en la solución de problemas como una alternativa más flexible a la aplicación de los arreglos estándar y ArrayList.	Código Fuente
4	8	Definición de Clases de Usuario 1	Comprender y valorar la creación de clases de usuario en la solución de problemas. Crear diagramas de clases UML. Problemas simples.	Código Fuente y Diagrama de Clases de UML
5	9	Definición de Clases de Usuario 2	Comprender y valorar la creación de clases en la solución de problemas. Crear diagramas de clases UML. Problemas intermedios.	Código Fuente y Diagrama de Clases de UML
5	10	Definición de Clases de Usuario 3	Comprender y valorar la creación de clases en la solución de problemas complejos. Crear diagramas de clases UML. Problemas complejos.	Código Fuente y Diagrama de Clases de UML
6	11	Definición de Clases de Usuario 4	Comprender y valorar la creación de clases en la solución de problemas complejos. Crear diagramas de clases UML. Problemas complejos que reutilicen código de laboratorios anteriores.	Código Fuente y Diagrama de Clases de UML
6	12	Miembros de clase	Comprender y valorar la utilización de miembros de clase en la solución de problemas complejos. Crear diagramas de clases UML.	Código Fuente y Diagrama de Clases de UML
7	13	Mecanismos de Agregación y Composición 1	Comprender, valorar y comparar los mecanismos de agregación y composición de clases. Problemas básicos.	Código Fuente y Diagrama de Clases de UML
7	14	Mecanismos de Agregación y Composición 2	Comprender, valorar y comparar los mecanismos de agregación y composición de clases. Problemas complejos.	Código Fuente y Diagrama de Clases de UML
8	15	Mecanismos de Agregación y Composición 3. Miembros de clase	Comprender, valorar y comparar los mecanismos de agregación y composición de clases, combinándolos con miembros de clase.	Código Fuente y Diagrama de Clases de UML
8	16	Mecanismos de Agregación y Composición 4. Miembros de clase. Reutilización	Comprender, valorar y comparar los mecanismos de agregación y composición de clases, combinándolos con miembros de clase. Reutilizando código de laboratorios anteriores.	Código Fuente y Diagrama de Clases de UML
9	17	Mecanismo de Herencia 1	Comprender y valorar el mecanismo de herencia de clases para fomentar la reutilización. Problemas básicos.	Código Fuente y Diagrama de Clases de UML
9	18	Mecanismo de Herencia 2	Comprender y valorar el mecanismo de herencia de clases para fomentar la reutilización. Problemas complejos.	Código Fuente y Diagrama de Clases de UML
10	19	Herencia y Polimorfismo	Comprender, valorar y aplicar el mecanismo de polimorfismo y relacionarlo con el mecanismo de herencia de clases.	Código Fuente y Diagrama de Clases de UML
10	20	Herencia y Polimorfismo. Miembros de clase.	Comprender, valorar y aplicar el mecanismo de polimorfismo y relacionarlo con el mecanismo de herencia de clases y combinarlo con los miembros de clase.	Código Fuente y Diagrama de Clases de UML
11	21	Herencia y Polimorfismo. Miembros de	Comprender, valorar y aplicar el mecanismo de polimorfismo y	Código Fuente y Diagrama

		clase y de instancia.	relacionarlo con el mecanismo de herencia de clases. Comprender, valorar y aplicar los miembros de clase y de instancia según corresponda.	de Clases de UML
11	22	Herencia y Polimorfismo. Clases Interface	Comprender, valorar y aplicar el mecanismo de polimorfismo y relacionarlo con el mecanismo de herencia de clases. Comprender, valorar y aplicar las clases Interface y su implementación.	Código Fuente y Diagrama de Clases de UML
12	23	Herencia y Polimorfismo. Clases Interface y Clases Abstractas	comprender, valorar y aplicar el mecanismo de polimorfismo y relacionarlo con el mecanismo de herencia de clases. Comprender, valorar y aplicar las clases Abstractas, Interface y su implementación.	Código Fuente y Diagrama de Clases de UML
12	24	Interfaces Gráficas de Usuario (GUI) 1	Comprender y valorar el uso de las interfaces gráficas de usuario para crear programas visuales más atractivos.	Código Fuente, Diagrama de Clases de UML y capturas de pantalla
13	25	Interfaces Gráficas de Usuario (GUI) 2. Layouts	Comprender y valorar el uso de las interfaces gráficas de usuario para crear programas visuales más atractivos usando layouts.	Código Fuente, Diagrama de Clases de UML y capturas de pantalla
14	26	Archivos de Texto	Comprender y valorar el uso del almacenamiento permanente usando archivos de texto.	Código Fuente, Diagrama de Clases de UML y capturas de pantalla
15	27	Archivos Binarios y de Objetos	Comprender, valorar y comparar el uso del almacenamiento permanente usando archivos binarios y de objetos.	Código Fuente, Diagrama de Clases de UML y capturas de pantalla

E. Definición del Videojuego del Proyecto Final (PF)

Se planteará a los estudiantes un marco general de trabajo y se explicará acerca del videojuego a desarrollar como proyecto final, despertando su imaginación y creatividad.

El videojuego propuesto puede ser de cualquier género, aunque se recomienda que sea del mismo género del videojuego desarrollado en el PG. Lo fundamental es que contenga variedad de personajes (objetos), con varias características (atributos) y con comportamiento amplio (métodos). Se recomienda el género de estrategia por la riqueza de sus variantes y la aplicación natural de los conceptos de la orientación orientada a objetos.

Deberán considerar el uso de los tópicos de orientación a objetos desarrollados en el curso, además del uso de GUI y de archivos. Podrán reutilizar lo que consideren necesario del PG, pero deberán ser originales en su propuesta. Además, deberán considerar los aspectos de interacción con el usuario, funcionalidad y jugabilidad.

F. Determinación de las actividades de desarrollo del videojuego (PF)

Cada equipo debe tener un nombre establecido por sus integrantes y el videojuego a desarrollar también debe ser bautizado por ellos mismos, creando una identidad de equipo.

Las actividades se dividen en 4 sprints de una semana de duración. El docente deberá elaborar el product backlog inicial con los requerimientos generales del videojuego y ponerlo a disposición de todos los equipos. Los equipos deberán elaborar sus propios product backlog para el videojuego que desarrollarán (respetando los requerimientos establecidos en el product backlog inicial) y también elaborarán los sprint backlog para cada sprint. Los estudiantes cumplirán los roles de Product Owner y serán parte del Equipo de Desarrollo. El docente del curso cumplirá el rol de Scrum Master y también se considerará parte del Equipo de Desarrollo.

El desarrollo del sprint backlog se realizará fuera del aula y fuera de los horarios de clase. La revisión de cada sprint se realizará en el laboratorio en una sesión de 2 horas y se buscará realizar las acciones de control para verificar el cumplimiento del sprint backlog correspondiente y la retroalimentación al equipo con aspectos a mejorar e incluir para el próximo sprint, así como satisfacer dudas. Es recomendable que se realicen Scrums periódicos (diario es deseable). Los entregables de cada sprint serán el sprint backlog, los diagramas UML, el código fuente y el videojuego funcional desarrollado en el sprint correspondiente. Ver Fig. 6.

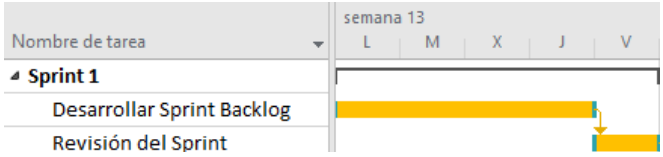


Fig. 6. Planificación de actividades para PF (para semanas 13 a 15)

El equipo mostrará las actividades realizadas en la última semana y comenzará a planear las actividades a realizar para la próxima semana, elaborándose una bitácora de dichas actividades.

Esto genera una retroalimentación inmediata y un seguimiento de proyecto muy necesario dada su inexperiencia comprensible al trabajar en proyectos. Considerar que es importante que el proyecto se realice en un entorno ágil y evitar la excesiva documentación [10].

Al finalizar el cuarto sprint ya se debe tener un software finalizado y se deberá hacer su presentación en público. Se expondrá el videojuego desarrollado ante todos los otros

equipos de desarrollo, el docente del curso, docentes y gamers invitados (estudiantes de otros años, docentes o gamers conocidos). Se les asignará un horario de exposición y un tiempo específico para la exposición (10 minutos) y para preguntas del público (5 minutos), tiempos recomendables pero ajustables dependiendo de la cantidad de equipos. Ver Fig. 7.

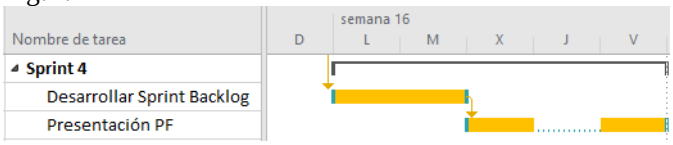


Fig. 7. Planificación de actividades para PF (Semana 16)

En la Tabla VIII se detallan las actividades a realizar y en la Fig. 8 se muestra la planificación detallada para el PF.

TABLA VIII
ACTIVIDADES EN LABORATORIO PARA PF

Semana 13 – 15
Actividades de Revisión del Sprint en laboratorio
Tema: Presentación del avance del proyecto de videojuego realizado en el sprint. Objetivo: dar a conocer el avance realizado hasta este punto por el equipo. Comentario: las soluciones que plantean cada equipo varían, aunque cada equipo empezó de una base común (el PG), toma el camino que considera correcto y justifica sus decisiones. La retroalimentación dada por el docente es muy importante y ayuda a realizar ajustes necesarios a cada proyecto. Entregable: sprint backlog, diagramas UML, el código fuente y el videojuego funcional
Semana 16
Actividades de Presentación Final en laboratorio
Tema: Presentación Final del Proyecto. Objetivo: presentar el producto creado a un público. Enunciado: tienen 10 minutos para realizar una demostración del videojuego desarrollado, además de mostrar los diagramas UML creados y el código escrito. Además, se dispondrá de 5 minutos para preguntas del público y también para recomendaciones. Comentario: el acto de presentar su producto a un público despierta una motivación extra en los estudiantes y la retroalimentación que obtienen hace incluso que ellos sigan perfeccionando su videojuego durante las vacaciones, aunque ya no haya una nota para ello. Entregable: Documentación final del proyecto y videojuego versión final.

G. *Integración del Modelo*

En la Fig. 9 y la Fig. 10 se muestran los esquemas con las actividades y roles para ambos proyectos (PG y PF) respectivamente.

Las actividades del PG y del PF se integran y se muestran en la Fig. 11 y la Fig. 12. Se puede observar la dependencia de actividades dentro de cada uno de los proyectos y también la dependencia entre actividades del PF con actividades del PG.

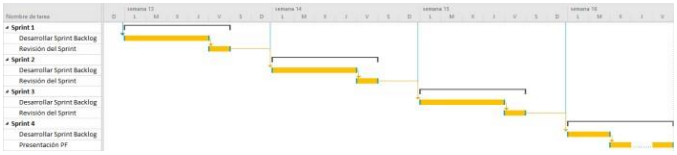


Fig. 8. Planificación detallada de actividades de PF (para semanas 13 – 16)

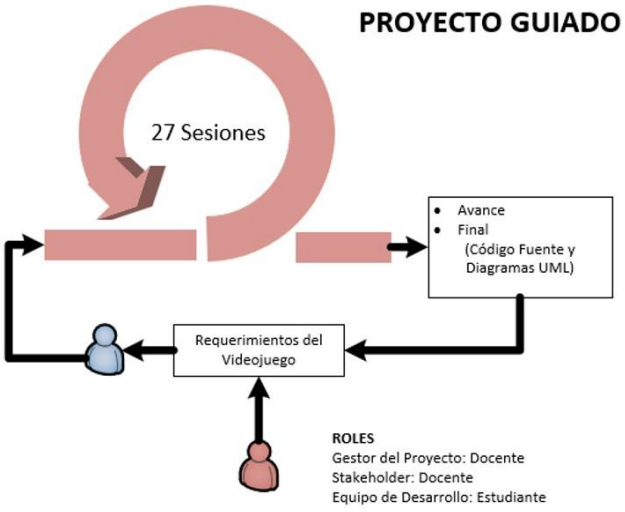


Fig. 9. Esquema con las actividades específicas y roles del PG

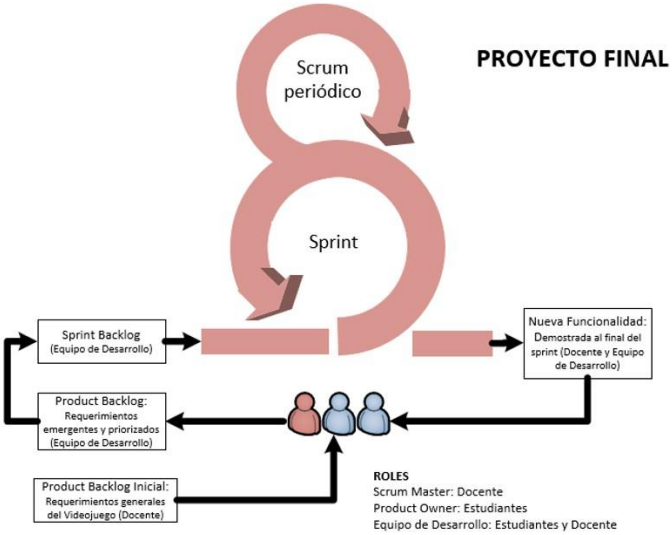


Fig. 10. Esquema con las actividades específicas y roles del PF

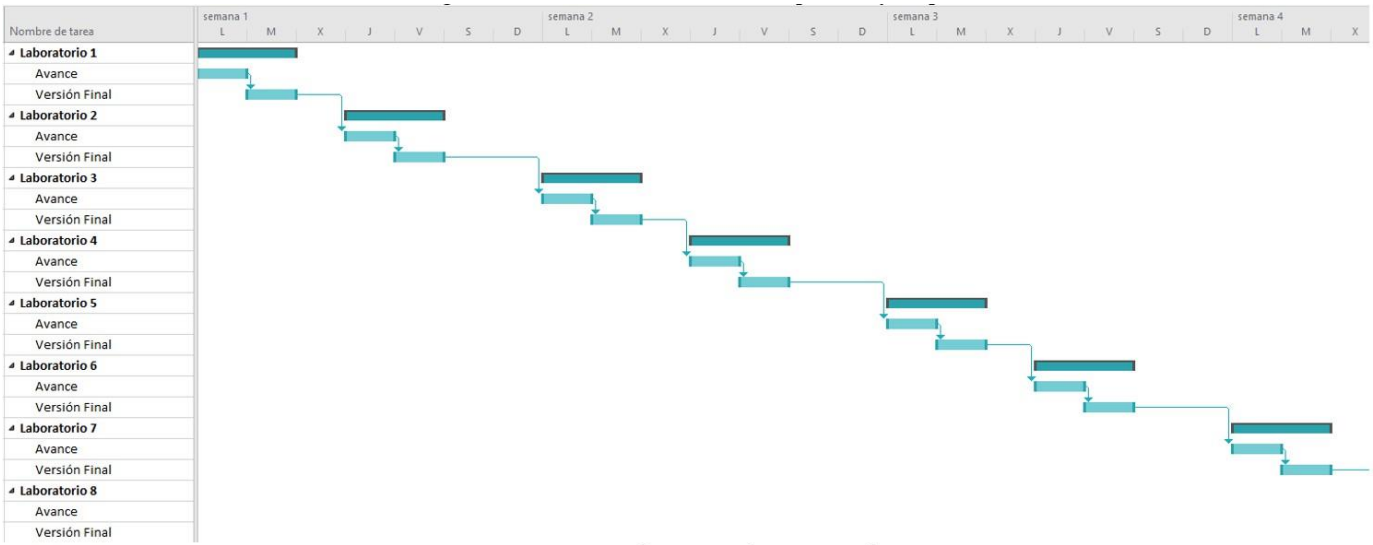


Fig. 11. Planificación de actividades para PG y PF para las primeras 12 semanas

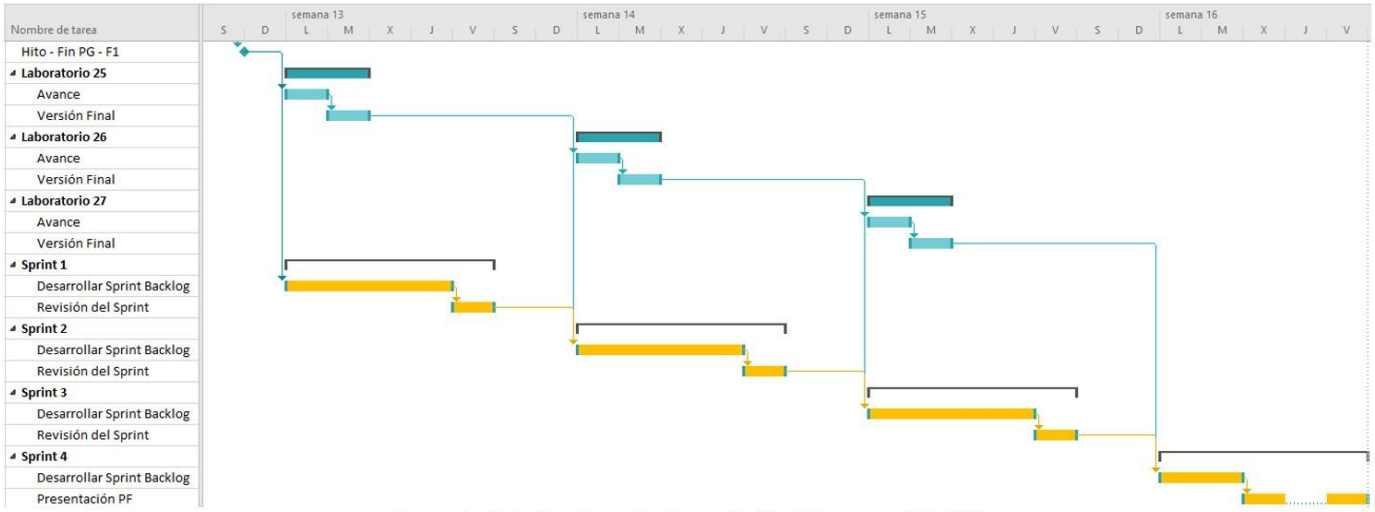


Fig. 12. Planificación de actividades para PG y PF para las semanas 13 - 16

H. Validación del Modelo

La propuesta de modelo de enseñanza se validó en el curso de Fundamentos de Programación 2 (FP2) de la Escuela Profesional de Ingeniería de Sistemas de la Universidad Nacional de San Agustín de Arequipa en los periodos 2019, 2020 y 2021.

Fundamentos de Programación 2 es un curso básico y fundamental de los planes de estudio 2013 y 2017 de la Escuela Profesional de Ingeniería de Sistemas. Dichos planes se elaboraron tomando en cuenta los resultados del estudiante indicados por Accreditation Board for Engineering and Technology (ABET), donde se destaca la importancia de las habilidades profesionales y las habilidades de conciencia, además del desarrollo de las habilidades técnicas para lograr excelencia en la formación de ingenieros.

El curso se orienta a la enseñanza de los conceptos de la programación orientada a objetos y otros tópicos avanzados, constituyéndose en el segundo curso de programación en dichos planes curriculares, pertenece al segundo semestre de estudios y utiliza como lenguaje de programación a Java. Tiene una duración de 17 semanas y cuenta con 8 horas semanales (2 horas teóricas, 2 horas prácticas y 4 horas de laboratorio), siendo en esas horas de laboratorio donde se realiza la experiencia.

El curso cumple el marco de trabajo general descrito en la sección III y en base a ello se aplicaron las actividades del PG y del PF propuestas en la sección IV.

Sobre la definición del videojuego del PG, se planteó la siguiente: la trama propuesta consiste en desarrollar un videojuego de estrategia por turnos que está enmarcado en la Edad Media donde las naciones que existen actualmente empezaban a forjarse, tales como Inglaterra, Francia, España, Alemania, etc., las cuales nacían a partir de reinos e imperios medievales que en el videojuego los llamamos "culturas". El videojuego a desarrollar tiene que ver tanto con el aspecto estratégico donde se distribuyen los ejércitos de cada cultura sobre un mapa general (tablero), como con la parte de las batallas a luchar entre los ejércitos de las culturas, lo que se llama el aspecto táctico de un juego de estrategia. Las batallas se libran entre 2 ejércitos de culturas rivales, donde cada ejército está compuesto por soldados. Dichos soldados tienen ciertas características y comportamientos que se deben implementar. La decisión del ganador de la batalla no debe ser arbitraria, se debe basar en alguna métrica que los estudiantes deben proponer, así la métrica más simple sería "cantidad de soldados del ejército", la cual sería justa si todos los soldados fueran iguales, pero no sería la mejor métrica ya que al avanzar en el proyecto aparecen diferentes tipos de soldados, unos mejores que otros y las cantidades de ellos no necesariamente van a decidir al ganador, tal y como sucedía en tiempos medievales. El juego continúa hasta que una cultura haya derrotado completamente a la otra al vencer a todos los ejércitos rivales. Ver Fig. 13.

En la Fig. 14 se pueden observar algunos de los entregables del PG y del PF.

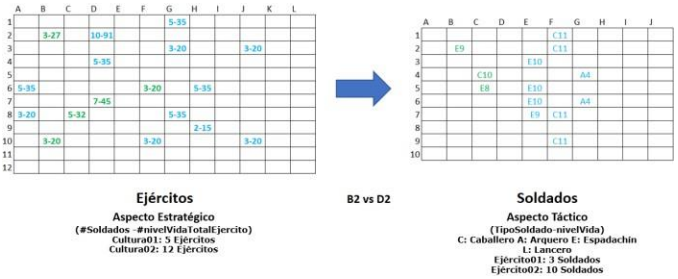
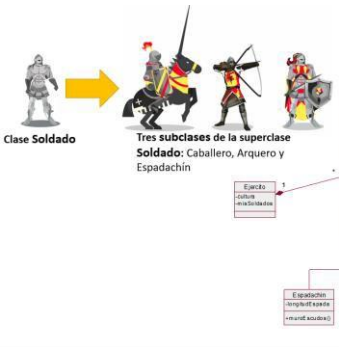


Fig. 13. Aspectos estratégico y táctico del videojuego propuesto para PG




```
import java.util.*;

public class Ejercito {
    private ArrayList<Soldado> misSoldados=new ArrayList<Soldado>();
    private String cultura;

    public Ejercito(String cult, int cantidad){
        cultura=cult;
        for(int i=0;i<cantidad;i++)
            misSoldados.add(new Soldado("S"+i,10,5,10));
    }
    public String toString(){
        String todos="";
        for(int i=0;i<misSoldados.size();i++)
            todos+=misSoldados.get(i)+"\n";
        return cultura+"\n"+todos;
    }
}
public class Caballero extends Soldado{
    private String armaActual="Lanza";
    private boolean montando=true;

    public Caballero(String s, int ata, int def, int vid){
        super(s,ata,def,vid);
    }

    public void envestir(){
        if (montando==true)
            for(int i=0;i<=2;i++)
                super.atacar();
        else
            super.atacar();
    }
    public void desmontar(){
        if(montando=true){
            montando=false;
            super.defender();
            cambiaArma();
        }
    }
    public void cambiaArma(){
        if(armaActual=="Lanza")
            armaActual="Espada";
        else
            armaActual="Lanza";
    }
    public void montar(){
        if(montando=false){
            montando=true;
            super.atacar();
            cambiaArma();
        }
    }
}
```

Fig. 14. Ejemplos de entregables de las actividades del PG y PF

V. Evaluación y Análisis de Resultados

La experiencia de aplicar el modelo de enseñanza propuesto fue muy motivadora para los estudiantes y el docente, debido a que se realizó el seguimiento de la forma en que se construía el conocimiento de los estudiantes de forma iterativa e incremental, sesión a sesión, destacando la expectativa y la motivación de los estudiantes.

Para el PF la mayoría de los estudiantes avanzaron el videojuego más allá de los requerimientos propuestos, aplicando incluso conceptos que no fueron enseñados durante el curso, realizando investigación, mejorando la jugabilidad del videojuego, siendo muy creativos e innovadores.

A lo largo de los años hubo un gran problema con el rendimiento de los estudiantes, de tal forma que las notas finales y el porcentaje de aprobación siempre fueron relativamente bajos durante los primeros años en que se dictó el curso. Para el 2018 se empezaron a aplicar actividades lúdicas cortas que favorecieron el rendimiento académico, pero después de la presente experiencia realizada en los años 2019, 2020 y 2021, todas las métricas tales como la cantidad de aprobados, promedio general y desviación estándar mejoraron (calificación vigesimal). Tabla IX y Fig. 15.

TABLA IX
Resultados del Rendimiento Final del Curso FP2 por años

	2013	2014	2015	2016	2017	2018	2019	2020	2021
NºEstudia.	123	112	115	100	115	92	102	82	99
Aprobados	49,15%	40,82%	45,45%	46,00%	52,18%	57,61%	72,55%	75,88%	80,31%
Desaprobad	50,85%	59,18%	54,55%	54,00%	47,82%	42,39%	27.45%	24,12%	19,69%
Promedio	10,10	9,12	9,21	9,96	10,78	11,58	12,10	12,88	12,65
Desv. Está.	3,86	4,36	4,94	3,59	2,65	2,55	2,12	2,29	2,25
Máxima	17	16	17	17	17	16	16	17	17
Mínima	1	1	1	1	2	5	5	8	4

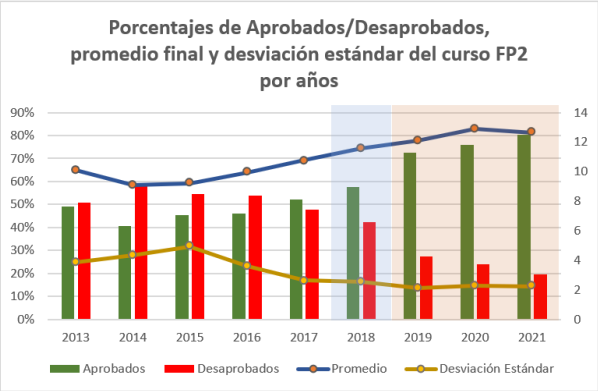


Fig. 15. Evolución anual del nivel de rendimiento del curso FP2

Conclusiones

En este artículo se presentó un modelo de enseñanza utilizado para estimular la motivación en los estudiantes y enseñarles, mediante el desarrollo de videojuegos, los conceptos de la programación orientada a objetos: clase, objeto, atributo, método, mensaje, composición/agregación, clase abstracta, miembros de clase, herencia y polimorfismo, interfaces gráficas de usuario, estructuras de datos como arreglos estándar, ArrayList, HashMap y el uso de archivos.

Se demostró que enseñar los conceptos de programación orientada a objetos a través de proyectos de desarrollo de software de videojuegos aplicando gestión de proyectos de software y Scrum, genera una motivación intrínseca en los estudiantes y permite lograr resultados destacados en una generación de nativos digitales puros que tienen su propio estilo de aprendizaje.

El proyecto de desarrollo de videojuegos constituye una referencia para la parte teórica y de laboratorio del curso, constituyéndose en una gran ayuda en la enseñanza de conceptos que podrían resultar bastante abstractos si se enseñaran de otra forma.

Los estudiantes experimentan una experiencia temprana de participar en proyectos de desarrollo de software, situación que es fundamental en la práctica profesional de cualquier ingeniero de sistemas/software/informático.

Este modelo de enseñanza puede ser aplicado en escuelas profesionales de Ingeniería de Sistemas/Software/Informática nacionales o extranjeras, incluso en escuelas profesionales de otros tipos de ingeniería y de ciencia que consideren a la programación de computadoras orientada a objetos como un eje importante en la formación profesional de sus estudiantes.

Referencias

[1] E. Sweedyk, M. deLaet, M. C. Slattery, and J. Kuffner, "Computer games and CS education: why and how," in Proceedings of the 36th SIGCSE technical symposium on Computer science education, St. Louis, Missouri, USA, 2005, pp. 256-257.

[2] S. Leutenegger & J. Edgington, "A games first approach to teaching introductory programming", SIGCSE 2007: 38th SIGCSE Technical Symposium on Computer Science Education, pp. 115-118, 2007.

[3] M. Aedo, E. Vidal, E. Castro & A. Paz, "Aproximación orientada a Entornos Lúdicos para la primera sesión de CS1 - Una experiencia con nativos digitales", 15th Latin American and Caribbean Conference for Engineering and Technology (LACCEI) - International Multi-Conference for Engineering, Education, and Technology, United States, 2017.

[4] D. Topalli & N. E. Cagiltay, "Improving programming skills in engineering education through problem-based game projects with scratch", Computers and Education, 120, pp. 64-74, 2018.

[5] M. Aedo, E. Vidal & E. Castro, "Experience in the teaching of Fundamentals of Object-Oriented Programming through the implementation of a Strategy Videogame", 17th Latin American and Caribbean Conference for Engineering and Technology (LACCEI) - International Multi-Conference for Engineering, Education, and Technology, Jamaica, 2019.

[6] M. Prensky, "Digital Natives, Digital Immigrants," Horiz., vol. 9, no. 5, pp. 1–6, 2001.

[7] J. García Cué, "Estilos de Aprendizaje y Tecnologías de la Información y la Comunicación en la Formación de Profesorado". España, 2006.

[8] M. Aedo, "Propuesta de un Modelo Metodológico para la Enseñanza de la Programación Orientada a Objetos basada en el Desarrollo de un Videojuego y Aplicando Gestión de Proyectos de Software y Scrum", Tesis M.S. Ingeniería de Software, UNSA, Arequipa, Perú, 2021.

[9] P. A. Sanger & J. Ziyatdinova, "Project based learning: Real world experiential projects creating the 21st century engineer", Proceedings of 2014 International Conference on Interactive Collaborative Learning, Emiratos Árabes Unidos, 2014.

[10] J. C. López-Pimentel, A. Medina-Santiago, M. Alcaraz-Rivera & C. Del-Valle-Soto, "Sustainable project-based learning methodology adaptable to technological advances for web programming", Sustainability, Suiza, 2021.