REVISTA
SAN GREGORIO

http://dx.doi.org/10.36097/rsan.v0i51.2122
Artículo Original

Patrones de diseño en el modelo computacional del proceso de ingreso a la educación superior cubana

Design patterns in the computational model of the process of admission to Cuban higher education

Autores

Irina García Ojalvo. *http://orcid.org/0000-0001-8000-8683*
Centro de Estudios para el Perfeccionamiento de la Educación Superior. Universidad de La Habana.
irina@cepes.uh.cu

Roberto Sepúlveda Lima. *http://orcid.org/0000-0002-9451-6395*
Ministerio de Educación Superior, La Habana, Cuba.
sepulveda@mes.gob.cu

Isidro Alfredo Abelló Ugalde. *http://orcid.org/0000-0001-6015-4261*
Centro de Estudios para el Perfeccionamiento de la Educación Superior. Universidad de La Habana.
isidro@cepes.uh.cu

Fecha de recibido: 2022-03-02
Fecha de aceptado para publicación: 2022-08-02
Fecha de publicación: 2022-09-30

Resumen

El continuo perfeccionamiento de las políticas de acceso a la educación superior cubana hace necesario contar con un sistema informático flexible, tanto para su mantenimiento como para su evolución, que permita responder más eficientemente a los cambios que se suceden cada curso escolar. El presente trabajo tiene el objetivo de analizar el uso de patrones de diseño en la concepción de un modelo computacional del proceso de ingreso. Se presenta la dimensión operacional del modelo elaborado teniendo en cuenta varios de los patrones referidos en la literatura. Se realizó una consulta a expertos en el desarrollo de software para valorar la conveniencia de utilizar estos patrones. Como resultado se obtuvo que más del 70% apreciaron este uso como muy adecuado o bastante adecuado. En la práctica el uso de patrones de diseño ha

permitido desarrollar un sistema informático capaz de evolucionar según los cambios que ha tenido el proceso de ingreso a través de los años.

Palabras clave: Educación superior; ingreso; patrones de diseño; software; software evolutivo.

Abstract

The continuous improvement of policies for access to higher education in Cuba makes it necessary to have a flexible computer system, both for its maintenance and for its evolution. This allows it to respond more efficiently to the changes that occur each school year. The present work has the objective of analyzing the use of design patterns in the conception of a computational model of the admission process. It is presented the operational dimension of the model developed taking into account several of the patterns referred to in the literature. Experts in software development were consulted to assess the convenience of using these patterns. As a result, it was obtained that more than 70% appreciated this use as very appropriate or quite appropriate. In practice, the use of design patterns has allowed development a computer system that has evolved according to the changes that the admission process has undergone over the years.

Keywords: Admission; design patterns; higher education; software; software evolution.

Introducción

En el mundo actual, globalizado e interconectado, están presentes un conjunto de retos socioeconómicos que impactan sensiblemente a todos los países, con grandes consecuencias, sobre todo para aquellos de menor desarrollo. Ante este escenario, la educación superior se convierte en uno de los ejes estratégicos y condición necesaria para el logro de los avances que en el desarrollo económico y social se proyectan para los próximos años.

Al respecto, en la Declaración de la III Conferencia Regional de Educación Superior CRES 2018, celebrada en Córdoba, Argentina, se abordó el papel de la educación superior para el logro de los objetivos del desarrollo sostenible. Aquí se planteó que "…la educación superior debe constituirse desde los liderazgos locales, estatales, nacionales e internacionales…" y remarca la necesidad de "… llevar a cabo una nueva e histórica transformación de la educación superior desde el compromiso y la responsabilidad social, para garantizar el pleno ejercicio al derecho a la educación superior pública gratuita y de amplio acceso…" (IESALC, 2018, pág. 3).

Esto ha conllevado a los sistemas de educación superior a implementar estrategias de ampliación de la oferta universitaria, a la revisión en profundidad de los procedimientos de acceso al sistema y a la generación de políticas inclusivas, para lograr el acceso universal, la permanencia y la titulación.

La demanda creciente por acceder a la educación superior exige que se atienda a múltiples necesidades de jóvenes y adultos, en distintos tipos de instituciones, programas, modalidades de enseñanza aprendizaje y estrategias formativas. Esto hace que el proceso de ingreso se complejice cada vez más, con el fin de alcanzar altos niveles de calidad, a través de la formulación de adecuadas políticas, estrategias y mecanismos de apoyo que propicien su gestión eficiente y eficaz.

La educación superior cubana no es una excepción en la intención y la necesidad de hacer efectivo el acceso universal, la permanencia y el egreso atendiendo a una formación de calidad con inclusión y pertinencia local y regional. En particular, desde hace más de cuatro décadas, ha encaminado esfuerzos significativos por perfeccionar su sistema de ingreso debido a su contribución en la satisfacción de las demandas de formación de los graduados universitarios que necesitan el país y cada provincia, así como el interés de la población por acceder a los estudios superiores. Ello ha estado particularmente vinculado con las acciones favorecedoras de la permanencia y egreso de los estudiantes y en general, con la elevación de la eficiencia académica (Haramboure & García, 2018).

La complejidad del proceso de ingreso a las Instituciones de Educación Superior (IES), así como la participación de múltiples factores y sus interconexiones, ha requerido el uso de sistemas informáticos que hagan factible esta actividad. A través de los años, el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) ha acompañado la gestión de los procesos de ingreso a la educación superior, brindando servicios cada vez de mayor calidad y alcance.

Los softwares o sistemas informáticos permiten hacer la recolección dinámica de datos, garantizan el tratamiento adecuado de los mismos y se unifican criterios sobre los posibles enfoques que los analistas puedan hacer sobre ellos y su correcta utilización en la toma de decisiones. De esta forma, se evita recargar el trabajo de los directivos, profesores y otro personal de la institución; se reduce el tiempo del procesamiento; se contribuye a la divulgación de la información y se elimina la posible duplicidad de esta, haciendo que los datos sean más confiables.

Para realizar el diseño e implementación de sistemas informáticos que respondan adecuadamente a los requisitos de la actividad en la que intervienen, se parte de la elaboración de

un modelo computacional (Wagner, 2018). Este modelo brinda una representación de la estructura y comportamiento de la actividad en cuestión. Los modelos, además de servir de base para el desarrollo de los programas, los hacen más comprensibles y fáciles de comunicar, por lo que constituyen un puente entre desarrolladores y usuarios de los sistemas informáticos.

Una de las herramientas más utilizadas para elaborar modelos y desarrollar soluciones computacionales son los patrones de diseño. Estos cumplen la función de capturar, recopilar y transmitir la experiencia del programador.

El objetivo de este trabajo es analizar el uso de algunos patrones de diseño en la concepción de la dimensión operacional del modelo computacional del proceso de ingreso a la educación superior cubana.

Metodología

1. Dimensión operacional del modelo computacional del proceso de ingreso a la educación superior cubana

El modelo computacional presentado en García, (2021), consta de las dimensiones teórica y operacional. La primera analiza el proceso de ingreso a la educación superior cubana desde el punto de vista de sus relaciones esenciales, principios y premisas que, entre otras características, determinan las propiedades de la dimensión operacional. En esta última, los núcleos de dominio, diseño e implementación van conformando las estructuras de datos y las funcionalidades en niveles crecientes de abstracción, hasta la construcción de la propuesta, que tuvo lugar a través de un proceso iterativo e incremental propio de las metodologías de desarrollo de software.

El núcleo de dominio tiene el objetivo de describir las entidades del proceso de ingreso, sus propiedades, funcionalidades y relaciones entre ellas. Las entidades identificadas fueron: (a) aspirantes, (b) plazas universitarias, (c) criterios de selección y (d) asignación de plazas. Estas se relacionan entre sí de manera particular según el momento del proceso de ingreso: planificación y organización, realización de los exámenes y asignación de las plazas.

Uno de los enfoques más utilizados para elaborar modelos y desarrollar soluciones computacionales es la programación orientada a objetos (POO) (Smith, 2015). La POO se basa en el modelo de objeto, que es una unidad que contiene todas sus características y comportamientos en sí misma, esto lo hace un todo independiente, pero que se interrelaciona con otros objetos de su misma clase o de otras clases, como sucede con los objetos en el mundo real. Por lo tanto, este

paradigma permite concebir, analizar, modelar, diseñar e implementar el mundo de manera similar a como se presenta en la realidad; el paso que hay desde la concepción y asimilación del problema, hasta la solución computacional del mismo, es un proceso que se hace de manera casi natural.

Es en este contexto que se define el núcleo de diseño del modelo, cuyo objetivo es determinar las jerarquías de clases según el paradigma POO, para representar las propiedades y mecanismos que determinan el comportamiento del proceso de ingreso a la educación superior.

Dentro de la POO, los patrones de diseño son un conjunto de métodos para la definición de jerarquías de clases que permiten dar solución a los problemas más comunes que se presentan en el desarrollo de sistemas informáticos eficientes, flexibles y evolutivos (Schmidt et al, 2013, Birare, 2019).

En 1995, un grupo de desarrolladores de software (Gamma et al, 1995) publicaron un texto fundamental con la descripción de patrones de diseño comunes, aplicables a la resolución de problemas informáticos. Cada patrón prescribe una estructura de clases, sus roles y colaboraciones y una adecuada asignación de métodos para resolver problemas de diseño, de una manera flexible y adaptable.

Con el fin de dar mayor flexibilidad y evolutividad a los sistemas informáticos derivados del modelo propuesto, se utilizaron los siguientes patrones de diseño (García et al, 2018).

- **Patrón Singleton:** Según Nesteruk (2019) y Joshi (2016), se utiliza en la creación y manejo de las variables globales del sistema. Garantiza que haya exactamente un único objeto de determinada clase, en este caso es la clase que contiene toda la información que necesitan los distintos módulos y procesos de la aplicación (Aljasser, 2016). Aquí se asegura que la información esté disponible, a través de un punto de acceso conocido. El patrón Singleton hace que la clase sea responsable de su único objeto, quitando así este problema a los otros procesos que lo usan. Adicionalmente, se pueden construir subclases y otros objetos deben ser capaces de usar las subclases sin modificar su propio código. Ambos elementos permiten dar mayor flexibilidad al sistema ante la necesidad de realizar modificaciones.
- **Patrón Composite:** Se utiliza en la definición del árbol de opciones que se presenta al usuario del sistema informático. Para Philip el al. (2020), este patrón estructural permite representar jerarquías de objetos del tipo todo-parte, brindando a las clases que usan la

estructura, la facilidad de ignorar la diferencia entre objetos primitivos y composiciones de estos y así tratarlos uniformemente. Esta estructura favorece la flexibilidad al facilitar la incorporación de nuevos tipos de componentes, en este caso, nuevas funcionalidades del sistema sin modificar el código existente.

- **Patrón Chain of Responsibility:** Este patrón permite desacoplar el objeto emisor de una petición de su receptor, dando a más de un objeto la posibilidad de responder a ella (Vinoski, 2002). Para esto, se encadenan los receptores y la petición pasa a través de la cadena hasta que es procesada por algún elemento. En el modelo computacional se aplica este patrón en el comportamiento del árbol de opciones del usuario, donde hay más de un objeto que manejan ciertas peticiones y no se conoce hasta el momento de la ejecución cuál de ellos debe intervenir. De esta manera, se simplifican las interconexiones entre objetos y se reduce el acoplamiento entre ellos. Esto hace que se agregue flexibilidad al sistema ya que se pueden añadir o cambiar responsabilidades modificando la cadena en tiempo de ejecución.
- **Patrón Template Method:** Es un patrón de comportamiento que define el esqueleto de programa de un algoritmo en un método, el cual difiere algunos pasos a las subclases. Permite redefinir ciertos pasos de un algoritmo sin cambiar su estructura, para lograr un comportamiento variable y evitar la duplicación de código; manteniendo el control de los puntos en que se puede permitir dicha variación (Bijlsma et al, 2018). En el diseño propuesto se utiliza este patrón en la definición genérica de los diálogos de búsqueda y selección de entidades para diferentes eventos y en el propio algoritmo de asignación de plazas.
- **Patrón Visitor:** Este patrón permite definir nuevas operaciones sin cambiar las clases de los elementos sobre los que opera (Springer et al, 2016). La operación se define en cada una de las clases que representan los posibles tipos sobre los que se aplica y en tiempo de ejecución se elige qué versión de la misma se debe ejecutar. El uso de este patrón permite extender el software para dar respuesta a nuevos requerimientos.
- **Patrón Strategy:** mediante el uso de este patrón, se puede definir una familia de algoritmos, encapsulándolos de manera que se pueda intercambiar uno por otro (Bala & Kaswan, 2014, Rivera et al, 2016). En el modelo propuesto se usa este patrón para el

diseño de los algoritmos de asignación. El objeto correspondiente al patrón y el objeto que lo solicita colaboran para implementar el algoritmo concreto en cada situación. Este mecanismo permite gestionar la familia de algoritmos de asignación de manera que se seleccione en tiempo de ejecución, el algoritmo adecuado, y que se puedan crear nuevos algoritmos en el sistema sin cambiar el código existente.

- **Patrón Observer,** el cual define una dependencia de uno a muchos entre objetos, de forma que, cuando un objeto cambia de estado, se notifica a los objetos dependientes para que se actualicen automáticamente. Con esto se consigue mantener la consistencia entre objetos relacionados, sin aumentar el acoplamiento entre clases, es decir, el objeto es capaz de notificar algo a otros sin tener que saber quiénes son. Esto permite que los objetos sean cambiados y reutilizados cuando la evolución del software lo requiera.

El núcleo de implementación tiene el objetivo de adaptar el modelo a un entorno de programación específico. Para ello fue necesario definir la metodología de desarrollo de software, en este caso la XP (eXtreme Programming), mediante la cual se potenció el uso de prácticas como las historias de usuarios, la planificación e integración continua, entregas pequeñas y frecuentes, la refactorización y el desarrollo dirigido a pruebas.

Otras tareas dentro del núcleo de implementación fueron las definiciones de los esquemas de datos, arquitectónico, de interfaz y de procedimientos. En ellos se realizó la transformación del modelo entidad-relación a tablas normalizadas típicas de los sistemas de bases de datos relacionales; se determinaron los componentes que llevan a cabo las tareas computacionales, sus interfaces y la comunicación entre ellos; se estableció la interfaz visual y estética del sistema en correspondencia con las características psicológicas, físicas y sociales de los usuarios y el modo, intensidad y secuencia de uso del software. Finalmente se definió la funcionalidad lógica del sistema informático en módulos esenciales:

1. Módulos correspondientes a los momentos del proceso de ingreso y otro de control de la aplicación
2. Componentes de usuarios y roles
3. Sistema distribuido: operación autónoma en cada territorio y sincronización con la base de datos nacional
4. Mecanismo de salva y recuperación de la base de datos
5. Control de etapas del proceso y de operaciones sensibles

Como concreción del modelo computacional del proceso de ingreso a la educación superior cubana propuesto, el Grupo de Automatización del Ingreso del Centro de Estudios para el Perfeccionamiento de la Educación Superior de la Universidad de La Habana (CEPES-UH), desarrolló el Sistema Automatizado Distribuido de Ingreso a la Educación Superior (SADIES). Este sistema informático ha sido utilizado para el apoyo a los procesos de ingreso a las universidades en todas las provincias del país.

2. Evaluación de la dimensión operacional del modelo computacional del proceso de ingreso a la educación superior cubana

Con el fin de evaluar la dimensión operacional, la estructura de sus núcleos y, en particular, la conveniencia del uso de los patrones de diseño, se realizó una consulta a expertos. El grupo de expertos se conformó fijando como criterio fundamental de selección la competencia de los candidatos en el área del desarrollo de software sobre la base de su currículo personal. Se identificaron siete (7) posibles candidatos, a los cuales se les aplicó un cuestionario de autoevaluación, como parte de la metodología para determinar su coeficiente de competencia k (Cruz & Martínez, 2019).

El cálculo reveló que todos poseen un coeficiente de competencia k superior a 0.85, por lo cual todos resultaron seleccionados como expertos para intervenir en la validación. Es de destacar que todos los expertos son graduados de Ingeniería Informática o Ciencias de la Computación, tres fueron doctores en ciencias y profesores con categoría docente de Titular o Auxiliar. La experiencia en el desarrollo de software estuvo entre 8 y 27 años, con 15 años como promedio.

La encuesta aplicada a los expertos tuvo el objetivo de recoger su valoración acerca del diseño y contenido de la dimensión operacional del modelo computacional propuesto. Contenía los aspectos relacionados con los núcleos de dominio, de diseño y de implementación del modelo y en específico del uso de los patrones de diseño presentados.

En el núcleo de dominio, se indagó acerca de la identificación de las entidades relevantes del proceso de ingreso y la caracterización de las relaciones entre las entidades. En el núcleo de diseño, acerca de la determinación del conjunto de clases abstractas y jerarquías de clase, según el paradigma POO, haciendo énfasis en el análisis de los patrones de diseño. En el núcleo de implementación se preguntó sobre el empleo de la metodología XP para el desarrollo del software, y los esquemas de datos, arquitectónico, de interfaz y de procedimientos.

Se empleó una escala de Likert con las categorías de “Muy Adecuado”, “Bastante Adecuado”, “Adecuado”, “Poco Adecuado” e “Inadecuado”. En el cuestionario se dio la posibilidad a los expertos de emitir sugerencias y observaciones sobre la propuesta presentada.

Para el procesamiento de los datos recogidos se utilizó la estadística descriptiva y el análisis de los resultados se apoyó en los estadígrafos de frecuencia, porcentaje, media y mediana, como medidas de tendencia central y desviación estándar, como medida de dispersión.

Resultados y discusión

Como resultado de la consulta a expertos se conoció que el núcleo de dominio fue evaluado como “Muy adecuado” o “Bastante adecuado” por el 71.4% de los encuestados, al igual que el núcleo de diseño, entre estas dos categorías. El núcleo de implementación se valoró, en general, como “Muy adecuado”, excepto el ítem “Empleo de la metodología XP para el desarrollo del software” que fue valorada entre “Adecuada” y “Muy adecuada” por el 71.4% de los expertos. Las medidas de tendencia central ubicaron a los ítems evaluados alrededor de las categorías de “Muy adecuado” y “Bastante adecuado”, aunque con desviaciones estándar mayores que uno.

Es de destacar la sugerencia emitida por los expertos para el perfeccionamiento del modelo computacional de valorar la incorporación de un esquema de seguridad, por la sensibilidad que tiene este tipo de sistemas para apoyar la gestión el ingreso a la educación superior. En la propuesta, los aspectos relacionados con la seguridad del modelo están contenidos dentro del núcleo de implementación, en el esquema de procedimientos, donde se expresan los riesgos vinculados con el proceso de ingreso, los riesgos de las tecnologías empleadas y la forma de mitigarlos.

En particular, el uso de los patrones de diseño presentados fue evaluado como “Muy adecuado” o “Bastante adecuado” por el 71.4% de los encuestados, de esta manera se reafirma la conveniencia de utilizar este tipo de recurso para dar mayor flexibilidad y evolutividad al software desarrollado.

Los patrones de diseño para el desarrollo de software constituyen uno de los avances más utilizados en la tecnología orientada a objetos (Schmidt et al, 2013). El concepto de patrones como técnica para el desarrollo, resume la experiencia y el conocimiento adquirido con el correr de los años, de forma que pueda ser transmitido, y se logre con ellos un lenguaje común dentro de la disciplina. De manera que, lo que antes los desarrolladores aplicaban en forma intuitiva para

obtener buenas soluciones, ahora es modelado en patrones que pueden ser utilizados por otros que se encuentran con problemas similares.

Por su parte Shalloway y Trott (2004), afirman que el uso de patrones hace que los diseños exitosos sean más fáciles de reutilizar. Expresar técnicas probadas como patrones las hace más accesibles a los desarrolladores de nuevos sistemas. Para Jaafar el al. (2016) y Hussain el al. (2019), los patrones ayudan a seleccionar alternativas de diseño que potencian las características de calidad del software.

Una de las dificultades del diseño tradicional es que no se ocupa en profundidad de la gestión de dependencias entre los objetos, sino de su conceptualización. En los sistemas informáticos unos objetos dependen de otros, de manera que cuando el software requiere cambios, las modificaciones en el código de uno de ellos pueden generar indeseables efectos colaterales en cascada. Estas dependencias son las que hacen el código más frágil o más robusto frente a los cambios necesarios por la propia evolución del sistema informático y la mayoría de los patrones de diseño están orientados a manejar la gestión de dependencias.

Uno de los aspectos fundamentales para que un sistema informático se desarrolle exitosamente consiste en anticipar los nuevos requisitos que puedan presentarse y diseñar el sistema de manera tal que pueda evolucionar de acuerdo a ellos (Marmsoler, 2018). Este es el caso de un sistema de apoyo al proceso de ingreso, actividad que está en constante transformación para dar respuesta a las crecientes y cambiantes exigencias que enfrenta la educación superior cubana, en torno a la equidad, flexibilidad e inclusión (García et al, 2020).

Para diseñar un sistema que sea robusto ante tales modificaciones se debe considerar cómo el sistema puede necesitar cambiar en su tiempo de vida. Un diseño que no tenga en cuenta los cambios corre el riesgo de necesitar mayores rediseños en el futuro, lo que hace más caro el mantenimiento del mismo (Al-Obeidallah et al, 2016).

Los patrones de diseño ayudan a evitar este encarecimiento, asegurando que el sistema pueda cambiar en una forma específica (Gamma et al, 1995). Cada patrón de diseño permite a algún aspecto de la estructura del sistema variar independientemente de los otros, haciéndolo más robusto frente a un tipo de cambio en particular (Hamid & Perez, 2016).

Como concreción del modelo computacional del proceso de ingreso a la educación superior cubana propuesto, se desarrolló el Sistema Automatizado Distribuido de Ingreso a la Educación Superior (SADIES). Este es un sistema informático utilizado para el apoyo a los procesos de

ingreso a las universidades en todas las provincias del país. Es un software diseñado y desarrollado por el Grupo de Automatización del Ingreso del Centro de Estudios para el Perfeccionamiento de la Educación Superior de la Universidad de La Habana (CEPES-UH).

SADIES permite organizar el proceso y efectuar la asignación de carreras. Para esto realiza la captación de los datos referidos tanto a los planes de plazas universitarias, como a las solicitudes de los aspirantes y proporciona las actas de comparecencia a los exámenes de ingreso y un sistema de anonimato que contribuye a la total transparencia en la calificación emitida por los tribunales. Además de distribuir las plazas universitarias, adaptándose a las resoluciones ministeriales de cada año, facilita los procedimientos de reclamación de calificaciones, re-oferta de plazas disponibles y emite reportes estadísticos sobre la marcha del proceso. También emite los listados de matrícula de todas las IES del país, listados de becarios, entre otros.

Un rasgo que distingue a SADIES de sistemas informáticos anteriores utilizados con la misma finalidad es el hecho de ser distribuido. Es decir, existe una comunicación de datos entre los servidores de las provincias y con un servidor central, de manera que la información está centralizada, lo que posibilita el acceso y control a nivel nacional. A su vez, la distribución de los datos garantiza que, si en determinado momento se pierde la comunicación con el servidor central, se pueda continuar procesando la información provincial. Además, este mecanismo ofrece mayor seguridad en cuanto a las necesidades de recuperación de datos que pueda existir.

El desarrollo de SADIES se fundamentó en el modelo concebido por García (2021), en el cual se hace uso de los patrones de diseño analizados en el presente artículo. Esto ha permitido que SADIES se constituya en un sistema informático robusto, evolutivo y flexible que se ha empleado para el ingreso a las tres modalidades de estudios, de manera satisfactoria. Este software permitió, por primera vez, gestionar diferentes vías de ingreso, tipos de cursos, carreras e IES, entre otras entidades relacionadas con el proceso de ingreso como la vinculación territorial, las convocatorias a examen, la cantidad y tipo de estos exámenes, el manejo de múltiples escalafones y las distintas variantes de opciones a solicitar por el aspirante.

Conclusiones

En la concepción de la dimensión operacional del modelo computacional del proceso de ingreso a la educación superior cubana, se utilizó el paradigma orientado a objetos y específicamente se emplearon varios patrones de diseño con el fin de aportar robustez, flexibilidad

y evolutividad al sistema informático. Estos patrones permitieron la incorporación de nuevos tipos de componentes; la modificación del comportamiento de objetos y redefinición de procedimientos en tiempo de ejecución. Esto posibilitó extender el software para dar respuesta a nuevos requerimientos, específicamente en la gestión de entidades globales, en el árbol de opciones del usuario, en los diálogos de búsqueda y selección de entidades, así como en el uso de diferentes algoritmos de asignación.

Como parte de la valoración de expertos sobre la dimensión operacional del modelo, se indagó acerca de la conveniencia del uso de los patrones de diseño. Los criterios favorables obtenidos de esta indagación permitieron validar el recurso propuesto.

Referencias

Aljasser, K. (2016). Implementing design patterns as parametric aspects using ParaAJ: The case of the singleton, observer and decorator design patterns. *Computer Languages, Systems & Structures*, 45, 1-15. https://www.sciencedirect.com/science/article/pii/S1477842415000913

Al-Obeidallah, M., Petridis, M., & Kapetanakis, S. (2016). A Survey on Design Pattern Detection Approaches. *International Journal of Software Engineering (IJSE)*, 7(3), 41-59. https://www.researchgate.net/profile/Mohammed-Al-Obeidallah/publication/318504683_A_Survey_on_Design_Pattern_Detection_Approaches/links/5b5db50b458515c4b2511118/A-Survey-on-Design-Pattern-Detection-Approaches.pdf

Bala, R., & Kaswan, K. (2014). Strategy Design Pattern. *Global Journal of Computer Science and Technology*, 14(6), 34-38. http://computerresearch.org/index.php/computer/article/view/144

Bijlsma, A., Passier, H., Pootjes, H., & Stuurman, S. (2018). Template Method test pattern. *Information Processing Letters*, 139, 8-12. https://www.sciencedirect.com/science/article/pii/S0020019018301339

Birare, K. (2019). Analysis of Design Pattern to Develop Object-Oriented Systems. *International Journal of Applied Evolutionary Computation*, 10(3). https://www.igi-global.com/article/analysis-of-design-pattern-to-develop-object-oriented-systems/229085

Cruz, M., & Martínez, M. (2019). Origen y desarrollo de un índice de competencia experta: el coeficiente k. *Revista Latinoamericana de Metodología de la Investigación Social*, 16, 40-56. http://www.relmis.com.ar/ojs/index.php/relmis/article/view/7

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison Wesley Professional. Series: Addison-Wesley Professional Computing Series.

García, I. (2021). *Modelo computacional de apoyo al proceso de ingreso a la educación superior cubana.* [Tesis doctoral]. Universidad de La Habana.

García, I., Galarza, J., & Sepúlveda, R. (2020). El proceso de ingreso a la educación superior. Los sistemas informáticos como herramientas para su ejecución. *Revista Cubana de Educación Superior*, 39(3). http://www.rces.uh.cu/index.php/RCES/article/view/400

García, I., Sepúlveda, R., & Abelló, I. (2018). Sistema automatizado distribuido de ingreso a la educación superior SADIES. Uso de patrones de diseño. *Revista Congreso Universidad*, VII(6).

Haramboure, R., & García, I. (2018). Impacto en la Universidad de La Habana de las modificaciones al ingreso en el curso 2016-2017. *XV Taller Internacional "La Educación Superior y sus Perspectivas". 11mo. Congreso Internacional de Educación Superior "Universidad 2018".* La Habana.

Hussain, S., Keung, J., Sohail, M., Khan, A., & Ilahi, M. (2019). Automated framework for classification and selection of software design patterns. *Applied Soft Computing*, 75, 1-20. https://www.sciencedirect.com/science/article/pii/S1568494618306173

IESALC. (2018). Declaración de la III Conferencia Regional de Educación Superior para América Latina y el Caribe. Córdoba, Argentina.

Jaafar, F., Guéhéneuc, Y.-G., Hamel, S., Khomh, F., & Zulkernine, M. (2016). Evaluating the impact of design pattern and anti-pattern dependencies on changes and faults. *Empirical Software Engineering*, 21(3), 896-931. https://link.springer.com/article/10.1007/s10664-015-9361-0

Joshi, B. (2016). Creational Patterns: Singleton, Factory Method, and Prototype. En B. Joshi, *Beginning SOLID Principles and Design Patterns for ASP.NET Developers.* Berkeley, CA: Apress. https://link.springer.com/chapter/10.1007/978-1-4842-1848-8_3

Marmsoler, D. (2018). Hierarchical Specification and Verification of Architectural Design Patterns. *Fundamental Approaches to Software Engineering. 21st International Conference, FASE 2018.* Thessaloniki, Greece: Springer Open. https://library.oapen.org/bitstream/handle/20.500.12657/27803/1002202.pdf

Nesteruk, D. (2019). Singleton. En D. Nesteruk, *Design Patterns in .NET: Reusable Approaches in C# and F# for Object-Oriented Software Design.* Berkeley, CA: Apress. https://link.springer.com/content/pdf/10.1007/978-1-4842-4366-4.pdf

Philip, M., Natarajan, K., Ramanathan, A., & Balakrishnan, V. (2020). Composite pattern to handle variation points in software architectural design of evolving application systems. *IET Software*, 14(2), 98-105. https://ieeexplore.ieee.org/iel7/4124007/9063968/09063990.pdf

Rivera, B., Becker, P., & Olsina, L. (2016). Quality Views and Stragegy Patterns for Evaluation and Improving Quality: Usability and User Experience Case Studies. *Journal of Web Engineering*, 15(5-6), 433-464. https://journals.riverpublishers.com/index.php/JWE/article/view/3801

Schmidt, D., Stal, M., Rohnert, H., & Buschmann, F. (2013). *Pattern-Oriented Software Architecture, Patterns for Concurrent and Networked Objects.* John Wiley & Sons.

Shalloway, A., & Trott, J. (2004). *Design Patterns Explained A New Perspective on Object-Oriented Design Second Edition.* Addison Wesley Professional.

Smith, B. (2015). Object-Oriented Programming. En J. Lott, & D. Patterson, *Advanced ActionScript 3: Design Patterns.* Berkeley, CA: Apress.

Springer, M., Masuhara, H., & Hirschfeld, R. (2016). Classes as Layers: Rewriting Design Patterns with COP. Alternative Implementations of Decorator, Observer, and Visitor. *COP'16.* Rome, Italy: ACM. https://dl.acm.org/doi/abs/10.1145/2951965.2951968

Vinoski, S. (2002). Chain of responsibility. *IEEE Internet Computing*, 6(6), 80-83. https://ieeexplore.ieee.org/abstract/document/1067742/

Wagner, G. (2018). Information and Process Modeling for Simulation – Part I: Objects and Events. Journal of Simulation Engineering, 1. https://jsime.org/index.php/jsimeng/article/view/2