Universidad de las Ciencias Informáticas

Facultad 3

Título:

Módulo de asignaturas optativas y optimización del Sistema para la generación del horario docente de la Facultad de Ingeniería Industrial de la CUJAE

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores:

José Daniel Sánchez Hechavarría

Carlos Manuel García García

Tutora:

MSc. Isabel González Flores

La Habana, Junio de 2015

"La función de un buen software es hacer que lo complejo aparente ser simple".

Grady Booch

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Autores:

_____

José Daniel Sánchez Hechavarría

_____

Carlos Manuel García García

Tutora:

_____

MSc. Isabel González Flores

AGRADECIMIENTOS

De José Daniel:

A mi mamá, por haber estado apoyándome todo el tiempo, gracias por tu amor.

A mi papá, gracias por tenerme presente.

A mi hermano Rolando, gracias por siempre estar presente.

A mi novia por su apoyo durante la carrera.

A todos los que de alguna manera u otra contribuyeron con mi formación y a hacer posible este momento.

De Carlos:

A mis padres por su inmenso amor, por su enorme sacrificio para que yo este hoy aquí y me forje como profesional, al resto de mi familia que siempre me ha apoyado y se ha preocupado por mí. A mi compañero de tesis por haber sido mi amigo estos 5 años, a mi tutora por haberme guiado y ayudado en este trabajo y a todas las amistades que han compartido conmigo estos inolvidables 5 años de universidad.

DEDICATORIA

De José Daniel:

A mi familia, mi novia y todos mis amigos

De Carlos:

A toda mi familia

A mis amigos

RESUMEN

El problema de la generación de horarios es uno de los clásicos de las ciencias de la computación referido a la asignación de recursos en un período de tiempo, de tal manera que se satisfaga un conjunto de objetivos deseados. La Facultad de Ingeniería Industrial de la CUJAE cuenta con un sistema que permite la planificación del horario docente, sin embargo no incluye la planificación de los cursos electivos ni optativos, razón por la cual se considera incompleto. Estos cursos se planifican de forma manual y están propensos a la ocurrencia de errores debido al factor humano, siendo un proceso lento, complejo y monótono, además la planificación actual no realiza un adecuado tratamiento de las restricciones débiles. El presente trabajo tiene como objetivo el desarrollo de un módulo que incluya la planificación de las asignaturas electivas y optativas, así como la implementación de un algoritmo que permita optimizar el horario generado. Se realizaron pruebas que permitieron comprobar el adecuado diseño e implementación de la solución, obteniéndose finalmente un horario completo que disminuye el número de violaciones de las restricciones débiles.

Palabras claves: Asignaturas optativas, horario docente, optimización, restricciones

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Actualmente la planificación constituye un proceso fundamental a tener en cuenta en cualquier organización, pues propicia su desarrollo, reduce riesgos y permite maximizar el aprovechamiento de los recursos y el tiempo. Es una actividad de gran importancia que se realiza en la mayoría de las organizaciones y que ocupa un lugar significativo en la vida cotidiana como la planificación del transporte, programación de eventos deportivos y horarios laborales. El ámbito educacional ha sido uno de los beneficiados por su uso, evidenciándose en la planificación del horario docente, que contribuye a lograr una buena organización del proceso educativo.

El problema de la planificación de horarios docentes es uno de los clásicos de las ciencias de la computación, según (Schaerf, 1999) consiste en programar una secuencia de clases entre profesores y estudiantes en un prefijado período de tiempo (típicamente una semana), satisfaciendo un conjunto de restricciones. Se considera un problema común y recurrente en todas las instituciones educativas.

Las restricciones de este tipo de problema se clasifican en restricciones fuertes y débiles (Schaerf, 1996), (Larrosa, y otros, 2003), (Chávez Bosquez, y otros, 2014), las primeras son de estricto cumplimiento y definen la factibilidad o validez del horario, mientras las segundas no son de estricto cumplimiento, sin embargo deben satisfacerse en la mayor medida posible, de ello depende la calidad del horario.

La Facultad de Ingeniería Industrial del Instituto Superior Politécnico José Antonio Echeverría, popularmente conocido como CUJAE, cuenta actualmente con un sistema que permite la generación del horario docente en pocos minutos y que satisface las restricciones fuertes establecidas en la institución, por lo que se considera factible; sin embargo no incluye la planificación de los cursos electivos ni optativos, razón por la cual se considera incompleto. Estos cursos contribuyen a la formación de los estudiantes de acuerdo al perfil de estudio de la carrera, se imparten a partir de segundo año y se planifican una vez que se tiene confeccionado el horario docente.

A pesar de que el programa de estudios no varía significativamente no se reutiliza el horario docente de cursos anteriores, pues la matrícula, cantidad de grupos y profesores cambian de un curso a otro. Además las afectaciones de grupos, profesores y locales también varían.

Actualmente la planificadora y la vicedecana docente son las encargadas de realizar el registro de las solicitudes de asignaturas electivas y optativas de los estudiantes, posteriormente proceden a planificarlas en los turnos libres del horario docente. Se debe tener en cuenta que cada estudiante no tenga cursos electivos u optativos simultáneos y que no se violen las restricciones fuertes definidas por la institución para la generación del horario docente. Esta actividad se realiza de forma manual y está propensa a la ocurrencia de errores debido al factor humano, suele demorar varios días a pesar de los años de experiencia de la planificadora y la vicedecana docente y son numerosas restricciones a cumplir siendo un proceso lento,

complejo y monótono. Una vez que se han planificado las asignaturas curriculares, los cursos electivos y optativos, de acuerdo a las restricciones fuertes establecidas, el horario se considera completo y listo para ser entregado a la comunidad universitaria.

En la solución actual no se realiza un adecuado tratamiento de las restricciones débiles, incumpliéndose en mayor medida las relacionadas con la obtención de un horario compacto, el cambio mínimo de locales entre turnos y la planificación de clases consecutivas a profesores que imparten una misma asignatura a más de un grupo.

Ante la problemática planteada se identifica el siguiente **problema a resolver**: ¿Cómo perfeccionar la generación del horario docente de la Facultad de Ingeniería Industrial de la CUJAE, de manera tal que contribuya a la obtención de un horario completo y a la disminución del número de violaciones de las restricciones débiles?

Objeto de estudio: Desarrollo de software en la planificación de horarios docentes.

Objetivo general: Optimizar la generación del horario docente de la Facultad de Ingeniería Industrial de la CUJAE, de manera que integre un módulo de planificación de las asignaturas electivas y optativas, para obtener un horario completo y disminuir el número de violaciones de las restricciones débiles.

Campo de acción: Algoritmos de planificación del horario docente en la Facultad de Ingeniería Industrial de la CUJAE.

Objetivos específicos:

- Elaborar el marco teórico de la investigación.
- Diseñar e implementar el módulo de planificación de las asignaturas electivas y optativas del Sistema para la generación de horario docente de la CUJAE.
- Implementar un algoritmo de optimización del horario docente.
- Valorar la efectividad de la solución propuesta.

La **idea a defender** en la investigación se define como: El desarrollo del módulo de planificación de las asignaturas electivas y optativas y un algoritmo de optimización permitirá la obtención de un horario completo y la disminución del número de violaciones de las restricciones débiles.

El trabajo se encuentra estructurado en tres capítulos:

Capítulo 1. Fundamentación teórica, en el capítulo se presentan los conceptos, teorías, métodos y aplicaciones informáticas existentes en el ámbito nacional e internacional, relacionados con la planificación de asignaturas electivas y optativas y la optimización del horario docente. Se realiza el análisis de la

metodología de desarrollo de software a utilizar, así como los lenguajes de programación y las herramientas apropiadas para darle solución al problema a resolver.

Capítulo 2. Propuesta de solución, en el capítulo se definen los requisitos funcionales y no funcionales, se describe la solución propuesta mediante los artefactos que define la metodología de desarrollo de software escogida, así como las características del nuevo módulo para la planificación de las asignaturas electivas y optativas. Además se muestra el procedimiento para la optimización del horario docente generado a través de la metaheurística búsqueda tabú.

Capítulo 3. Validación de la solución, en el capítulo se presenta las pruebas realizadas al diseño y la implementación de la propuesta de solución, para comprobar su funcionamiento y factibilidad.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el capítulo se referencian los conceptos y teorías relacionados con el problema de planificación de horarios docentes utilizados en la Facultad de Ingeniería Industrial de la CUJAE, ejemplos de sistemas informáticos aplicados en el contexto internacional y nacional, además se especifican las herramientas y tecnologías a usar para darle solución al problema de la investigación.

1.2 Definición de términos

Se definen a continuación los principales términos aplicados en la Facultad de Ingeniería Industrial de la CUJAE relacionados con la investigación.

- **Actividad:** Turno de clase de una asignatura impartido por un profesor a uno, dos o tres grupos en un local determinado.
- **Asignatura**: Cada uno de los tratados o materias que se enseñan en un instituto docente, o forman un plan académico de estudios (RAE, 2015).
- **Asignatura optativa**: Conjunto de conocimientos organizados de forma estructurada de cualquier materia o tema que reciben los estudiantes de manera optativa u opcional con el objetivo de prepararse profesionalmente, desarrollar habilidades y capacidades.
- **Bloque**: Período de tiempo definido por un intervalo de semanas en el que se impartirán las asignaturas electivas y optativas.
- **Curso**: Tiempo señalado de cada año de la carrera, distribuido en dos semestres.
- **Grupo**: Conjunto de estudiantes que pertenecen al mismo año e intervienen en las mismas actividades.
- **Horario completo:** Horario que incluye las asignaturas lectivas, optativas y electivas definidas por una institución educacional, que cumple estrictamente todas las restricciones fuertes.
- **Local**: Lugar destinado a desarrollar una actividad docente.
- **Tipo de frecuencia**: Tipo de actividad impartida, puede ser conferencia, clase práctica, seminario, laboratorio o taller.
- **Optimización:** Selección del mejor elemento de un conjunto, de acuerdo a un criterio dado (Computing-Society, 2015).
- **Profesor**: Persona que imparte o enseña una asignatura.

1.3 Problema de planificación de horarios docentes

Existen diferentes conceptos relacionados con el problema de la panificación de horarios docentes, según (Schaerf, 1999) consiste en programar una secuencia de clases entre profesores y estudiantes en un prefijado período de tiempo (típicamente una semana), satisfaciendo un conjunto de restricciones de varios tipos.

1.3.1 Tipos de restricciones

El término restricciones se define como las condiciones que debe satisfacer el horario generado, en la literatura estudiada se clasifican en dos tipos (Schaerf, 1996), (Larrosa, y otros, 2003), (Chávez Bosquez, y otros, 2014):

Restricciones fuertes (duras): Condiciones que deben cumplirse obligatoriamente, la violación de alguna origina un horario no válido.

Restricciones débiles (suaves): Condiciones que denotan preferencias del usuario y se desea que se cumplan en la medida de lo posible. En muchos casos no pueden ser cumplidas de forma simultánea.

1.3.2 Complejidad computacional

La teoría de la complejidad computacional, como parte de la teoría de la computación, estudia los recursos requeridos por un algoritmo para resolver un problema (Garey, y otros, 1979). Utilizando los parámetros tiempo y espacio que se traducen como el número de pasos de ejecución de un algoritmo y la memoria utilizada para resolver un problema respectivamente, se pueden clasificar los problemas que puede resolver una computadora en tres clases P, NP y NP-Completo.

Problemas P

La clase de problemas que se pueden resolver mediante algoritmos eficientes se denota como P, indicando tiempo polinomial. En otras palabras, si el tiempo de ejecución de un algoritmo es menor que un valor calculado a partir del número de variables implicadas (generalmente variables de entrada) usando una fórmula polinómica, se dice que dicho problema pertenece a la clase P. La mayor parte de estos problemas (también conocidos como tratables) tienen soluciones prácticas utilizando un algoritmo determinístico (Galve Frances, 1993), (Arora, y otros, 2009).

Problemas NP

Un problema pertenece a la clase NP si este puede ser resuelto en tiempo polinomial pero utilizando un algoritmo no determinístico. El concepto de no-determinismo implica que un algoritmo al ser confrontado con varias opciones, puede determinar la correcta, nunca haciendo elecciones incorrectas que le lleven a un paso previo. La importancia de esta clase de problemas de decisión es que contiene muchos problemas

de búsqueda y de optimización para los que se desea saber si existe una cierta solución o si existe una mejor solución que las conocidas (Galve Frances, 1993), (Arora, y otros, 2009).

Problemas NP-Completo

La clase de complejidad NP-Completo es un subconjunto de los problemas de decisión NP tal que todo problema NP se puede convertir en cada uno de los problemas NP-Completos. Esta clase de problemas son los más difíciles NP y es probable que no formen parte de la clase de complejidad P (Garey, y otros, 1979), (Arora, y otros, 2009). La razón es que de tenerse una solución polinomial para un problema NP-Completo (también conocidos informalmente como intratables), todos los problemas NP tendrán también una solución en tiempo polinomial.

El problema de planificación de horarios docentes es un subproblema de asignación de recursos que, según (Baquero, y otros, 2008), es clasificado como NP-completo, la forma de resolverlo es a través de algoritmos computacionales. Esta clasificación determina que no existe un algoritmo determinístico que encuentre la solución al problema en un tiempo razonable.

1.4 Métodos de solución

En (Baquero, y otros, 2008) se plantea que para dar solución al problema de la planificación de horarios docentes existen dos tipos de modelos según el enfoque de desarrollo que han alcanzado: Los modelos matemáticos, que tienen una base matemática o se apoyan en ella, y los heurísticos, que utilizan métodos algorítmicos.

- **Métodos matemáticos:** Una de las ventajas del método matemático es que permite mencionarlo como el más riguroso y exacto, pero con la desventaja de su difícil formulación matemática sumada a la gran cantidad de recursos computacionales que requiere (Baquero, y otros, 2008).
- **Métodos heurísticos:** Los métodos heurísticos o aproximados son procedimientos eficientes para encontrar buenas soluciones aunque no se pueda comprobar que sean óptimas. En estos métodos, la rapidez del proceso es tan importante como la calidad de la solución obtenida. Además representan una facilidad para la puesta en marcha en el computador y en la confección algorítmica; sin embargo, la principal desventaja es el poco énfasis que le da al modelo matemático aun cuando este haya sido planteado desde el inicio del problema, por lo cual impide el desarrollo de una aplicación general (Baquero, y otros, 2008).

Teniendo en cuenta que no se trata de un sistema que implique la pérdida de vidas o recursos valiosos, no se considera necesario encontrar la solución óptima, cuya búsqueda involucra un alto grado de complejidad, sumado a la gran cantidad de recursos computacionales que se requieren. Se opta por el uso de métodos

heurísticos, debido a que es necesario encontrar una solución factible que cumpla con las restricciones fuertes y minimice las violaciones de las restricciones débiles en un tiempo razonable.

1.4.1 Metaheurísticas

El término metaheurística se obtiene de anteponer a heurística el sufijo “meta” que significa “más allá” o “a un nivel superior”. Los conceptos actuales de metaheurística están basados en las diferentes interpretaciones de lo que es una forma inteligente de resolver un problema, son estrategias inteligentes para diseñar o mejorar procedimientos heurísticos muy generales con un alto rendimiento. A partir de entonces han surgido diferentes propuestas de pautas para diseñar procedimientos que permitan resolver el problema de la planificación de horarios docentes (Moreno Pérez, 2004). A continuación se presentan algunas de las metaheurísticas utilizadas para resolver el problema de planificación de horarios docentes.

Algoritmos genéticos

Son métodos sistemáticos para la resolución de problemas de búsqueda y optimización que aplican métodos de la evolución biológica. Consisten en hallar de qué parámetros depende el problema, codificarlos en un cromosoma y aplicar métodos de la evolución como selección y reproducción sexual con intercambio de información y alteraciones que generen diversidad (Merelo, 2004), (F. Suárez, y otros, 2012), (Cobas Friman, 2012).

El algoritmo puede demorar en converger y en dependencia de los parámetros que se usen al implementarlo, puede converger prematuramente. Este problema se presenta en poblaciones pequeñas, donde una variación aleatoria en el ritmo de reproducción provoca que un genotipo se haga dominante sobre los otros. Este tipo de algoritmo es independiente del problema, lo cual lo hace un algoritmo robusto, por ser útil para cualquier situación, pero a la vez débil, pues no está especializado en ninguno.

Colonia de hormigas

Es una metaheurística capaz de encontrar soluciones de buena calidad a problemas de optimización altamente complejos. Se basa en la habilidad que poseen las hormigas naturales para encontrar el alimento a partir de individuos relativamente simples pero con una estructura social altamente eficiente. La base en ambos sistemas (natural y artificial) es la comunicación indirecta entre todos los individuos a partir de rastros de feromonas (Peñuela, 2008), (Mora García, 2009), (Pedemonte, 2009).

Presenta la desventaja de que en algunos casos no se obtiene un óptimo local, debido a que la búsqueda cesa de forma prematura a medida que la cantidad de feromonas aumenta. Además la población de hormigas se mantiene constante al realizar las iteraciones suponiendo un gasto innecesario de recursos pues mientras que el espacio de búsqueda se hace más pequeño la colonia mantiene su tamaño.

Procedimientos de búsqueda adaptativos, aleatorios y ágiles

Conocidos por su acrónimo en inglés GRASP *(Greedy Randomized Adaptive Search Procedures)* es un proceso iterativo en el que cada iteración consta de dos fases, una de construcción, en la cual una solución factible se produce, y una fase de búsqueda local, en la que se busca un óptimo local en la vecindad de la solución construida. La mejor solución global se mantiene como el resultado. Una característica especialmente atractiva de GRASP es la facilidad con la que se puede implementar. Pocos parámetros deben establecerse por lo tanto, el desarrollo puede centrarse en la aplicación de estructuras de datos eficientes para asegurar iteraciones GRASP rápidas (Festa, y otros, 2009), (Larico Mullisaca, 2010), (Borjas Giraldo, 2013).

Su principal desventaja como generador de soluciones iniciales para búsqueda local es su falta de diversidad. Además al ser un algoritmo totalmente aleatorio produce una amplia diversidad de soluciones, sin embargo la calidad de las mismas es muy pobre y si se usan como soluciones iniciales en la mayoría de los casos conduce a una convergencia lenta hacia un mínimo local.

Recocido simulado

La técnica de recocido simulado es una técnica de búsqueda aleatoria utilizada en la solución de problemas combinatorios de optimización. Este algoritmo parte de una solución inicial dada, de una función objetivo que depende de la solución y de un parámetro de control T conocido como temperatura, que es una función decreciente del número de iteraciones. En cada iteración se genera al azar una solución en el vecindario de la mejor solución encontrada hasta el momento. Si la solución generada es mejor que la solución actual esta se adopta como la nueva solución y se continúa la búsqueda. Si por el contrario la solución encontrada no es mejor que la solución actual existe la posibilidad de que el algoritmo acepte esta solución si la temperatura es mayor que un número aleatorio U (Torres Delgado, y otros, 2007).

En (Osorio, y otros, 2012) se plantea que este método es fácil de comprender y de implementar debido a que su principio es muy sencillo, pero su complejidad radica en el cálculo de sus parámetros, ya que no existe un estándar comprobado para estos. Además se evidencia que en muchas iteraciones se repite algún resultado generado en iteraciones anteriores, ya que el algoritmo no tiene mecanismo de memoria, lo que aumenta el tiempo de ejecución y el tiempo necesario para alcanzar la mejor solución.

Búsqueda tabú

En (Chávez Bosquez, y otros, 2005) se plantea a la búsqueda tabú como una metaheurística, ya que incluye en sus propias reglas algunas técnicas heurísticas, su papel es dirigir y orientar la búsqueda de otro procedimiento más local de búsqueda y explora el espacio de soluciones más allá del óptimo local. Esta técnica ha sido utilizada con éxito en diversas universidades, obteniéndose resultados de buena calidad, según pruebas de evaluación internacionales.

La búsqueda tabú mantiene una lista de movimientos tabú, lo que representa horarios que, habiendo sido visitados recientemente, están prohibidos para evitar ciclos en la búsqueda y así escapar de óptimos locales. La lista tabú es generalmente de un tamaño fijo, los movimientos más antiguos se eliminan a medida que se añaden nuevos movimientos. Como los movimientos tabú pueden impedir alcanzar la búsqueda de nuevas soluciones mejoradas, se mantiene a menudo un nivel de aspiración, lo que representa la mejor solución visitada hasta el momento. Si un calendario tabú alcanza el nivel de aspiración, puede ser eliminado de la lista tabú (Burke, y otros, 1997).

Después del estudio de las metaheurísticas aplicadas en la optimización de horarios y coincidiendo con (Glover, y otros, 1993), (De la Cruz H., y otros, 2003) y (Chávez Bosquez, y otros, 2005) se selecciona el método búsqueda tabú como alternativa de optimización del horario docente de la Facultad de Ingeniería Industrial de la CUJAE. Esta metaheurística incluye medidas o estrategias que conducen un proceso de exploración de manera tal que el espacio de soluciones de un problema sea examinado, evitando la optimalidad local. Aplica la exploración sensitiva debido a que muchas veces una mala selección estratégica puede contener información más importante que una buena selección al azar. Intensifica la búsqueda en las regiones del espacio de soluciones que encuentra atractivas, en términos de calidad, explorando detenidamente las vecindades de las mejores soluciones encontradas e induce al procedimiento de búsqueda para explorar regiones no visitadas, como una estrategia de diversificación.

1.5 Soluciones informáticas existentes

A continuación se presentan soluciones informáticas a nivel internacional y nacional relacionadas con el problema de la planificación de horarios docentes.

Internacionales

- **Sistema inteligente de soporte en la generación de horarios académicos para la carrera de Ingeniería de Sistemas de la Universidad Salesiana**: Este sistema propone el uso de algoritmos genéticos para resolver problemas de optimización combinatoria en la generación de horarios, sin embargo no incluye la planificación de asignaturas optativas (Pineda Arias, y otros, 2011).
- **Aplicación para la generación de horarios en la web:** Es un sistema que se encuentra totalmente en línea por lo que no necesita instalación. Permite la edición manual así como las opciones de asignación de grupos articulados. Esta aplicación web no incluye funcionalidades para la reservación de asignaturas optativas, además de que no utiliza algoritmos metaheurísticos para la optimización de los horarios generados (Timetable Web, 2012).

- **Untis:** Desarrollado con tecnología privativa (no es de código abierto), de carácter comercial, posee una versión gratuita con licencia temporal de tres días pero limita el número de grupos y no permite guardar los datos registrados (Untis, 2014).
- **Open Course Timetabler:** Aplicación de código abierto, gratuito, desarrollado en la plataforma .NET de Windows. Sus principales desventajas radican en la no existencia de una interfaz gráfica de usuario, las interacciones se realizan desde la línea de comandos, además solo se encuentra disponible en inglés y croata (Curak, 2007).

Los sistemas internacionales estudiados no serán utilizados, pues son de carácter comercial o están desarrollados con software privativo y en nuestro país se opta por el uso de software libre para lograr la independencia tecnológica en materia de software, además la mayoría de estos sistemas no están disponibles en idioma español.

Nacionales

- **Sistema informático para la confección de horarios docentes en la Facultad de Informática de la Universidad de Cienfuegos**: Sistema capaz de generar el horario acorde a las expectativas propias de la institución que utiliza la búsqueda tabú como método de optimización. Es una solución implementada a la medida, acorde a las necesidades de la institución y no incluye la planificación de asignaturas electivas ni optativas (Echevarría Cartaya, 2009).
- **Horr:** Sistema de planificación y publicación de horarios docentes usado en la Universidad de las Ciencias Informáticas, compuesto por una aplicación de escritorio (HorrPlanner) para la elaboración de forma manual del horario docente y una aplicación web (HorrPublisher) para su posterior publicación en la web. Es un asistente de planificación que no permite generar el horario docente y no verifica si la actividad que se va a realizar esta acorde al local que se le asigna. Además no presenta una buena estructura organizacional en cuanto a la clasificación de locales, actividades y asignación de profesores (Barrera, 2011).
- **Sistema para la generación del horario docente de la Facultad de Ingeniería Industrial del CUJAE:** Permite la generación y publicación del horario docente en un tiempo razonable, teniendo en cuenta las características y restricciones propias de la institución. No permite la planificación de asignaturas electivas ni optativas y realiza un inadecuado tratamiento de las restricciones débiles (Pérez Martínez, y otros, 2014).

Los sistemas nacionales estudiados no cumplen con los requerimientos necesarios para ser usados como solución al problema de la planificación de horarios docentes que presenta la Facultad de Ingeniería Industrial de la CUJAE, pues son soluciones desarrolladas a medida para cada institución o no permiten la

generación del horario. Por estos motivos se pretende incluir nuevas funcionalidades al sistema existente en la Facultad de Ingeniería Industrial de la CUJAE, en aras de obtener un horario completo y disminuir el número de violaciones de las restricciones débiles que presenta la solución actual.

1.6 Metodología de desarrollo de software

Las metodologías de desarrollo de software están definidas como un conjunto de procedimientos, herramientas y soporte documental que ayuda a los desarrolladores a realizar nuevo software. De manera general se consideran como un conjunto de pasos y procedimientos que deben seguirse para el desarrollo del software y se pueden agrupar en dos grandes grupos (Piattini Velthuis, 1996):

- **Metodologías tradicionales o pesadas:** Hacen mayor énfasis en la planificación y control del proyecto, en especificación precisa de requisitos y modelado, estableciendo estrictamente las actividades involucradas, los roles definidos, los artefactos que se deben producir, las herramientas y la documentación usada.
    - RUP *(Rational Unified Procces)*
    - MSF *(Microsoft Solution Framework)*
    - Win-Win Spiral Model
    - Iconix
- **Metodologías ágiles o ligeras:** Orientadas a la generación de código con ciclos muy cortos de desarrollo manteniendo un proceso incremental, son capaces de permitir cambios en los requisitos de último momento, además el equipo de desarrollo mantiene una comunicación constante con el cliente.
    - XP *(Extreme Programming)*
    - SCRUM
    - Crystal Clear
    - FDD *(Feature-Driven Development)*
    - DSDM *(Dynamic Systems Development Method)*
    - ASD *(Adaptive Software Development)*

Se muestra un listado de los principios de las metodologías ágiles propuesto por (Beck, y otros, 2001) en el Manifiesto por el desarrollo ágil del software.

- La mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
- Aceptar los cambios en los requisitos, incluso en etapas tardías del desarrollo. Los procesos ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
- Entregas de software funcional frecuentemente con preferencia al período de tiempo más corto posible.
- Los responsables de negocio y los desarrolladores trabajan juntos de forma cotidiana durante todo el proyecto.

Se opta por el uso de una metodología ágil porque en la investigación se hace mayor énfasis en la obtención del software funcional en poco tiempo sobre documentación extensiva, en la comunicación continua entre el equipo de desarrollo y el cliente sobre la negociación contractual, así como respuestas rápidas ante cambios de última hora sobre el seguimiento de un plan, coincidiendo con lo expresado en el Manifiesto por el desarrollo ágil del software (Beck, y otros, 2001).

Metodologías ágiles

A continuación se describen dos de las metodologías de desarrollo ágil más referenciadas (Letelier Torres, y otros, 2003), (Mendes Calo, y otros, 2009), (Palacio, 2014).

SCRUM

En (Palacio, 2014) se define como un modelo de desarrollo ágil caracterizado por:

- Adoptar una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto.
- Establecer la calidad del resultado más en el conocimiento tácito de las personas en equipos organizados, que en la calidad de los procesos empleados.
- Solapamiento de las diferentes fases del desarrollo, en lugar de realizarlas una tras otra en un ciclo secuencial o de cascada.

Las iteraciones en SCRUM son la base del desarrollo ágil y condicionan su evolución a través de reuniones breves diarias en las que el equipo revisa el trabajo realizado el día anterior y el previsto para el día siguiente. Según lo planteado en (Villarreal, 2008) lo primero que aclaran los creadores de SCRUM es que no se debe seguir el método al pie de la letra, sino que se debe adaptar a las necesidades de cada organización. Además especifican que para que la metodología funcione deben realizarse todos los pasos de manera

apropiada. Son especialmente importantes las reuniones y las interacciones entre las personas, así como también la responsabilidad de cada integrante en el proyecto.

Programación Extrema (*Extreme Programming*, XP en adelante)

Se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios (Letelier Torres, y otros, 2003). Se diferencia de otros métodos para la construcción de software en conceptos como (Jackson, y otros, 2004):

- Retroalimentación por parte del usuario en ciclos cortos y continuos.
- Enfoque de planificación incremental, la cual genera un plan global de desarrollo, el cual se espera que evolucione a lo largo de la vida del proyecto.
- Capacidad para programar en forma flexible la implementación de funcionalidades, respondiendo a las necesidades cambiantes de negocios.
- Confianza en pruebas de software automatizadas, definidas y escritas por programadores y clientes para controlar el proceso de desarrollo, permitiendo la evolución del sistema y captando los defectos lo antes posible.
- Confianza en la comunicación oral, las pruebas y el código fuente para obtener la estructura e intención del sistema.
- Confianza en el proceso de diseño evolutivo que perdura mientras lo haga el sistema.
- Confianza en la colaboración muy correlacionada entre todos los programadores involucrados.

Se selecciona la metodología XP porque se ajusta a las necesidades y características de la investigación, pues el equipo de trabajo está formado por dos desarrolladores en constante intercambio de información con el cliente, admitiendo en la solución cambios de última hora, siendo el período de trabajo aproximadamente de seis meses.

1.7 Lenguajes de programación

Los lenguajes de programación se utilizan para controlar el comportamiento de una máquina, particularmente una computadora. Consisten en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente (Saavedra Gutierrez, 2007).

Ruby: Lenguaje de programación interpretado, reflexivo y orientado a objetos. Diseñado para la productividad y la diversión del desarrollador, siguiendo los principios de una buena interfaz de usuario y su implementación oficial es distribuida bajo una licencia de software libre. Al ser orientado a objeto define

como tales a todos los tipos de datos, incluidas las clases y tipos que otros lenguajes definen como primitivas, (como enteros, booleanos, y nulos). Ha sido descrito como un lenguaje de programación multiparadigma que permite la programación procedural (definiendo funciones y variables fuera de las clases haciéndolas parte del objeto raíz) (Matsumoto, 2015). El tiempo de procesamiento puede llegar a ser demasiado lento, debido a esto no se opta por su uso.

Python: En la actualidad se desarrolla como un proyecto de código abierto. Este permite dividir el programa en módulos reutilizables desde otros programas Python. Viene con una gran colección de módulos estándar que se pueden utilizar como base de los programas. Es utilizado como lenguaje de programación interpretado, lo que ahorra un tiempo considerable en el desarrollo del programa, pues no es necesario compilar ni enlazar. El intérprete se puede utilizar de modo interactivo, lo que facilita experimentar con características del lenguaje, escribir programas desechables o probar funciones durante el desarrollo del mismo (Martelli, 2008). Python al ser un lenguaje interpretado es más lento que lenguajes compilados o de ensamblador, además conforme se crean aplicaciones más complejas es más complicado escribir el código, así como que posee baja oferta de servicios de alojamiento en la web que lo soportan debido a que es muy complejo implementar esta tecnología en web (Martín García, 2013).

Java: Lenguaje de programación orientado a objetos. El lenguaje en sí mismo toma mucha de su sintaxis de Lenguaje de Programación C y C++, con un modelo de objetos más simple. Las aplicaciones Java están típicamente compiladas en un *bytecode*, en tiempo de ejecución, el *bytecode* es normalmente interpretado o compilado a código nativo para la ejecución. Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de forma que prácticamente todo el Java de Sun es ahora software libre (aunque la biblioteca de clases de Sun que se requiere para ejecutar los programas Java aún no lo es). Otra característica está su independencia de la plataforma, significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware y sistema operativo (Gosling, y otros, 2005).

PHP: Lenguaje interpretado, multiplataforma y diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor *(server-side scripting)* pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK+. Publicado bajo la licencia de PHP, la Fundación de Software Libre considera esta licencia como software libre (Lerdorf, 2015).

El código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Tiene la capacidad de expandir su potencial utilizando la enorme cantidad de módulos existentes (llamados ext's o extensiones), así como conectarse con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL (Welling, y otros, 2005).

Luego del estudio de los lenguajes de programación expuestos y siguiendo los requerimientos con que debe cumplir el sistema se opta por el uso de la programación web, usando PHP como lenguaje de programación y Java para la implementación del algoritmo de optimización. La posibilidad de la comunidad universitaria con acceso a la red escolar de la CUJAE, pueda consultar los resultados del sistema es una ventaja importante que ofrecen estos lenguajes de programación.

1.8 Tecnologías

jQuery 1.9

Es una biblioteca de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. jQuery es software libre y de código abierto (The jQuery Foundation, 2015).

Bootstrap 2

Es un marco de trabajo de software libre para el diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales. La configuración de Bootstrap tiene la opción de personalizar en la documentación. Los desarrolladores eligen en un formulario los componentes y ajustes deseados. El paquete incluye la hoja de estilo pre-compilada (Sears, y otros, 2015).

1.9 Herramientas

A continuación se muestran las herramientas utilizadas en el desarrollo de la solución de acuerdo a los lenguajes de programación seleccionados.

Visual Paradigm 8.0

Herramienta para desarrollo de aplicaciones utilizando modelado UML, ideal para quienes están interesados en construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. Constituye una herramienta privada pero en la Universidad de las Ciencias Informáticas, centro donde estudian los autores de la investigación, se tiene una licencia que es usada con fines educativos. Dentro de sus características se tiene (Visual Paradigm International, 2015):

- Modelo y código permanecen sincronizados en todo el ciclo de desarrollo
- Interoperabilidad con modelos UML
- Generación de bases de datos
- Ingeniería inversa de bases de datos

Debido a la experiencia en el uso de esta herramienta CASE en los proyectos productivos y porque es la herramienta de modelado utilizada en la confección del sistema de generación de horarios al que se integrarán las nuevas funcionalidades se opta por su utilización.

PostgreSQL 9.2

Es un sistema gestor de base de datos (SGBD) relacional orientado a objetos y libre, publicado bajo licencia BSD *(Berkeley Software Distribution)* y con su código fuente disponible libremente. Utiliza un modelo cliente-servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. La aplicación pgAdmin III es el entorno de escritorio visual que permite conectarse a las bases de datos del PostgreSQL que estén ejecutándose en cualquier plataforma facilita la gestión y administración de bases de datos, ya sea mediante instrucciones SQL o con ayuda de un entorno gráfico. Dentro de las características principales del PostgreSQL se encuentran (Martinez, 2010):

- Alta concurrencia
- Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando
- Copias de seguridad en caliente
- Múltiples métodos de autentificación
- Acceso encriptado vía SSL

Se opta por su uso debido a las características mencionadas y porque es el sistema gestor de base de datos utilizado en el sistema en el que se incluirá el módulo de asignaturas optativas y la optimización del horario docente.

NetBeans IDE 8.0

Permite programar en distintos lenguajes en los cuales incluye PHP, HTML, CSS y JavaScript, además presenta una serie de características que facilitan en gran parte el desarrollo:

- Multiplataforma
- Historial de cambios
- Facilidades para la programación: Completamiento de código, comprobación y corrección de errores en tiempo real, resaltado de variables o etiquetas seleccionadas

Se opta por esta última herramienta debido a que es un producto libre, gratuito, sin restricciones de uso y se ajusta a las necesidades de la investigación.

1.10 Validación del diseño

Para la evaluación de la calidad del diseño se propone el uso de las métricas Tamaño operacional de clase (TOC) y Relaciones entre clases (RC) propuestas en (Lorenz, y otros, 1994).

TOC: Está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

- Responsabilidad: Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
- Complejidad de implementación: Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
- Reutilización: Un aumento del TOC implica una disminución del grado de reutilización de la clase.

RC: Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

- Acoplamiento: Un aumento del RC implica un aumento del acoplamiento de la clase.
- Complejidad de mantenimiento: Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
- Reutilización: Un aumento del RC implica una disminución en el grado de reutilización de la clase.
- Cantidad de pruebas: Un aumento del RC implica un aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

1.11 Evaluación de la solución

Las pruebas de software son las investigaciones empíricas y técnicas cuyo fin es proporcionar información objetiva e independiente sobre la calidad del producto. Esta actividad forma parte del proceso de control de calidad global y reducen la probabilidad de que aparezcan defectos ocultos en el software (Barrientos, 2014). Para la evaluación de la solución se proponen las pruebas de caja blanca y caja negra.

Pruebas de caja blanca

Las pruebas de caja blanca del software se basan en un examen cercano al detalle procedimental. Se prueban las notas lógicas del software y la colaboración entre componentes, al proporcionar casos de pruebas que ejerciten conjuntos específicos de condiciones, bucles o ambos (Pressman, 2005).

El camino básico es una técnica de prueba de caja blanca propuesta inicialmente por McCabe en (McCabe, 1976). En (Pressman, 2005) se plantea que permite al diseñador de casos de prueba obtener una medida

de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

La complejidad ciclomática es una métrica de software que proporciona una medida cuantitativa de la complejidad lógica de un programa. Cuando se emplea en el contexto del método de prueba de la ruta básica, el valor calculado mediante la complejidad ciclomática define el número de rutas independientes en el conjunto básico de un programa, y proporciona un límite superior para el número de pruebas que deben aplicarse para asegurar que todas las instrucciones se hayan ejecutado por lo menos una vez (Pressman, 2005). La complejidad ciclomática, V (G), se calcula de una de tres maneras:

- El número de regiones corresponde a la complejidad ciclomática.
- V (G) = E - N + 2, donde E es el número de aristas y N el número de nodos.
- V (G) = P + 1, donde P es el número de nodos predicado (son aquellos de los que salen varios caminos).

Pruebas de caja negra

Las pruebas de caja negra se aplican a la interfaz del software, examinan algún aspecto funcional de un sistema que tiene poca relación con la estructura lógica del software y tratan de encontrar errores en las siguientes categorías (Pressman, 2005):

- Funciones incorrectas o ausentes
- Errores de interfaz
- Errores en la estructura de datos o en accesos a bases de datos externas
- Errores de rendimiento
- Errores de inicialización y terminación

Prueba de aceptación

En las pruebas de aceptación se especifican, desde la perspectiva del cliente, los escenarios para probar de cada funcionalidad y su correcta implementación. El objetivo final de estas pruebas es garantizar que los requerimientos han sido cumplidos y que el sistema es aceptable (Rodríguez Corbea, y otros, 2007).

1.12 Conclusiones parciales

La investigación realizada evidenció la necesidad de implementar nuevas funcionalidades al sistema existente para la generación del horario docente de la Facultad de Ingeniería Industrial de la CUJAE.

El estudio de los métodos utilizados para darle solución al problema de planificación de horarios permitió la selección de la metaheurística búsqueda tabú para la optimización del horario generado.

El análisis del trabajo de diploma predecesor, al cual se le da continuación, permitió la selección de las herramientas, lenguajes y metodología que regirán el desarrollo del presente trabajo definiendo a XP como metodología de desarrollo de software, PHP 5.4 como lenguaje de programación en el lado del servidor, Visual Paradigm 8.0 para el modelado del sistema, PostgreSQL 9.2 como sistema gestor de bases de datos, NetBeans 8.0 como entorno de desarrollo integrado para la implementación de la solución y Java como lenguaje para la implementación y creación de los componentes de la búsqueda tabú con el objetivo de optimizar las restricciones débiles.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

2.1 Introducción

El presente capítulo incluye las principales características de la propuesta de solución acorde a la metodología de desarrollo de software seleccionada. Se definen los requisitos funcionales y no funcionales que debe cumplir el sistema, además se delimitan las nuevas funcionalidades como la optimización de restricciones débiles, la arquitectura y los patrones de diseño empleados.

2.2 Restricciones del sistema

Para una mayor comprensión del sistema a continuación se muestran las restricciones fuertes que se satisfacen en la solución del trabajo de diploma predecesor y las restricciones débiles a optimizar.

Restricciones fuertes

1. No puede haber clases simultáneas en un mismo local.
2. No puede haber clases simultáneas asignadas a un mismo grupo.
3. No puede haber clases simultáneas asignadas a un mismo profesor.
4. Una asignatura no puede tener más de una frecuencia de clase un mismo día.
5. No se puede asignar clases consecutivas en el tercer y cuarto turno.
6. No planificar clases a profesores, grupos o en locales los días/turnos con afectaciones previamente declaradas.
7. El horario debe ser compacto, no debe dejar turnos intermedios sin planificar.

Restricciones débiles

1. Un grupo no puede tener clases en un local que no tenga la capacidad que requiere.
2. El horario debe ser compacto.
3. Después de una clase de tipo conferencia se debe dejar, al menos, un día de diferencia con relación al próximo turno de clase de la asignatura.
4. Los grupos deben tener clases en un mismo local, el cambio de local debe ser mínimo.
5. La asignatura Educación física no debe tener clase que la preceda o la suceda.
6. A los profesores que imparten clase a más de un grupo, de una misma asignatura, se les debe planificar clases en turnos consecutivos un mismo día.
7. Las clases de laboratorio deben planificarse en la sección de clases que corresponde.

2.3 Abreviaturas del sistema

En las siguientes tablas se presentan las nuevas abreviaturas que se utilizan en el sistema.

Asignaturas electivas de segundo año	
Asignatura electiva 1	AE1
Asignatura electiva 2	AE2

Tabla 1. Abreviatura de las asignaturas electivas de segundo año

Asignaturas optativas de tercer año	
Organización del trabajo en el grupo empresarial	OTGE
Patrones para el modelado de procesos de negocio	PMPN
Organización del trabajo y sistema de gestión integrada al capital humano	OTSG
Educación vial	EV
Gestión del mantenimiento	GM
Procesos tecnológicos de la industria ligera	PTIL
Ergonomía ocupacional	EO
Diseño y procesamiento de encuestas	DPE
Derecho internacional	DI
VI para la gestión de AE	VIG
Gestión del conocimiento	GC

Tabla 2. Abreviatura de las asignaturas optativas de tercer año.

Asignaturas optativas de cuarto año	
Unidad docente de economía	UDE
Unidad docente MININT	UDM
Dirección por valores	DPV
Inocuidad alimentaria	IA
Comunicación organizacional	CO

Tabla 3. Abreviatura de las asignaturas optativas de cuarto año

Tipo de frecuencia	
Conferencia	C
Clase práctica	CP
Seminario	S
Laboratorio	L
Taller	T
Prueba inter-semestral	E

Tabla 4. Abreviatura de los tipos de frecuencia

Locales	
Aula	A

Aula de conferencia	AC
Aula especializada	AE
Laboratorio	L
Taller	T

Tabla 5. Abreviatura de locales

2.4 Búsqueda tabú para la optimización de restricciones débiles

Para minimizar la violación de las restricciones débiles se emplea la búsqueda tabú, metaheurística que guía a un procedimiento de búsqueda local impidiendo que la solución caiga en óptimos locales mediante el uso de estructuras de memoria y de la exploración sensitiva.

La búsqueda tabú permite moverse a una solución aunque no sea tan buena como la actual, de modo que se pueda escapar de óptimos locales y continuar estratégicamente la búsqueda de soluciones aún mejores (Díaz, y otros, 1996). A continuación se describen los componentes mínimos de esta metaheurística y su forma de representación en la propuesta de solución.

2.4.1 Solución inicial

Se requiere iniciar el proceso de exploración con una solución, puede ser aleatoria o generada mediante otro algoritmo. Se debe tener en cuenta que el desempeño final del algoritmo dependerá en gran medida del grado de factibilidad de la solución inicial (Baquero, y otros, 2008).

La solución inicial a utilizar es la generada por el sistema existente, debido a que satisface las restricciones fuertes establecidas y coincidiendo con (Baquero, y otros, 2008) en que la selección de una buena configuración inicial puede conducir a soluciones de alta calidad con bajos esfuerzos computacionales.

- Nombre de clase: *Solution*
- Definición: Contiene la configuración de una variación de la semana de clases y un valor numérico que representa el número de restricciones débiles violadas.

2.4.2 Lista tabú

Según (Glover, y otros, 1993) radica en la lista de movimientos que se consideran prohibidos, pues han sido utilizados durante el proceso de búsqueda para generar una solución. El rol principal de este elemento es prevenir ciclos en la búsqueda. Si el tamaño de la lista es muy pequeño entonces se puede regresar a

soluciones visitadas y caer en ciclos; un tamaño muy grande puede volver muy restrictiva la búsqueda. Desafortunadamente, en un problema de optimización es difícil o incluso imposible encontrar un valor que evite caer en ciclos y que no limite excesivamente la búsqueda para todas las instancias de un problema de tamaño dado.

Para establecer el tamaño de la lista tabú, que se traduce al número de iteraciones que un movimiento es considerado prohibido, se deja el parámetro abierto para permitir al usuario asignar el valor que considere más conveniente.

- Nombre de clase: *TabuList*
- Definición: Mantiene una colección de elementos del tipo *TabuElement* que contienen soluciones generadoras y un entero que define el tiempo tabú de dicha solución.

2.4.3 Esquema de vecindad

En lugar de explorar todo el espacio de soluciones, como ocurre con un algoritmo de fuerza bruta, la búsqueda tabú realiza una búsqueda en un subconjunto de este espacio, llamado vecindad, que es un entorno reducido que debe ser seleccionado cuidadosamente de acuerdo a las características particulares del problema. Con el objetivo de limitar el espacio de búsqueda se utilizó la siguiente vecindad.

Vecindad de intercambios aleatorios

La vecindad está compuesta por intercambios de:

- Posición de dos eventos seleccionados aleatoriamente.
- Posición de dos eventos que se encuentran en dos salones seleccionados aleatoriamente, en un mismo período de tiempo.
- Posición de dos eventos que se encuentran en dos períodos de tiempo seleccionados aleatoriamente, en un mismo local.

Para que un intercambio sea considerado válido al menos uno de los eventos no debe ser vacío. La principal ventaja de esta vecindad es la diversificación de la búsqueda.

- Nombre de clase: *MoveManager*
- Definición: Determina cuáles movimientos están disponibles en un tiempo dado. Es recomendable reutilizar los movimientos generados para evitar una constante instanciación para cada iteración.

2.4.4 Criterio de aspiración

Este criterio se define para los movimientos considerados prohibidos, se introduce en la búsqueda tabú para determinar cuándo pueden ser reemplazadas o eliminadas las restricciones tabú sobre cierto elemento dentro de la lista tabú.

- Nombre de clase: *AspirationCriteria*
- Definición: Se define como un entero asociado a cada elemento tabú que indica el tiempo de permanencia de una solución en estado tabú.

2.4.5 Función objetivo

El proceso de optimización involucra el uso de una ecuación que contemple las restricciones a maximizar o minimizar, a esta ecuación se le conoce como función objetivo. Es posible asignar más prioridad a ciertas restricciones mediante valores constantes en la función objetivo. La forma en que el algoritmo explora el espacio de búsqueda puede cambiar dependiendo estos valores.

- Nombre de clase: *ObjectiveFunction*
- Definición: Es usada para evaluar una solución mediante la comprobación del cumplimiento de las restricciones débiles definidas.

Los elementos mencionados se utilizan para buscar el espacio de soluciones a través de una solución inicial. En cada iteración del algoritmo identifica la vecindad de la solución actual, la lista tabú reciente y el conjunto que satisface el criterio de aspiración al evaluarlo en la función objetivo. Este conjunto identifica las posibles soluciones que se encuentran en la lista tabú y que podrían ser aceptados. A continuación se muestra gráficamente el proceso.

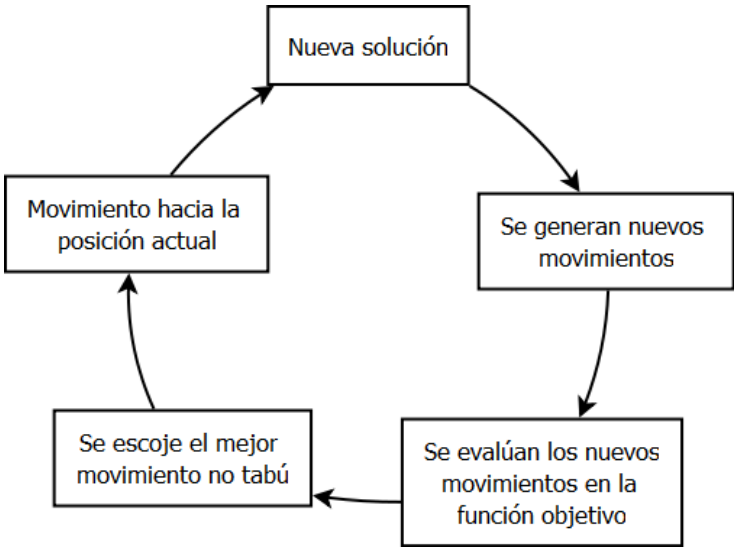

Figura 1. Iteración de la metaheurística búsqueda tabú (Elaboración propia)

Criterio de finalización del algoritmo

El proceso de búsqueda tabú finaliza si se cumple alguno de los siguientes criterios:

- Alcanzar una solución que sea menor o igual que una evaluación de dos en la función objetivo. El valor se establece en dos debido a que representa el menor criterio de aspiración para una posible solución.
- No obtener una mejor solución luego de evaluar en la función objetivo las diez posibles soluciones.

2.5 Fase exploración

Según (Letelier Torres, y otros, 2003) en esta fase los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas a emplear en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo.

2.5.1 Requisitos funcionales

Describen las interacciones entre el sistema y su ambiente, en forma independiente a su implementación. El ambiente incluye al usuario y cualquier otro sistema externo con el cual interactúe el sistema (Sommerville, 2002). A continuación se muestran los requisitos funcionales del sistema.

RF 1. Insertar asignatura optativa.

RF 2. Modificar asignatura optativa.

RF 3. Mostrar asignatura optativa.

RF 4. Eliminar asignatura optativa.

RF 5. Insertar asignatura electiva.

RF 6. Modificar asignatura electiva.

RF 7. Mostrar asignatura electiva.

RF 8. Eliminar asignatura electiva.

RF 9. Insertar bloque.

RF 10. Modificar bloque.

RF 11. Mostrar bloque.

RF 12. Eliminar bloque.

RF 13. Insertar secuencia de actividades.

RF 14. Modificar secuencia de actividades.

RF 15. Mostrar secuencia de actividades.

RF 16. Eliminar secuencia de actividades.

RF 17. Establecer local predeterminado para las asignaturas Dibujo Básico y Dibujo Aplicado.

RF 18. Planificar exámenes de asignaturas optativas.

RF 19. Extraer configuración inicial.

RF 20. Generar vecindad.

RF 21. Comprobar restricciones fuertes.

RF 22. Comprobar restricciones débiles.

RF 23. Seleccionar mejor configuración.

RF 24. Optimizar por intervalos.

RF 25. Mostrar cantidad de restricciones incumplidas.

2.5.2 Historia de usuario

XP propone las historias de usuario (HU) para especificar las funcionalidades del software. Se trata de tarjetas en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. En (Jeffries, y otros, 2001) se plantea que el tratamiento de las HU es muy dinámico y flexible, pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas.

A partir de los requisitos funcionales mencionados se definen cuatro HU descritas en el Anexo 1. Cada HU debe ser lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. A continuación se presenta la estructura de una HU.

Historia de usuario	
Número:	**Nombre:**
Usuario:	**Iteración asignada:**
Prioridad en negocio:	**Puntos estimados:**
Riesgo en desarrollo:	**Puntos reales:**

Descripción:
Observaciones:

Tabla 6. Estructura de una historia de usuario

Donde:

- Número: Identificador de la HU.
- Nombre: Nombre que identifica a la HU.
- Usuario: Involucrados en la ejecución de la HU (ver siguiente epígrafe: Personas relacionadas con el sistema).
- Iteración asignada: Iteración en que se implementará la HU.
- Prioridad en el negocio: Prioridad de la HU con respecto al resto de las HU (alta, media o baja).
- Riesgo en desarrollo: Riesgo en la implementación de la HU (alto, medio o bajo).
- Puntos estimados: Estima el esfuerzo asociado a la implementación de la HU. Un punto equivale a una semana ideal de programación, generalmente de uno a tres puntos.
- Puntos reales: Resultado del esfuerzo asociado a la implementación de la HU. Un punto equivale a una semana ideal de programación, generalmente de uno a tres puntos.
- Descripción: Descripción sintetizada de la HU.
- Observaciones: Información adicional.

2.5.3 Personas relacionadas con el sistema

En la siguiente tabla se definen los niveles de acceso de los usuarios que interactúan con el sistema.

Persona	**Descripción**
Usuario	Cualquier persona que acceda al sistema de forma anónima para consultar el horario docente.
Planificador	Usuario que se ha registrado y además posee los permisos correspondientes para interactuar y realizar modificaciones en el sistema.

Tabla 7. Personas relacionadas con el sistema

2.5.4 Requisitos no funcionales del sistema

Son propiedades o cualidades que el producto debe tener, son aspectos importantes que se deben cumplir para lograr un producto atractivo, usable, rápido y confiable (Pressman, 2005).

La metodología XP no describe estos requisitos en las HU, pues generalmente los clientes no conocen las terminologías técnicas aplicadas para su descripción. El equipo de desarrollo es el encargado de la captura de estos requisitos a partir del intercambio con el cliente, para la obtención de solución que cumpla con determinados estándares de calidad y con las características del negocio a informatizar. A continuación se presentan los requisitos no funcionales identificados.

Requisitos de usabilidad:

RNF1. La tipografía debe ser uniforme, de un tamaño adecuado y con un contraste que resalte los textos.

Requisitos de rendimiento:

RNF2. El sistema debe tener un tiempo máximo de 10 minutos para la generación del horario y cinco segundos para cualquier operación de consulta. La duración del proceso de optimización dependerá del tamaño de los datos de entrada.

Requisitos de portabilidad:

RNF3. El sistema se debe instalar en un servidor con una distribución de Linux, preferentemente CentOS o Ubuntu, por ser las de más soporte en Cuba; independientemente podrá ser instalado en sistemas operativos Windows.

Requisitos de seguridad:

Confidencialidad:

RNF4. Se establecerán mecanismos de autenticación a través de nombres de usuario y contraseñas que definan los permisos correspondientes a los involucrados con la planificación del horario docente.

RNF5. Los mensajes de error mostrados a los usuarios deben ser genéricos sin dar detalles de información, para no comprometer la seguridad e integridad de los datos.

RNF6. Se deben cifrar las contraseñas.

RNF7. Se establecerán requisitos para la fortaleza de la contraseña (longitud mínima y caracteres especiales).

RNF8. El sistema podrá estar inactivo durante diez minutos, al cabo de este tiempo se cerrará la sesión que esté abierta.

Requisitos de software:

Cliente:

RNF9. Navegador web a partir de las siguientes versiones: Internet Explorer 8, Mozilla Firefox 4, Google Chrome 16 y Opera 9.1.

Servidor:

RNF10. Gestor de base de datos PostgreSQL 9.2

RNF11. Servidor web Apache 2.6.6

RNF12. PHP 5.4.16

RNF13. Entorno de ejecución de Java versión 7

Requisitos de hardware:

Cliente:

RNF17. Memoria RAM: 256Mb

RNF18. Microprocesador: 1.70GHz doble núcleo

RNF19. Tarjeta de red

Servidor:

RNF20. Memoria RAM: 2 Gb

RNF21. Microprocesador: 2.1 GHz doble núcleo

RNF22. Tarjeta de red

2.6 Modelo de datos

En el modelo de datos se describe la estructura lógica y física de la información que se persiste en el sistema. Se establecen las relaciones de las tablas usadas para generar el horario docente. La tabla “encuentro” se define como la principal, pues se encarga de almacenar el horario final una vez que el proceso de generación y optimización concluya. Se utilizan nomencladores para almacenar los datos fijos que podrán gestionarse desde el sistema. En la siguiente figura se muestra el diagrama entidad-relación.

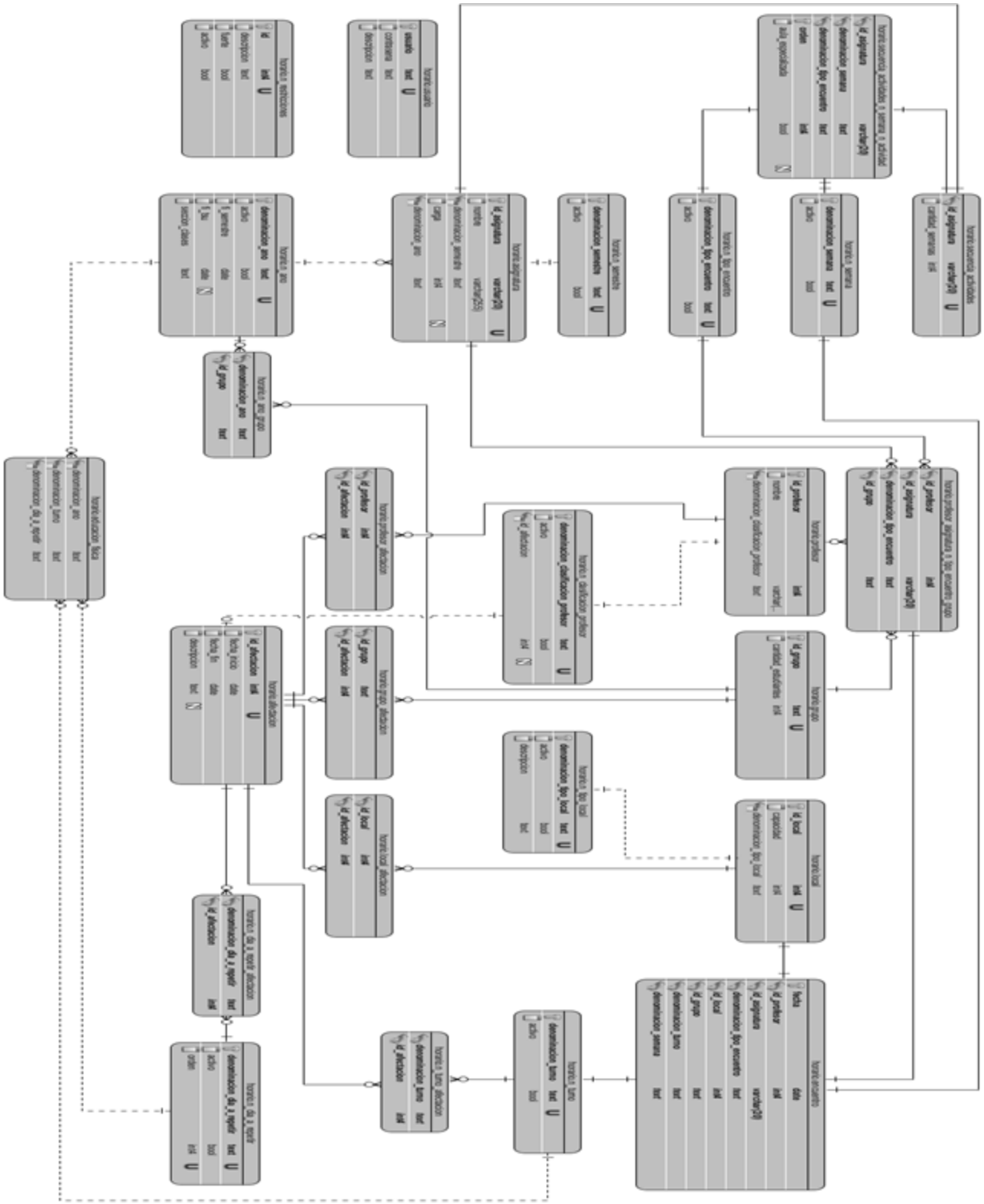

Figura 2. Modelo de datos (Elaboración propia)

2.7 Patrones de diseño

Según (Larman, 1999), el patrón es una pareja de problema-solución con un nombre y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas que codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades.

A continuación se muestran los patrones generales de software para asignar responsabilidades, conocidos por su acrónimo en inglés como *General Responsibility Asignment Software Patterns* (GRASP) utilizados en el diseño.

Experto

Los objetos realizan las funciones relacionadas con la información que poseen, el comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clase sencillas y más cohesivas que son más fáciles de comprender y de mantener (Visconti, y otros, 2006). Se evidencia el uso del patrón experto al instanciar la clase *ObjetiveFunction*, responsable de calcular la evaluación al comprobar el cumplimiento de las restricciones débiles como se muestra en la siguiente figura.

```
public int evaluar(ArrayList<ArrayList<Encuentro>> semana) {
    this.semana = semana;
    findStatusData();
    setCantEstudiantes();
    try {
        return (checkCapacidad() +
                checkDejarDiaLuegoDeConferencia() +
                checkEducacionFisicaPrecedaSuceda() +
                checkGruposEnMismoLocal() +
                checkHorarioCompacto() +
                checkProfesoresMasDeUnGrupoMismaAsignatura() +
                checkLabsSesionCorresponda());
    } catch (SQLException ex) {
        Logger.getLogger(ObjetiveFunction.class.getName()).log(Level.SEVERE, null, ex);
    }
    return 0;
}
```

Figura 3. Uso del patrón experto (Elaboración propia)

Creador

El patrón creador sugiere quién debe ser el responsable de la creación de nuevos objetos o clases. Su propósito fundamental es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Un diseño bien asignado puede soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reusabilidad (Larman, 1999). Se evidencia el uso de este patrón en la clase *TabuSolver*, encargada de instanciar todos los componentes de la búsqueda tabú como se muestra en la siguiente figura.

```
public TabuSolver() {
    this.FO = new ObjetiveFunction();
    this.tabuList = new TabuList(5);
    this.mv = new MoveManager();
    this.answers = new ArrayList<>();
    this.solution = new Solution();
}

public Solution resolver() {
    tabuList.add(new TabuElement(solution, 2));
    int compare = FO.evaluar(SolutionContainer.semana);
    mv.generate(SolutionContainer.semana, 8);
    // ...
```

Figura 4. Uso del patrón Creador (Elaboración propia)

Controlador

Según lo planteado en (Visconti, y otros, 2006) el controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además que el método de su operación, los objetos externos de conexión y la capa de presentación no deberían tener la responsabilidad de llevar a cabo los eventos del sistema (Larman, 1999). Se evidencia el uso del patrón controlador en la clase *App*, encargada de ejecutar el tipo de optimización que se desea realizar especificando si es de forma completa o por intervalos o si se desea contar el número de restricciones débiles como se muestra en la siguiente figura.

```
public static void main(String[] args) {
    if (args.length > 0) {
        switch (args[0]) {
            case "solve":
                Main.main(args);
            case "count":
                Count.main(args);
            case "interval":
                Test.main(args);
            default:
                System.out.println("|Only 'solve', 'count' or 'interval' parameters are allowed!|");
        }
    }
}
```

Figura 5. Uso del patrón Controlador (Elaboración propia)

Bajo acoplamiento

El bajo acoplamiento es un principio que se debe tener en cuenta durante las decisiones de diseño. Es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño. Estimula la asignación de una responsabilidad, de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento. Soporta el diseño de clases más independientes para reducir el impacto de los cambios y hacerlas más reutilizables (Larman, 1999). Se evidencia el uso del patrón bajo acoplamiento porque se desacopla el conocimiento de la clase *TabuElement* con la clase *Encuentro* como se muestra en la siguiente figura.

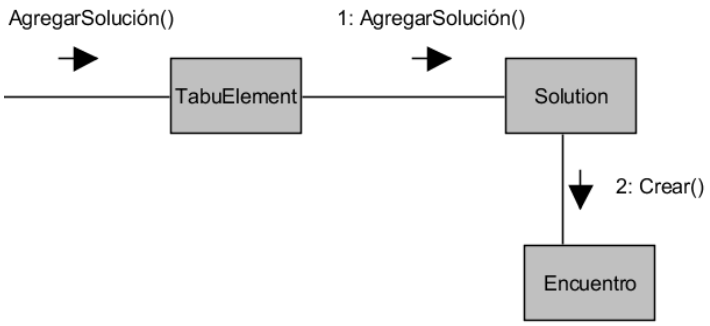

Figura 6. Uso del patrón Bajo acoplamiento (Elaboración propia)

Alta cohesión

La alta cohesión es una meta que ha de tenerse en cuenta en cada momento en todas las decisiones de diseño. Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño. Una clase de alta cohesión posee un número relativamente pequeño de funciones, con una importante funcionalidad relacionada y poco trabajo que hacer. Colabora con otros objetos para compartir el esfuerzo si la tarea es grande (Larman, 1999).

Este patrón se evidencia cuando se designan clases con funcionalidades específicas para la comprobación de las restricciones fuertes y débiles y la generación de vecindades que colaboran en la clase *TabuSolver*, encargada de devolver la mejor solución encontrada como se muestra en la siguiente figura.

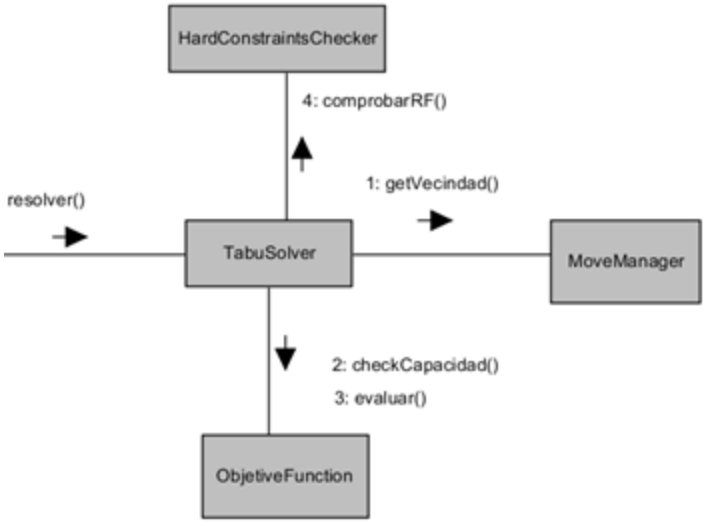

Figura 7. Uso del patrón Alta cohesión (Elaboración propia)

2.8 Fase planificación de la entrega

Se plantea en (Letelier Torres, y otros, 2003) que en esta fase el cliente establece la prioridad de cada HU y correspondientemente, los programadores estiman el esfuerzo necesario para su implementación. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses.

No	Historia de usuario	Estimación (semana)
1	Gestionar bloques	2
2	Gestionar asignaturas optativas	2
3	Gestionar asignaturas electivas	2
4	Optimizar restricciones débiles	3

Tabla 8. Plan de duración de las iteraciones

2.9 Fase iteraciones

Según lo planteado por (Beck, 1999) esta fase contiene las iteraciones que debe ejecutar el sistema antes de su entrega, las cuales no durarán más de tres semanas. En la primera iteración se intenta establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Al final de la última iteración el sistema estará listo para entrar en producción.

2.9.1 Plan de iteraciones

En este plan se especifica las HU que serán implementadas en cada iteración del sistema. Se concretaron dos iteraciones para la realización del sistema:

- Iteración 1: Tiene como objetivo la implementación de las HU: Gestionar bloques, gestionar asignaturas optativas y gestionar asignaturas electivas que sirven como base para la generación del horario.
- Iteración 2: Tiene como objetivo la implementación de la cuarta historia de usuario que es la encargada de optimizar el cumplimiento de las restricciones débiles.

2.9.2 Plan de entregas

En la siguiente tabla se muestra la distribución por semanas y las fechas de entrega de las funcionalidades descritas en las HU.

Iteración	Historia de usuario	Estimación (semana)	Fecha inicio - fin
1	Gestionar bloques	2	20/02/2015 – 05/03/2015
1	Gestionar asignaturas optativas	2	05/03/2015 – 25/03/2015
1	Gestionar asignaturas electivas	2	25/03/2015– 10/04/2015
2	Optimizar restricciones débiles	3	10/04/2015 – 05/05/2015

Tabla 9. Plan de entregas

2.10 Arquitectura

El desarrollo de la solución se basa en el patrón arquitectónico Modelo-Vista-Controlador (MVC), uno de los más utilizados en las aplicaciones web pues divide el sistema en tres capas de abstracción, propiciando el aislamiento entre el modelado del dominio, el control de los eventos del sistema y las interfaces del sistema.

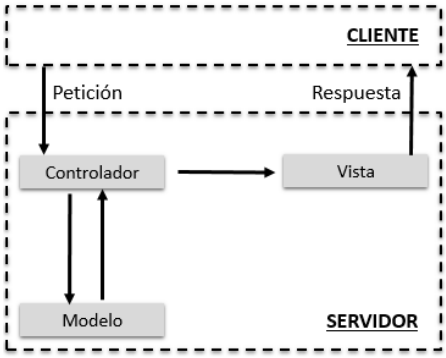

Figura 8. Funcionamiento del patrón arquitectónico MVC (Elaboración propia)

A continuación se muestra la arquitectura propuesta para el Sistema para la generación del horario docente de la Facultad de Ingeniería Industrial de la CUJAE basada en MVC.

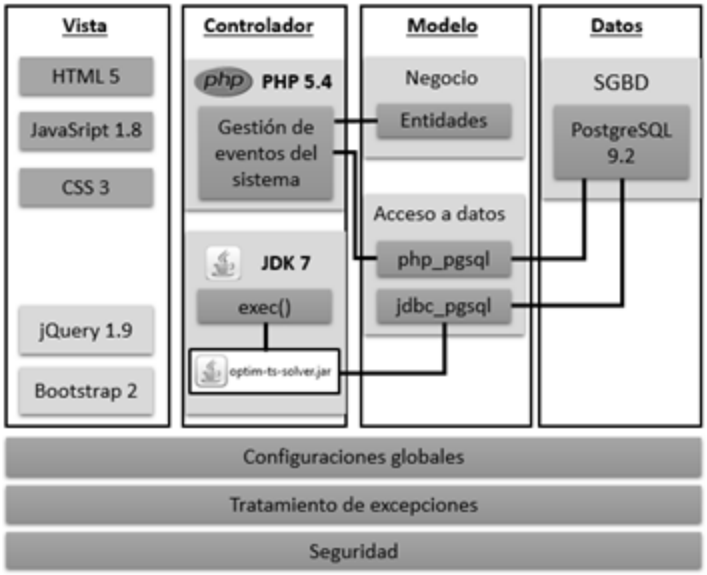

Figura 9. Arquitectura del Sistema para la generación del horario docente de la Facultad de Ingeniería Industrial de la CUJAE (Elaboración propia)

- **Modelo:** Define las entidades presentes en el negocio y el acceso a la base de datos a través de la extensión php_pgsql y la biblioteca jdbc_pgsql.
- **Vista:** Presenta todas las interfaces del sistema diseñadas con los marcos de trabajo Bootstrap 2 y jQuery en conjunto con los archivos CSS, JavaScript y HTML con las cuales el usuario gestiona visualmente los datos referentes a los profesores, locales, asignaturas, grupos, bloques, su relación, su configuración, la generación del horario y el proceso de optimización.
- **Controlador:** Facilita y controla la comunicación entre las vistas y los modelos mediante la gestión de eventos del sistema. Integra la biblioteca para la optimización del horario *optim-ts-solver.jar* a través de la función de PHP *exec()* que permite la ejecución de un programa externo.

Se definen las configuraciones globales para el acceso a la base de datos correspondientes a la dirección del servidor, usuario y contraseña. Se realiza el tratamiento de excepciones y se definen los mecanismos de seguridad en el sistema (encriptación de contraseñas y autenticación basada en roles).

2.11 Estándar de codificación

Con el objetivo de aportarle mayor facilidad de lectura y modificación se hace uso de un conjunto de normas comunes a los desarrolladores.

Nombres de las funciones: Contienen caracteres alfanuméricos, deben comenzar con minúsculas y cuando tienen más de una palabra la primera letra de estas debe capitalizarse.

Nombres de las variables: Deben ser descriptivos y concisos, deben comenzar con minúsculas y en caso de contener más de una palabra esta debe ser separada por guión bajo (_).

Llamadas a funciones: Se llamarán sin utilizar espacios entre el nombre de la función, el paréntesis de apertura y el primer parámetro, tampoco entre el último parámetro, el paréntesis de cierre y el punto y coma; los espacios se usarán entre las comas y cada uno de los parámetros.

```
public static ArrayList getData(int dataType) {
    // ...
```

Figura 10. Ejemplo de estándar de codificación "llamadas a funciones" (Elaboración propia)

2.12 Descripción de la solución

La propuesta de solución incluye nuevas funcionalidades que se integran al sistema existente para la generación del horario de la Facultad de Ingeniería Industrial de la CUJAE un módulo para la gestión de asignaturas optativas y electivas, manteniendo las líneas de diseño trazadas en la solución anterior, así como la posibilidad de optimizar las restricciones débiles para la confección de un mejor horario.

2.12.1 Planificación de asignaturas optativas y electivas

Las asignaturas optativas son materias relacionadas con el perfil de la carrera y se imparten a partir el segundo semestre de tercer año.

Las asignaturas electivas son materias destinadas a enriquecer la formación del estudiante porque no se relacionan con el perfil de estudio de la Facultad y se imparten a los estudiantes en el segundo año de la carrera.

Asignaturas de 1er año (9)

Hay EF planificada | Adicionar

Identificador	Nombre	Tipo	Requiere profesor	Requiere local	Opciones
IF	Introducción a la Informática	Regular	✔	✔	
II	Introducción a la Ingeniería	Regular	✔	✔	
H	Historia de Cuba	Regular	✔	✔	
FyS	Filosofía y Sociedad	Regular	✔	✔	
AL	Geometría Analítica y Álgebra Lineal	Regular	✔	✔	
OTGE	Organización del trabajo en el grupo empresarial	Optativa	✔	✔	
PMPN	Patrones para el modelado de procesos de negocio	Optativa	✔	✔	
M1	Matemática 1	Regular	✔	✔	
DB	Dibujo Básico	Regular	✔	✔	

Figura 11. Ejemplo del listado de asignaturas (Elaboración propia)

Adición de asignaturas optativas y electivas

En el menú lateral al seleccionar la opción Asignaturas y el año correspondiente se listan las asignaturas registradas para ese año y la opción de adicionar permite la inserción de una nueva. Cuando se adiciona

una nueva asignatura el usuario debe indicar si es regular, optativa o electiva, además de su identificador, nombre completo y si requiere profesor y local de la Facultad.

Figura 12. Ejemplo de adición de asignatura (Elaboración propia)

Creación de bloques para la distribución de asignaturas optativas

Al seleccionar la opción Bloques de las configuraciones generales se listarán los bloques definidos. A través de la opción Adicionar se pueden crear nuevos bloques definiendo su identificador y el rango de semanas que abarca. El número máximo de semanas de un bloque se corresponde con el máximo de semanas del año en el que se debe utilizar**.**

Figura 13. Ejemplo de adición de bloque (Elaboración propia)

Creación de la secuencia de actividades de las asignaturas optativas y electivas

En el menú lateral al seleccionar la opción Asignaturas y el año correspondiente se listan las asignaturas registradas para ese año. Para cada asignatura se muestra la opción de adicionar o modificar su secuencia de actividades, en el caso de las asignaturas electivas u optativas además se debe indicar el bloque en que se debe impartir, se debe especificar la semana en que comienza, la cantidad de semanas de duración y el día y turno en que se impartirá cada frecuencia de clases. Además se deben registrar las actividades que requieren aula especializada y la secuencia de actividades.

Figura 14. Ejemplo de creación de secuencia de actividades (Elaboración propia)

2.12.2 Optimización del horario docente

El algoritmo para optimizar el cumplimiento de las restricciones débiles se aplica en la solución mediante los pasos siguientes:

Paso 1: Se toma como configuración inicial una semana para un grupo en específico, dicha configuración satisface las restricciones fuertes definidas. Antes de realizar las siguientes operaciones se comprobará si esta solución satisface las restricciones débiles para evitar iteraciones innecesarias y disminuir el tiempo de ejecución del algoritmo.

Paso 2: A partir de la configuración inicial se genera un espacio de 10 nuevas configuraciones denominado vecindad mediante el intercambio de dos turnos de forma aleatoria. Para que cada movimiento se considere válido se tienen en cuenta los siguientes aspectos:

- De los dos encuentros a intercambiar seleccionados aleatoriamente debe haber al menos uno que no sea vacío.
- Los encuentros de las asignaturas Dibujo básico o Dibujo aplicado no se intercambian porque siempre se deben impartir en el primer turno de la mañana.
- Los encuentros del tipo Laboratorio de Física no se intercambian porque los locales destinados a esta actividad no pertenecen a la Facultad y se utilizan en los días y turnos específicos durante el semestre.

- Las asignaturas optativas y electivas no se intercambian pues se imparten en días y turnos específicos para todos los grupos del año correspondiente.
- Se debe comprobar la validez de la secuencia de actividades después de realizar el intercambio.
- Las nuevas configuraciones deben cumplir las restricciones fuertes definidas.

Paso 3: La solución generadora se establece como tabú para evitar ciclos durante la búsqueda, pues los movimientos aleatorios generados en la vecindad pueden ser iguales a la configuración inicial.

Se itera por cada configuración generada y se evalúa en la función objetivo para comprobar el cumplimiento o no de las restricciones débiles. La evaluación se define como un número entero que representa la sumatoria del resultado de cada restricción débil.

Las restricciones débiles tienen un nivel de prioridad en la organización. Se definen como un número entero que puede obtener un valor de uno a tres donde un número mayor representa una restricción más importante. Los valores asociados se especifican en la siguiente tabla.

Restricción débil	**Evaluación**
Los grupos deben tener clases en un mismo local, el cambio de local debe ser mínimo.	1
El horario debe ser compacto.	3
A los profesores que imparten clase a más de un grupo, de una misma asignatura, se les debe planificar clases en turnos consecutivos un mismo día.	3
Las clases de laboratorio deben planificarse en la sección de clases que corresponde.	1
Después de una clase de tipo “conferencia” se debe dejar, al menos, un día de diferencia con relación al próximo turno de clase.	2
Un grupo no puede tener clases en un local que no tenga la capacidad que requiere.	3
La asignatura Educación física no debe tener clases que la preceda o la suceda.	2

Tabla 10. Valores asociados a las restricciones débiles

Paso 4: Se comprueba si los resultados devueltos por la función objetivo son mejores que la evaluación actual, de ser afirmativo se agregan a una colección de respuestas (conjunto de posibles soluciones con las mejores evaluaciones en la función objetivo). Si no existe una solución mejor se mantiene la anterior.

Paso 5: Se ordena la colección de respuestas por la evaluación obtenida en la función objetivo y se selecciona la mejor que representa la de menor evaluación debido a que se está minimizando el incumplimiento de las restricciones débiles.

Paso 6: Se elimina de la tabla encuentro la solución anterior y se inserta la nueva.

Paso 7: Se repite el proceso hasta completar todos los grupos registrados en el sistema.

El proceso de optimización se realiza de forma aleatoria y es posible la obtención de mejores resultados si se aplica reiteradamente. Por esta razón se brinda la posibilidad de realizar nuevamente la optimización completa o por intervalos de grupos y semanas.

Para la optimización por intervalos el usuario tiene la posibilidad de introducir rangos de grupos y semanas que se podrán combinar como expresiones regulares:

- Definición por rangos para grupos: Grupo de inicio – grupo final (I11-I14, se seleccionan los grupos I11, I12, I13 e I14).
- Definición por comas para grupos: Grupo1, grupo2, grupoN (I13, I22, I24 se seleccionan los grupos I13, I22, I24).
- Definición por rangos para semanas: Semana de inicio – semana final (01-05, se seleccionan las semanas 01, 02, 03, 04 y 05).
- Definición por comas para semanas: Semana1, semana2, semanaN (01, 03, 05 se seleccionan las semanas 01, 03, 05).

2.13 Conclusiones parciales

A partir de las restricciones y las HU definidas se desarrolló la propuesta de solución para la planificación de asignaturas electivas y optativas, así como la optimización del horario factible generado por el Sistema de generación del horario de la Facultad de Ingeniería Industrial de la CUJAE.

La arquitectura basada en el patrón arquitectónico Modelo-Vista-Controlador y otros patrones de diseño para la asignación de responsabilidades, permitió el empleo de buenas prácticas de programación.

La elaboración del plan de entrega y la definición de las iteraciones en que se debe implementar cada HU, permitió organizar el trabajo precisando qué implementar en cada instante.

La definición y caracterización de las funcionalidades en las HU guió la creación de un módulo para la gestión de asignaturas optativas y electivas.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN

3.1 Introducción

En este capítulo se realizan pruebas a la solución desarrollada con el objetivo de evaluar los resultados obtenidos en el diseño e implementación del sistema. Se analizan las diferentes pruebas de software que fueron aplicadas para medir la calidad del sistema, así como los resultados de cada una de ellas.

3.2 Verificación del diseño

La ejecución de las pruebas de diseño se basa en los métodos de Tamaño operacional de clases y Relación entre clases los cuales se especifican a continuación con sus respectivos criterios de evaluación y los resultados arrojados luego de su aplicación.

Resultado de las pruebas TOC

Atributo de calidad	Categoría	Criterio
Responsabilidad	Baja	<= Promedio
	Media	Entre Promedio y 2 * Promedio
	Alta	> 2 * Promedio
Complejidad de implementación	Baja	<= Promedio
	Media	Entre Promedio y 2 * Promedio
	Alta	> 2 * Promedio
Reutilización	Baja	> 2 *Promedio
	Media	Entre Promedio y 2 * Promedio
	Alta	<= Promedio

Tabla 11. Rango de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC

A continuación se muestran las clases del sistema evaluadas en los atributos de calidad mencionados.

Clase	Cantidad de procedimientos	Responsabilidad	Complejidad de implementación	Reutilización
ConstraintsCounter	3	Baja	Baja	Alta
HardConstraintsChecker	7	Baja	Baja	Alta

MoveManager	9	Media	Media	Media
ObjetiveFunction	11	Media	Media	Media
Solution	7	Baja	Baja	Alta
SolutionContainer	0	Baja	Baja	Alta
TabuElement	6	Baja	Baja	Alta
TabuList	7	Baja	Baja	Alta
TabuSolver	1	Baja	Baja	Alta
Encuentro	20	Alta	Alta	Baja
Queries	24	Alta	Alta	Baja
SingleConection	8	Media	Media	Media
PrintSchedule	6	Baja	Baja	Alta
Timer	1	Baja	Baja	Alta

Tabla 12. Evaluación de las clases del sistema mediante la métrica TOC

Responsabilidad

Responsabilidad	Cantidad de clases	Promedio
Baja	9	64,2
Media	3	21,4
Alta	2	14,2

Tabla 13. Pruebas TOC del atributo de calidad “Responsabilidad”

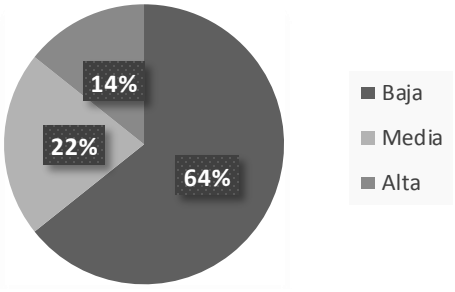

Figura 15. Pruebas TOC del atributo de calidad “Responsabilidad” (Elaboración propia)

Al analizar el atributo de calidad se observa la existencia de un mayor promedio de clases con baja responsabilidad según lo planteado por la métrica de acuerdo a la cantidad de procedimientos de las clases.

Complejidad

Complejidad	Cantidad de clases	Promedio
Baja	9	64,2

Media	3	21,4
Alta	2	14,2

Tabla 14.Pruebas TOC del atributo de calidad “Complejidad”

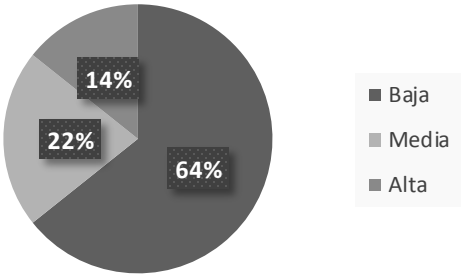

Figura 16. Pruebas TOC del atributo de calidad “Complejidad” (Elaboración propia)

Al analizar el atributo de calidad se observa la existencia de un mayor promedio de clases con baja complejidad según lo planteado por la métrica de acuerdo a la cantidad de procedimientos de las clases.

Reutilización

Reutilización	Cantidad de clases	Promedio
Alta	9	64,2
Media	3	21,4
Baja	2	14,2

Tabla 15. Pruebas TOC del atributo de calidad “Reutilización”

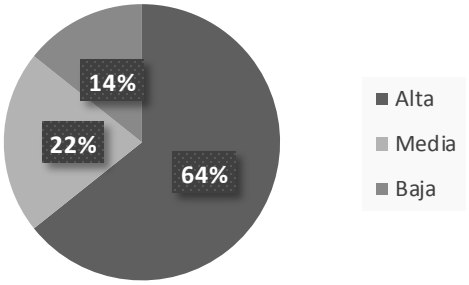

Figura 17. Pruebas TOC del atributo de calidad “Reutilización” (Elaboración propia)

Al analizar el atributo de calidad se observa la existencia de un mayor promedio de clases con alta reutilización según lo planteado por la métrica de acuerdo a la cantidad de procedimientos de las clases.

Como resultado de la aplicación de la métrica TOC se evidencia que las clases del sistema poseen una baja responsabilidad, baja complejidad de implementación y alta reutilización por lo que el diseño de las clases en cuanto a cantidad de funcionalidades por cada una es satisfactorio.

Resultado de las pruebas RC

La ejecución de las pruebas de diseño se basó en los métodos que se mencionan a continuación con sus respectivos criterios de evaluación.

Atributo de calidad	Categoría	Criterio
Acoplamiento	Ninguno	0
	Baja	1
	Media	2
	Alta	>2
Complejidad de mantenimiento	Baja	<= Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	> 2* Promedio
Reutilización	Baja	>2* Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	<= Promedio
Cantidad de pruebas	Baja	<= Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	> 2* Promedio

Tabla 16. Rangos de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica RC

A continuación se muestran las clases del sistema evaluadas en los atributos de calidad mencionados.

Clase	Cantidad de relaciones de uso	Acoplamiento	Complejidad de mantenimiento	Reutilización	Cantidad de pruebas
HardConstraintsChecker	1	Bajo	Baja	Alta	Baja
MoveManager	1	Bajo	Baja	Alta	Baja
ObjetiveFunction	1	Bajo	Baja	Alta	Baja
Solution	1	Bajo	Baja	Alta	Baja
TabuElement	1	Bajo	Baja	Alta	Baja
TabuList	1	Bajo	Baja	Alta	Baja
TabuSolver	4	Alto	Alta	Baja	Alta

Tabla 17. Evaluación de las clases del sistema mediante la métrica RC

Acoplamiento

Acoplamiento	Cantidad de clases	Promedio
Ninguno	0	0
Bajo	6	85,7
Medio	0	0
Alto	1	14,3

Tabla 18. Pruebas RC del atributo de calidad “Acoplamiento”

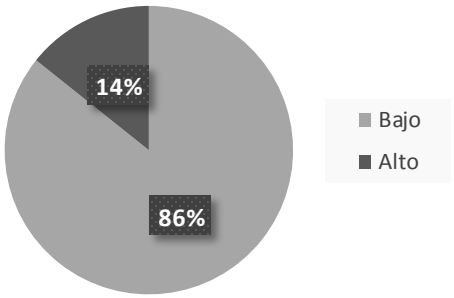

Figura 18. Pruebas RC del atributo de calidad “Acoplamiento” (Elaboración propia)

Al analizar el atributo de calidad se observa la existencia de un mayor promedio de clases con bajo acoplamiento según lo planteado por la métrica de acuerdo a la cantidad de relaciones entre clases.

Complejidad de mantenimiento

Complejidad de Mantenimiento	Cantidad de clases	Promedio
Baja	6	85,7
Media	0	0
Alta	1	14,2

Tabla 19. Pruebas RC del atributo de calidad “Complejidad de mantenimiento”

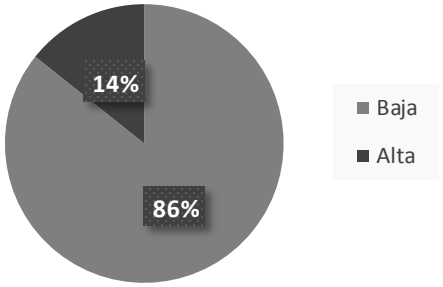

Figura 19. Pruebas RC del atributo de calidad “Complejidad de mantenimiento” (Elaboración propia)

Al analizar el atributo de calidad se observa la existencia de un mayor promedio de clases con baja complejidad de mantenimiento según lo planteado por la métrica de acuerdo a la cantidad de relaciones entre clases.

Cantidad de pruebas

Cantidad de Pruebas	Cantidad de clases	Promedio

Baja	6	85,7
Media	0	0
Alta	1	14,2

Tabla 20. Pruebas RC del atributo de calidad “Cantidad de pruebas”

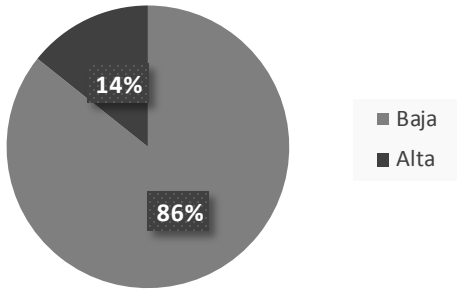

Figura 20. Pruebas RC del atributo de calidad “Cantidad de pruebas” (Elaboración propia)

Al analizar el atributo de calidad se observa la existencia de un mayor promedio de clases con baja cantidad de pruebas a realizar según lo planteado por la métrica de acuerdo a la cantidad de relaciones entre clases.

Reutilización

Reutilización	Cantidad de clases	Promedio
Baja	1	14,2
Media	0	0
Alta	6	85,7

Tabla 21. Pruebas RC del atributo de calidad “Reutilización”

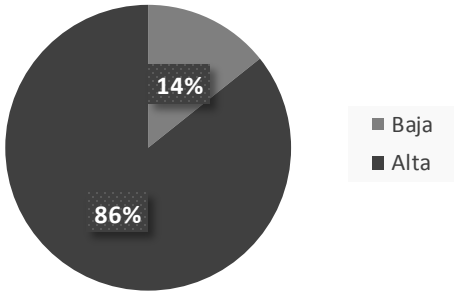

Figura 21. Pruebas RC del atributo de calidad “Reutilización” (Elaboración propia)

Al analizar el atributo de calidad se observa la existencia de un mayor promedio de clases con alta reutilización según lo planteado por la métrica de acuerdo a la cantidad de relaciones entre clases.

Los resultados obtenidos en la aplicación de la métrica RC evidencian que las clases del sistema poseen una baja complejidad de mantenimiento, bajo acoplamiento, baja cantidad de pruebas y alta reutilización por lo que de forma general se valoran como buenos los resultados.

3.3 Pruebas de caja blanca

Resultados de la prueba del camino básico

A continuación se muestra el resultado de aplicar la prueba de caja blanca mediante la técnica del camino básico aplicado al método *getAllMoves()* que devuelve una lista de los movimientos generados a evaluar en la función objetivo como se muestra en la siguiente figura.

```
public void getAllMoves() {
    for (ArrayList<ArrayList<Encuentro>> a : all) {
        // ...
        if (encX != null && encY != null && checkRestriccionesDeCambio(encX)) {
            if ("mañana".matches(sesion)) {
                // ...
            } else if ("tarde".matches(sesion)) {
                // ...
            }
        } else if (encX != null && encY == null && checkRestriccionesDeCambio(encX)) {
            if ("mañana".matches(sesion)) {
                // ...
            } else if ("tarde".matches(sesion)) {
                // ...
            }
        } else if (encY != null && encX == null && checkRestriccionesDeCambio(encY)) {
            if ("mañana".matches(sesion)) {
                // ...
            } else if ("tarde".matches(sesion)) {
                // ...
            }
        }

        // ...
    }
}
```

Figura 22. Método encargado de la generación de vecindades (Elaboración propia)

A continuación se muestra el grafo dirigido de flujo de la funcionalidad *getAllMoves()*.

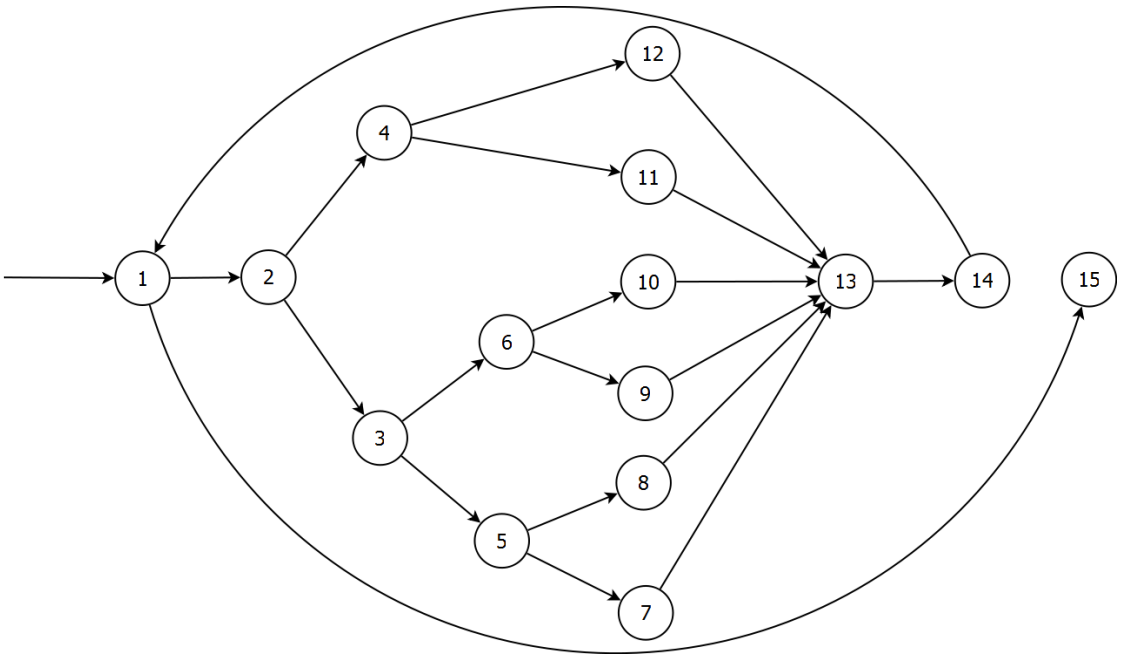

Figura 23. Grafo de flujo (Elaboración propia)

Tomando como referencia el grafo dirigido del flujo se calculó la complejidad ciclomática empleando las tres variantes propuestas por (Pressman, 2005).

- Existen 7 regiones.
- V (G) = 20 aristas - 15 nodos + 2 = 7
- V (G) = 6 nodos predicado + 1 = 7

La complejidad ciclomática es igual a siete, esto significa que existen siete posibles caminos linealmente independientes y representa el mínimo número de casos de prueba para el método tratado, a continuación se muestran los caminos existentes.

Número	Caminos
1	1-15
2	1-2-3-5-7-13-14-1-15
3	1-2-3-5-8-13-14-1-15
4	1-2-3-6-9-13-14-1-15
5	1-2-3-6-10-13-14-1-15
6	1-2-4-11-13-14-1-15
7	1-2-4-12-13-14-1-15

Tabla 22. Caminos por donde el flujo puede circular

El próximo paso es ejecutar los casos de pruebas para cada camino y se compara con los resultados esperados, verificando que las instrucciones se ejecuten por lo menos una vez. A continuación se muestra el caso de prueba para el primer camino básico 1, en el Anexo 2 se muestran los restantes casos de prueba.

Caso de prueba para el camino básico 1	
Descripción	Se comprueba que se han recorrido las 10 vecindades generadas
Condiciones de ejecución	Que no se haya alcanzado el final de la lista de vecindades.
Entrada	ArrayList<ArrayList<Encuentro>> semana
Resultado esperado	Que continúe con la ejecución del resto del código.

Tabla 23. Caso de prueba para el camino básico 1

Al concluir la ejecución de los casos de pruebas se pudo verificar que todas las instrucciones se ejecutaban al menos una vez.

3.4 Pruebas de caja negra

Pruebas de aceptación

Las pruebas de aceptación aplicadas a la solución se realizan utilizando el documento de diseño de casos de pruebas, en el que se especifican los distintos escenarios que se corresponden a cada una de las historias de usuario. Para la realización de las pruebas se utilizaron datos válidos e inválidos, eligiendo los valores de entrada, con el objetivo de abarcar la mayor cantidad posible de combinaciones, teniendo en cuenta que la cantidad de casos de prueba fuera muy elevada.

Las pruebas de aceptación se ejecutaron por un especialista del equipo de calidad del Centro de Gobierno Electrónico en tres iteraciones, en la primera se encontraron 10 no conformidades (nueve significativas y una no significativa), en la segunda seis no conformidades (todas significativas) que fueron solucionadas en su totalidad y en la tercera iteración no se detectaron inconformidades como se muestra en la siguiente figura. El acta de liberación emitida se encuentra en el Anexo 3.

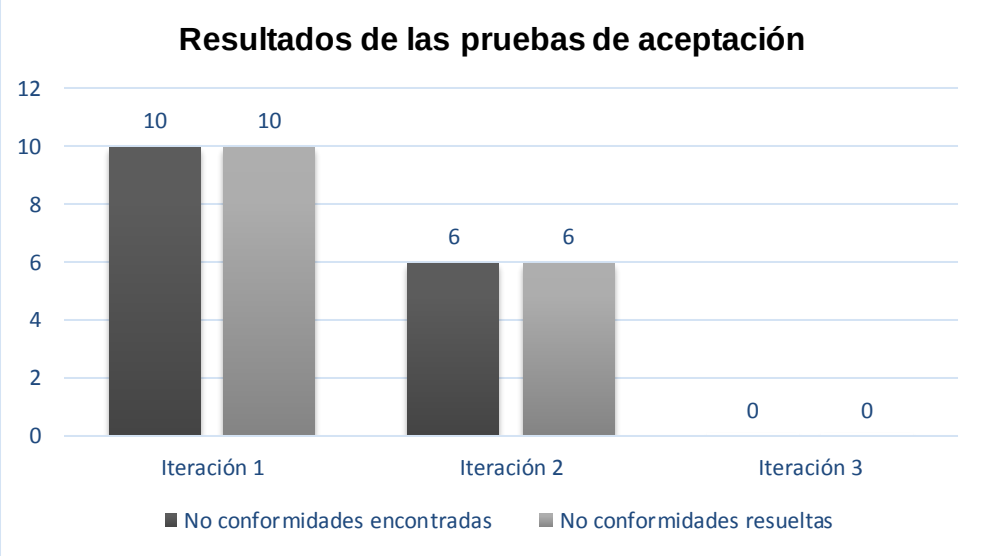

Figura 24. Resultado de las pruebas de aceptación (Elaboración propia)

Resultado del cumplimiento de las restricciones débiles

Para realizar la comparación de la calidad entre el horario factible inicial y el horario optimizado se utilizan los datos del primer semestre del curso docente 2013-2014 de la Facultad de Ingeniería Industrial de la CUJAE, teniendo en cuenta 128 profesores, 32 grupos, 23 locales y 36 asignaturas.

Restricciones débiles	Original	Optimizada	% de mejora
Un grupo no puede tener clases en un local que no tenga la capacidad que requiere.	2	2	0
El horario debe ser compacto.	186	32	82,8
Después de una clase de tipo conferencia se debe dejar, al menos, un día de diferencia con relación al próximo turno de clase de la asignatura.	470	452	3,8
Los grupos deben tener clases en un mismo local, el cambio de local debe ser mínimo.	167	78	53,3
La asignatura Educación física no debe tener clase que la preceda o la suceda.	4	4	0
A los profesores que imparten clase a más de un grupo, de una misma asignatura, se les debe planificar clases en turnos consecutivos un mismo día.	199	144	27,6
Las clases de laboratorio deben planificarse en la sección de clases que corresponde.	0	0	0
Total	**1028**	**712**	**30,7**

Tabla 24. Comparativa del cumplimiento de restricciones débiles

Al realizar una iteración del proceso de optimización se puede apreciar que para las restricciones débiles de mayor importancia se obtiene una mejora de entre un 27% a un 83%. Este proceso se puede repetir para la obtención de mejores resultados, de forma completa o por intervalos de grupos y semanas en dependencia de la decisión del usuario.

De manera general el promedio de mejora para la primera iteración, en comparación con la solución inicial factible, es de un 31% y satisface todas las restricciones fuertes establecidas, por lo que se considera un horario de mayor calidad porque satisface en mayor medida las preferencias de profesores y estudiantes.

3.5 Conclusiones parciales

La aplicación de las métricas TOC y RC permitió la comprobación del adecuado diseño de las clases.

La realización de las pruebas de caja blanca y caja negra facilitó la identificación de errores en la solución para su rápida corrección.

Se pudo comprobar que la solución propuesta permite disminuir el número de violaciones de las restricciones débiles del horario factible inicial del Sistema para la generación del horario docente de la Facultad de Ingeniería Industrial de la CUJAE, además incluye la planificación de las asignaturas lectivas, optativas y electivas, por lo que el horario generado se considera completo.

CONCLUSIONES GENERALES

La colaboración continua entre los desarrolladores y el cliente permitió el diseño y la implementación del módulo de planificación de las asignaturas optativas del Sistema para la generación del horario docente de la Facultad de Ingeniería Industrial de la CUJAE.

La elaboración del marco teórico de la investigación permitió determinar las herramientas, tecnologías y lenguajes de programación a utilizar, así como la metaheurística más adecuada para la optimización del horario docente.

La selección de la metaheurística búsqueda tabú y el análisis de las restricciones definidas garantizaron la implementación del algoritmo de optimización del horario docente.

Las métricas y pruebas de software aplicadas a la propuesta de solución permitieron valorar de forma satisfactoria la efectividad de la solución propuesta.

Con la implementación del módulo de planificación de las asignaturas electivas y optativas se pudo obtener un horario completo que cumple con las restricciones fuertes establecidas, cuya optimización permitió disminuir el número de violaciones de las restricciones débiles en un 31%, en comparación con la solución inicial.

RECOMENDACIONES

Para futuras versiones del Sistema para la generación del horario docente de la Facultad de Ingeniería Industrial de la CUJAE se recomienda:

- Aplicar nuevas técnicas de generación de vecindades tales como “Intercambio de locales” e “Intercambio de eventos que causan restricciones”, que permitan diversificar el espacio de soluciones.
- Definir nuevos criterios de aspiración que definan el tiempo en el que una posible solución puede salir del estado tabú.

BIBLIOGRAFÍA

Arora, Sanjeev y Barak, Boaz. 2009. *Computational Complexity: A Modern Approach.* s.l. : Cambridge University Press, 2009. ISBN 978-0-521-42426-4.

Baquero, Franco y Fredy, John. 2008. Catálogo de publicaciones en línea. [En línea] 2008. [Citado el: 2015 de 02 de 05.] http://rcientificas.uninorte.edu.co/index.php/ingenieria/article/view/1752/1138.

Barrera, MSc. David Silva. 2011. Sistema de Planificación y publicación de horarios docentes UCI (Horr). *Manual de usuario.* Universidad de las Ciencias Informáticas, Habana, Cuba : s.n., 2011.

Barrientos, Pablo Andrés. 2014. *Enfoque para pruebas de unidad basado en la generación aleatoria de objetos.* La Plata : Universidad Nacional de La Plata, 2014.

Beck, Kent. 1999. *Extreme Programming Explained.* s.l. : Pearson Education, 1999. ISBN: 0201616416.

Beck, Kent, y otros. 2001. Manifiesto por el Desarrollo Ágil de Software. *Manifesto for Agile Software Development.* [En línea] 2001. [Citado el: 25 de 03 de 2015.] http://www.agilemanifesto.org/iso/es/.

Borjas Giraldo, Giancarlo. 2013. *Análisis, diseño e implementación de un sistema de información para la administración de horarios y rutas en empresas de transporte público.* Lima : Ponifica Universidad Católica de Perú, 2013. 20052267.

Burke, Edmund Kieran, y otros. 1997. *Automated university timetabling: The state of the art.* s.l. : The Computer Journal, 1997. pp. 565–571.

Chávez Bosquez, Oscar A., De los Santos Torres, Guillermo y Gómez Ramos, José Luis. 2005. *Búsqueda tabú aplicada a un problema NP-Completo: Generación de horarios en la DAIS.* México : División Académica de Informática y Sistemas, Universidad Juárez Autónoma de Tabasco, 2005.

Chávez Bosquez, Oscar, Pozos Parra, Pilar y Gómez Ramos, José Luis. 2014. *Búsqueda Tabú con criterio de aspiración probabilístico aplciada a la generación de horarios escolares.* Tabasco : CIMPA – UCR, 2014. 1409-2433.

Cobas Friman, Katia. 2012. *Algoritmos genéticos aplicados a problemas de planificación.* Manzanillo : Universidad de las Ciencias Informáticas Facultad Regional Granma, 2012.

Computing-Society. 2015. *The Nature of Mathematical Programming.* s.l. : Mathematical Programming Glossary, 2015.

Curak, Ivan. 2007. Open Course Timetabler | SourceForge.net. *Open Course Timetabler | SourceForge.net.* [En línea] 2007. [Citado el: 6 de Marzo de 2015.] http://sourceforge.net/projects/openctt/.

De la Cruz H., Jair J., y otros. 2003. Revistas Científicas, Universidad del Norte. *Análisis comparativo de las aproximaciones heurísticas Ant-Q, recocido simulado y búsqueda tabú en la solución del problema del agente viajero.* [En línea] 2003. [Citado el: 7 de Enero de 2015.] http://rcientificas.uninorte.edu.co/index.php/ingenieria/article/viewFile/2382/1545.

Díaz, A y Glover, Fred. 1996. *Optimización heurística y redes neuronales.* Madrid : Parainfo, 1996.

Echevarría Cartaya, Yuviny. 2009. *Sistema Informático para la Confección de Horarios Docentes en la Facultad de Informática de la Universidad de Cienfuegos.* 2009.

F. Suárez, Víctor, Guerrero, Álvaro y Castrillón, Omar. 2012. *Programación de Horarios Escolares basados en Ritmos Cognitivos usando un Algoritmo Genético de Clasificación No-dominada, NSGA-II.* Manizales : Información Tecnológica, 2012. 10.4067/S0718-07642013000100012.

Festa, Paola y Resende, Mauricio G. 2009. *Effective Application of GRASP.* 2009.

Galve Frances, Javier. 1993. *Algoritmica. Diseño y análisis de algoritmos funcionales e imperativos.* s.l. : RA-MA EDITORIAL, 1993. ISBN 978-84-7897-116-9.

Garey, Michael y Johnson, David S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* s.l. : Bell Telephone Laboratories, 1979. ISBN 0-7167-1045-5.

Glover, Fred y Laguna, Manuel. 1993. Tabu Search. Modern Heuristic Techniques for Combinatorial Problems. s.l. : Colin R. Reeves (Ed.), Blackwell Scientific Publications, Oxford, pp. 70-150, 1993.

Gosling, James, y otros. 2005. *The Java language specification.* s.l. : Addison-Wesley, 2005. ISBN 0-321-24678-0.

Jackson, A, y otros. 2004. *Behind the rules: XP.* 2004. págs. 87-94.

Jeffries, Ron, Anderson, Ann y Hendri, Chet . 2001. *Extreme Programming Installed.* s.l. : Addison-Wesley Professional, 2001.

Larico Mullisaca, Celso Ever. 2010. *Un Algoritmo GRASP-Reactivo para resolver el problema de cortes 1D.* Lima : Universidad Nacional Mayor de San Marcos, 2010.

Larman, Craig. 1999. *UML Y PATRONES. Introducción al análisis y diseño orientado a objetos.* Prentice Hall,México : Pearson, 1999. ISBN:970-17-0261-1.

Larrosa, J y Meseguer, P. 2003. *Restricciones blandas: modelos y algoritmos.* s.l. : Revista Iberoamericana de Inteligencia Artificial, 2003. págs. 69–82. Vol. 7.

Lerdorf, Rasmus. 2015. PHP: Hypertext Preprocessor. [En línea] 19 de Febrero de 2015. [Citado el: 24 de Mayo de 2015.] http://php.net/.

Letelier Torres, Patricio, y otros. 2003. *Metodologías Ágiles en el Desarrollo de Software.* Alicante : Grupo ISSI, 2003.

Lorenz, Mark y Kidd, Jeff. 1994. *Object-Oriented Software Metrics.* Nueva Jersey : Englewood Cliffs, 1994.

Martelli, Alex. 2008. *Python. Guía de referencia.* España : ANAYA MULTIMEDIA, 2008.

Martín García, Elena. 2013. *Desarrollo de una aplicación web para la creación de exámenes de opción múltiple.* Madrid : s.n., 2013.

Martinez, Rafael. 2010. www.postgresql.org.es. *www.postgresql.org.es.* [En línea] 2 de Octubre de 2010. [Citado el: 3 de Junio de 2014.] http://www.postgresql.org.es/sobre_postgresql.

Matsumoto, Yukihiro. 2015. Lenguaje de Programación Ruby. *Ruby Programming Language.* [En línea] 2015. [Citado el: 22 de Marzo de 2015.] https://www.ruby-lang.org/es/about/.

McCabe, Tom. 1976. *A Software Complexity Measure.* s.l. : IEEE Trans. Software Engineering, 1976. págs. 308-320. Vol. II.

Mendes Calo, Karla , Estevez, Elsa y Fillottrani, Pablo . 2009. *Un Framework para Evaluación de Metodologías Ágiles.* Argentina : s.n., 2009.

Merelo, Juan Julian. 2004. *Introducción a los algoritmos genéticos.* 2004.

Mora García, Antonio Miguel. 2009. *Resolución del Problema Militar de Búsqueda de Camino Óptimo Multiobjetivo Mediante el Uso de Algoritmos de Optimización Basado en Colonia de Hormigas.* Granada : Departamento de Arquitectura y Tecnología de Computadores, 2009.

Moreno Pérez, José A. 2004. *Metaheurísticas: Concepto y Propiedades.* Universidad de La Laguna : s.n., 2004.

Osorio, Juan Carlos, Lasso, Diego Fernando y Alonso Ruiz, Gabriel. 2012. Job shop scheduling: Biobjetivo mediante enfriamiento simulado y enfoque de Pareto. *Scielo.* [En línea] 2012. [Citado el: 12 de Diciembre de 2014.] http://www.scielo.org.co/pdf/rium/v11n21/v11n21a10.pdf.

Palacio, Juan. 2014. *Scrum Manager I: Las reglas de scrum.* s.l. : Scrum Manager®, 2014. 1404240651012.

Pedemonte, Martín. 2009. *Ant Colony Optimization para la resolución del Problema de Steiner Generalizado.* Montevideo : PEDECIBA, 2009. ISSN 0797-6410.

Peñuela, Cesar Augusto. 2008. *Colonia de hormigas aplicada a la programación óptima de horarios de clase.* 2008.

Pérez Martínez, Leordan y Diaz Morejón, Luis Manuel. 2014. *Sistema para la generación del horario docente de la Facultad de Ingeniería Industrial de la CUJAE.* La Habana : Universidad de las Ciencias Informáticas, 2014.

Piattini Velthuis, Mario G. 1996. *Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión Rama.* Madrid : RA-MA EDITORIAL, 1996.

Pineda Arias y Gustavo, Israel. 2011. Sistema inteligente de soporte en la generación de horarios académicos para la carrera de ingeniería de sistemas de la Universidad Salesiana. *Intelligent support system in generating academic schedules for systems engineering career of the Salesian University.* [En línea] Robótica e Inteligencia Artificial SUR, Julio de 2011. [Citado el: 5 de Marzo de 2015.] http://dspace.ups.edu.ec/handle/123456789/1624.

Pressman, Roger S. 2005. *Ingeniería del Software. Un enfoque práctico.* s.l. : McGraw-Hill, 6ta edición, 2005. ISBN: 9701054733.

RAE. 2015. RAE. *RAE.* [En línea] 23 de Noviembre de 2015. [Citado el: 23 de Noviembre de 2015.] http://lema.rae.es.

Rodríguez Corbea, Maite y Ordóñez Pérez, Meylin. 2007. *La metodología XP aplicable al desarrollo del software educativo en Cuba.* La Habana : Universidad de las Ciencias Informáticas, 2007.

Saavedra Gutierrez, Jorge A. . 2007. El mundo informático. [En línea] 5 de 05 de 2007. [Citado el: 18 de 04 de 2015.] https://jorgesaavedra.wordpress.com/2007/05/05/lenguajes-de-programacion/.

Schaerf, A. 1996. *Tabu search techniques for large high-school timetabling problems.* Amsterdam : Centrum voor Wiskunde en Informatica, 1996. ISSN 0169-118X.

Schaerf, Andrea. 1999. TEI Central Macedonia. *A Survey of Automated Timetabling.* [En línea] 1999. [Citado el: 19 de abril de 2015.] http://www.teicm.gr/arximidis/pdf/kazarlis/PE1/1d.pdf. ISSN: 0269-2821.

Sears, Connors y Rebert, Chris. 2015. Bootstrap The world's most popular mobile-first and responsive front-end framework. *Bootstrap The world's most popular mobile-first and responsive front-end framework.* [En línea] 2015. [Citado el: 24 de Marzo de 2015.] http://getbootstrap.com/.

Sommerville, Ian. 2002. *Ingeniería de Software.* 6ta Edición. s.l. : Prentice-Hall, 2002. ISBN 970-26-0206-8.

The jQuery Foundation. 2015. jQuery. *jQuery.* [En línea] The jQuery Foundation, 2015. [Citado el: 23 de Marzo de 2015.] https://jquery.com/.

Timetable Web. 2012. CALENDARIO WEB - horario escolar en línea - horario de escuela en línea. *TIMETABLE WEB - Online school timetable - online school schedule.* [En línea] TIMETABLE WEB, 2012. [Citado el: 5 de Marzo de 2015.] http://www.timetableweb.com/index.php.

Torres Delgado, José Fidel y Vélez Gallego, Mario César. 2007. *Algoritmo de recocido simulado para la descomposición robusta del horizonte de tiempo en problemas de planeación de la producción.* 2007.

Untis. 2014. Generador de Horarios Untis, Horarios Software / Horario escolares! *Timetable Software Untis - School Scheduling - Timetabling Software for universities.* [En línea] Gruber & Petters GmbH, 2014. [Citado el: 6 de Marzo de 2015.] http://www2.grupet.at/home_es.php.

Villarreal, Gonzalo Luján. 2008. Notas de Scrum. La Plata : Proyecto de Enlace de Bibliotecas (PrEBi), 2008.

Visconti, Marcello y Astudillo, Hernán. 2006. *Fundamentos de Ingeniería de Software.* 2006.

Visual Paradigm International. 2015. Herramientas de diseño de software para equipos ágiles con UML, BPMN y más. *Software Design Tools for Agile Teams, with UML, BPMN and More.* [En línea] 2015. [Citado el: 20 de Marzo de 2015.] http://www.visual-paradigm.com/support/documents/vpuserguide/12/13/5963_visualparadi.html.

Welling, Luke y Thomson, Laura. 2005. *Desarrollo web con php y mysql php 5 y mysqul 4.1 y 5: disco compacto.* Madrid: s.n., 2005.