Trabajo de Diploma para Optar por el Título de Ingeniero en Ciencias Informáticas

Título: Sistema informático de planificación automática de pre-defensas y defensas de los trabajos de diploma en la Facultad 2

Autoras: Lizzi Díaz López

Sulema Pérez Serviño

Tutora: Ing. Bárbara Triana Morales

Msc. Yahima Vigo Valdés

Co-Tutora: Ing. Anié Bermudez Peña

Ciudad de La Habana, junio de 2013

Declaración de Autoría

Declaramos ser autoras 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_____.

_____ _____

Autor **Tutor**

_____ _____

Autor **Tutor**

Pensamiento

El hombre razonable se adapta al mundo; el irrazonable intenta adaptar el mundo a sí mismo. Así pues, el progreso depende del hombre irrazonable.

Dedicatoria

Le dedico esta tesis al amor más grande que tengo en mi vida, mi abuela, por ser mi mayor tesoro. A mi abuelo que es mi sol, faro y guía. A mi hermano por ser mi niño lindo y espero esté orgulloso de verme convertida en una profesional. A mis padres que han sido mi inspiración durante todos estos años.

Lizzi

Le dedico esta tesis a mis padres por toda su dedicación, comprensión, sacrificio y esmero, por confiar tanto en mí. A mi mamá por tu cariño, paciencia, por hacer posible este sueño te dedico todos mis triunfos. A mi papá por ser un ejemplo a seguir y estar siempre a mi lado cuando lo he necesitado. Los amo mucho y me siento muy orgullosa de ustedes.

A toda mi familia por su apoyo incondicional a lo largo de estos años.

Sulema

Agradecimientos

Agradezco a la Revolución por haberme dado la oportunidad de ser una profesional y a esta maravillosa universidad que me ha formado como persona.

Quiero agradecer a mi abuela Lupe, por haber sido madre, padre, abuela, hermana y amiga. Gracias por inculcarme el hábito de estudiar y de ser mejor persona cada día. Quiero agradecer a mi abuelo Julio que siempre me ha aconsejado y apoyado en todas mis decisiones, brindándome sus más sabios consejos. A mi mamá por ser mi compañera y mi amiga. A Izquierdo por apoyarme en todo incondicionalmente. A mi papá por darme fuerzas y ánimos para continuar y a Idarmis y a mi hermana por haberse preocupado por mi todos estos años de estudios. Le agradezco a Liván porque siempre estuvo conmigo brindando su apoyo en todo momento. Gracias por estar siempre pendiente de mí y ayudarme, te quiero. Quiero agradecer a mi compañera de tesis porque sin ella no hubiese sido lo mismo. Ojalá y hubiese tenido la oportunidad de haberla conocido antes, porque es una maravillosa personas y ha sido mi amiga incondicional durante estos meses.

Le agradezco a mis "Curis" (Deylert y Celia) por ser mis amigos del alma y no haberme fallado nunca y a Lore por estar siempre dispuesta a ayudarme. Agradezco a todos los muchachos de la FEU-UCT por hacer que mis días en esta universidad fueran mejores así como a los de la FEU de la facultad por haber compartido tantos momentos que nunca olvidaré y a mis hermanos del Teatro por cada ensayo y cada

Agradecimientos

festival en los que participé . Además quiero agradecerle a Haniel porque estuvo pendiente de cada oración y cada palabra que redacté, ayudándome a corregir y a mejorar todo este documento. No tengo palabras para agradecerle.

A mis tutoras Yahima, Baby y Anié por estar pendientes de todo y ayudarnos incondicionalmente. A todos aquellos que durante estos 5 años han formado parte de mi vida y me han ayudado a estar donde estoy.

Gracias.

Lizzi

Quiero agradecer a mis hermanas Susana y Susel por apoyarme en todas las decisiones y por su cariño.A mis abuelos por ser unos segundos padres, por aconsejarme y quererme tanto.

Agradecer a mis tías Alicia y Esther por malcriarme en todos estos años.

A Xenia por ser mi cómplice y mejor amiga.

A mi compañera de tesis Lizzi por ser mi amiga, soportarme, por tu dedicación y tiempo.

A Marvis por siempre estar dispuesta a escucharme, no juzgarme, por tu sinceridad. Gracias por tu confianza, tu amistad, por estar presente siempre para apoyarme, aconsejarme, guiarme cuando en algo voy mal. Por ser más que una amiga, una hermana, y poder contar siempre con tu amistad.

Agradecimientos

A Anailis y Dayana por todos los años que compartimos juntas, por ser más que mis amigas ser mis hermanas y aconsejarme en todo momento.

A Reyalex por aparecer en mi vida en el momento que más necesitaba un amigo. Por estar siempre dispuesto a escucharme, a darme un consejo o regañarme cuando hacía falta. Por ser mi amigo y confidente, ojala nos hubiéramos conocido antes.

A Arianna, Amado, Yaque, Sandra, Willi y Wendy por compartir conmigo estos años y ayudarme a lograr este sueño que hoy se hace realidad, muchas gracias por su amistad.

A Angel, porque llegaste en el momento oportuno a mi vida, me demostrarte apoyo y amistad incondicional, por ayudar a que este sueño se haga realidad.

A mis tutoras Yahima, Bárbara y Anié por su ayuda y la confianza que depositaron en mi para la realización de este trabajo,gracias.

A los profesores Iris, Maidelis, Julio Omar y,Jaimel por ayudarme en estos años.

Agradecer a todas las personas que conocí durante estos años y que de una forma u otra me han ayudado a mi formación como profesional.

Por último, quisiera agradecerle a la revolución y al Compañero ***Fidel Castro*** *por darme la posibilidad de estudiar en esta universidad tan maravillosa.*

Sulema

Resumen

La planificación docente en las universidades se encuentra enfocada a organizar un conjunto de actividades en períodos de tiempos determinados. Las exposiciones de los trabajos de diploma se incluyen en las actividades docentes a planificar cada cuso académico. En el presente trabajo se propone la construcción de un sistema informático para solucionar las inconsistencias que ha presentado el proceso de planificación de exposiciones de los trabajos de diploma en la Facultad 2 de la Universidad de las Ciencias Informáticas. Por tal motivo se realiza la investigación de heurísticas y metaheurísticas, seleccionando un algoritmo genético que puedan brindar una mejor solución en la confección del horario de exposiciones. Para la implementación del algoritmo es necesaria la incorporación del *framework* JGAP (en inglés *Java Genetic Algorithms Package*) el cual brinda un conjunto de mecanismos genéticos ya programados. Para validar la propuesta del algoritmo se desarrolló un sistema informático que permite la planificación del horario de exposiciones para las pre-defensas y defensas de los trabajos de diploma.

Palabras clave: planificación de horarios, *timetabling*, algoritmo genético.

Índice

Índice de Figuras

Índice de Tablas

Introducción

La vida diaria de toda persona requiere de la elaboración de un esquema mental sobre ¿qué? y ¿cómo? va a realizar diferentes actividades. Todo individuo necesita planificar a corto, mediano y largo plazo, para prever, prepararse y dirigir los planes diarios hacia nuevas metas. Desde que el hombre se convirtió en un ser racional, asociar el surgimiento de la planificación al surgimiento del raciocinio no es un fenómeno casual. La planificación "*es considerada un proceso racional que requiere de la inteligencia organizada del hombre para poder ser ejecutada*" [1]. Uno de los exponentes más destacados en materia de planificación, James Arthur Finch Stoner, planteó: *"la planificación no es más que el proceso de establecer metas y definir medios para alcanzar las mismas"* [2].

En la actualidad la planificación se realiza a partir de intervalos de tiempo durante el cual se desarrolla habitual o regularmente una actividad, también conocido por el nombre de horario [3]. Los horarios (en inglés *timetables*) son listas organizadas usualmente en formato tabular de filas y columnas, informando sobre una serie de eventos organizados, particularmente relacionados con el tiempo. La generación de horarios es uno de los problemas clásicos de las ciencias de la computación, definido como: *"la asignación sujeta a restricciones, de un grupo de recursos ubicados en tiempo y espacio, de tal manera que se satisfagan un conjunto de objetivos deseados"* [3]. Los problemas de asignación de horarios pertenecen al campo de la optimización combinatoria y pueden modelarse matemáticamente como problemas de programación lineal [1] [4].

Diferentes áreas de la sociedad necesitan planificar sus procesos para lograr el cumplimiento de objetivos que fueron trazados. La asignación de horarios en el área docente es definida como: *"la asignación de cierto número de eventos a un determinado número de espacios horarios y locales de clase, respetando ciertas restricciones"* [5]. La asignación de horarios en instituciones educativas consiste en planificar (generalmente en

[1] Programación Lineal: procedimiento o algoritmo matemático aplicado a la resolución de problemas, formulados a través de un sistema de inecuaciones lineales, optimizando la función objetivo, también lineal.

una semana) las asignaturas que se imparten en un período académico (año, trimestre o semestre), los profesores a distribuir por asignatura y los grupos de alumnos que reciben las mismas. Además debe tenerse en cuenta los días, períodos de horas disponibles y los locales a incluir en la planificación.

Las instituciones universitarias realizan una planificación de las actividades docentes con el objetivo de mantener la organización del proceso de enseñanza. El proceso docente en las universidades incluye la planificación de evaluaciones para la culminación de estudios. Cada centro educativo tiene como tarea organizar las evaluaciones que realizarán los estudiantes para culminar estudios. Las universidades cubanas se rigen según el Reglamento Metodológico del Ministerio de Educación Superior (MES). En el artículo 149 queda establecido que la defensa del trabajo de diploma y la prueba estatal son tipos de evaluación de la culminación de los estudios cuyo objetivo es comprobar el grado de dominio de los estudiantes de los objetivos generales de la carrera [6].

En la Universidad de las Ciencias Informáticas (UCI) los estudiantes deben desarrollar un trabajo de diploma con el fin de demostrar las habilidades adquiridas durante los cursos académicos. Las facultades por las que está compuesta la UCI tienen como tarea realizar el proceso de planificación de los trabajos de diploma en un rango de tiempo previamente establecido por la Vicerrectoría de Formación. La facultad 2 de la UCI desarrolla cada curso el proceso de planificación de las exposiciones de los trabajos de diploma. El proceso consiste en la confección de un horario teniendo en cuenta la disponibilidad de los locales, la cantidad de tesis a planificar y el tribunal asociado a las mismas. Actualmente los locales destinados a la planificación de las exposiciones son utilizados para planificar actividades docentes. Por tal motivo los locales se encuentran parcialmente ocupados en el período de tiempo asignado para realizar la planificación.

Para evitar coincidencias en la planificación es necesario cumplir con un conjunto de restricciones que garanticen un horario que cumpla con las necesidades de la facultad. Las restricciones de obligatorio cumplimiento que deben tenerse en cuenta son: un profesor miembro de un tribunal no puede ser asignado a más de una actividad docente al mismo tiempo, un tribunal no puede estar asignado a más de una discusión de tesis al mismo tiempo y un local no debe ser asignado a más de una actividad docente a la misma hora. El proceso de planificar en estos casos se vuelve complejo, susceptible a errores y

vulnerable a cambios de última hora. Los problemas que ha presentado la planificación conllevan a una obligatoria redistribución (nueva planificación) del horario de discusiones, afectando en ocasiones su entrega en tiempo.

Atendiendo a todo lo expresado anteriormente se ha identificado el siguiente **problema a resolver**: ¿Cómo garantizar que el proceso de planificación de las exposiciones de pre-defensas y defensas de los trabajos de diploma cumpla con el conjunto de restricciones exigidas?

El **objeto de estudio** de la investigación se centra en el proceso de planificación de actividades docentes.

El **campo de acción** está enmarcado en los sistemas de planificación automática de horarios docentes.

Se define como **objetivo general** desarrollar un sistema informático para automatizar el proceso de planificación de las exposiciones de pre-defensas y defensas de los trabajos de diploma. Para dar cumplimiento al objetivo general planteado se proponen los siguientes **objetivos específicos**:

- Modelar el problema de asignación de horarios de las exposiciones de los trabajos de diploma.
- Diseñar el algoritmo que permita resolver el problema de planificación de exposiciones de los trabajos de diploma.
- Implementar el sistema informático utilizando el algoritmo diseñado.
- Validar la solución informática a través de pruebas para comprobar que la planificación propuesta es adecuada con los datos especificados.

El trabajo se encuentra estructurado en cuatro capítulos, los cuales se especifican a continuación:

Capítulo 1 Fundamentación Teórica: Se realiza un estudio relacionado con los principales conceptos y características del proceso de planificación. Además se analizan algunos sistemas de planificación docente y los principales algoritmos que pueden ser

aplicados para darle solución al problema. Se explican las herramientas y metodologías seleccionadas para realizar la implementación del sistema.

Capítulo 2 Análisis de la solución: Se describen las características esenciales que debe tener la solución propuesta, sustentadas por la modelación de los procesos de negocio y la definición de los requisitos.

Capítulo 3 Diseño del Sistema: Se abordan todos los elementos referidos al diseño de la arquitectura del sistema, los diagramas de clases y los patrones de diseño utilizados.

Capítulo 4 Implementación y Prueba: Está orientado a la realización del diagrama de componentes, el diseño del algoritmo seleccionado y la implementación del mismo utilizando un *framewok* que provee los principales operadores genéticos. Por último se validará el sistema a través de pruebas comprobando que la planificación propuesta por el sistema informático es adecuada con los datos especificados.

Capítulo 1: Fundamentación Teórica

En el presente capítulo se analizan los aspectos relacionados con el campo de la optimización combinatoria, los sistemas de planificación de horarios, así como las principales técnicas que se han aplicado a la solución de problemas de asignación de horarios. Se fundamenta la selección de tecnologías y herramientas para el desarrollo del sistema.

1.1 Problemas de optimización combinatoria

Los problemas de optimización combinatoria se encuentran enfocados en la búsqueda de la mejor combinación, construyendo funciones sobre el espacio de las combinaciones que permitan determinar cuál es la mejor. El objetivo de estos problemas es encontrar el máximo (o el mínimo) de una determinada función sobre un conjunto finito de soluciones denotadas por S. El conjunto de soluciones S trabaja con variables discretas[2], restringiendo su dominio a valores finitos [7].

Para describir los problemas de optimización de forma mucho más concisa es necesario desarrollar un modelo matemático. Si el modelo matemático de cualquier problema de optimización se ajusta al formato general del modelo de programación lineal (PL), el problema es considerado entonces un problema de programación lineal [8]. El modelo matemático en los problemas de PL se formula a partir de la cantidad de recursos disponibles, el conjunto de restricciones lineales para definir las soluciones admisibles y la función objetivo que debe ser optimizada (minimizada o maximizada) [9].

Los problemas donde las variables que intervienen tienen valor entero reciben el nombre de problemas de Programación Entera (PE). El modelo matemático es formulado como un modelo de programación lineal teniendo en cuenta que todas las variables que intervienen deben ser enteras [10]. Los problemas de optimización combinatoria presentan un espacio de soluciones elevado y la evaluación de todas sus soluciones para determinar el óptimo conllevan mucho tiempo computacional [11]. Por tal motivo se necesitan procedimientos que aseguren un menor tiempo computacional. Existen técnicas que son aplicadas a problemas de este tipo como es el caso de las heurísticas y las metaheurísticas [12].

2 Variables discretas: son variables cuantitativas que toman valores aislados, no admiten valores intermedios entre dos valores específicos.

Los métodos heurísticos se utilizan para encontrar soluciones a problemas para los que no existe un algoritmo que converja a la solución ni una fórmula explícita que la encuentre. Un método heurístico es definido como: *"un procedimiento para resolver un problema de optimización mediante una aproximación intuitiva, donde la estructura del problema se utiliza de forma inteligente para obtener una buena solución"* [13]. Las heurísticas pueden estancarse en soluciones de baja calidad (óptimos locales muy lejanos al óptimo global). Para permitir una mejora adicional en la calidad de las soluciones han sido diseñadas técnicas de propósito general que guían la construcción de soluciones o la búsqueda local en las distintas heurísticas. Estas técnicas son conocidas por el término metaheurísticas [14].

Las técnicas metaheurísticas son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, donde las heurísticas no son efectivas. Las metaheurísticas proporcionan un marco general para crear nuevos algoritmos híbridos, combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos. Son aplicadas a algoritmos que dan solución a los problemas de optimización combinatoria [15].

El problema de planificación de las exposiciones de pre-defensas y defensas de los trabajos de diploma es clasificado como un problema de optimización combinatoria. Se clasifica de esta forma debido a que el proceso de planificación está enfocado en la búsqueda de la mejor combinación de los tribunales y locales en un determinado período de tiempo. El modelo matemático define una función objetivo y un conjunto de restricciones para evaluar la calidad de una planificación, pudiendo modelarse como un problema de Programación Lineal. Teniendo en cuenta que las variables que intervienen en la solución del problema son discretas (cantidad de aulas, cantidad de tribunales y turnos de clase), puede clasificarse de forma más específica como un problema de Programación Lineal Entera.

1.1.1 Problemas de Asignación de Horarios

Los Problemas de Asignación de Horarios (PAH) (en inglés timetabling problem) son definidos como: *"un problema de asignación, sujeto a un conjunto de restricciones y recursos disponibles en un espacio de tiempo, de tal manera que se satisfaga en la medida de lo posible un conjunto de objetivos deseables"* [16]. Los PAH pertenecen al área de la optimización combinatoria, donde cada horario representa una solución al problema [17]. Para confeccionar un horario es necesario tener en cuenta la disponibilidad horaria de varios

recursos sujetos a un conjunto de restricciones. Para modelar el PAH es necesario definir las restricciones asociadas y la función objetivo que se tendrá en cuenta para evaluar las soluciones. En el problema se suelen identificar dos tipos de restricciones: fuertes y débiles.

- Restricciones fuertes: Son de estricto cumplimiento, definen la factibilidad o validez del horario.
- Restricciones débiles): No son de estricto cumplimiento, pero su grado de satisfacción define la calidad del horario.

Las restricciones y los criterios que determinan si un horario es mejor que otros son factores que varían considerablemente, dependiendo de los requerimientos específicos de cada institución. Por tal motivo se hace muy difícil diseñar una solución que se adapte a todas las instancias. El problema de asignación de horarios propone una planificación que cumpla con todas las restricciones impuestas y obtenga una solución de calidad en un tiempo razonable [18].

Teniendo en cuenta las definiciones anteriores el problema de planificación de los trabajos de diploma en la Facultad 2 es clasificado como un problema de asignación de horarios. Para realizar el proceso es necesario verificar la disponibilidad horaria de los locales que serán incluidos en la planificación y la disponibilidad de los miembros asignados a un tribunal de trabajo de diploma con el objetivo de encontrar un horario común para todos. Para la confección del horario luego de encontradas las disponibilidades de los recursos que intervendrán se tiene en cuenta varias restricciones que responden a las necesidades de la facultad.

Con los avances de la informática muchos problemas de asignación de horarios presentes en los centros educativos han sido resueltos a través de sistemas informáticas de planificación, ahorrando tiempo en su confección y garantizando buenos resultados.

1.2 Sistemas de planificación automática de horarios docentes

Existen disímiles sistemas de planificación de horarios que proponen soluciones a problemas de planificación de horarios docentes. En dependencia de las características del centro de estudios pueden ser adaptados a problemas de planificación docente más específicos. A

continuación se describen algunos de los sistemas generadores de horarios docentes que constituyen la tendencia actual al planificar en los centros educativos:

- **GHC 2012** (Generador de Horarios para Centros de Enseñanza): Permite la confección, edición y transferencia de horarios escolares semanales. Para utilizar GHC 2012 se requiere Microsoft Windows al menos con la versión XP. Está limitado a solo 10 profesores y para ampliar la capacidad en el número de profesores y recibir mantenimiento, debe solicitarse una licencia [19].

- ***Timetab*:** Potente *software* generador de horarios escolares sencillos de manejar. Presenta un asistente para la introducir los datos y permite generar rápidamente un horario para un centro educativo. Presenta como limitación que solo es aplicable a un máximo de 4 grupos y 6 profesores. Compatible para sistema operativo: Windows en sus versiones 95/98/Me/NT/2000/XP y es un *software* propietario [20].

- ***LantivTimeTabler*:** Generador de horarios creado por Lantiv Internacional. Solo se encuentra disponible para el sistema operativo Windows sus versiones /2000/XP/Vista y no permite importar de una base de datos previamente creada [21].

- ***Mimosa software***: Aplicación planificadora de cursos, utilizada en más de 60 países, permite la creación de calendarios de forma automática o interactiva. Trabaja en todas las versiones de WINDOWS (3.11/9x/ NT/ 200x/ /XP), necesita 4 Mb de memoria RAM para poder ejecutarse [22].

La facultad 2 necesita de un sistema que sea adaptable a características específicas, permitiendo manejar gran cantidad de recursos. En necesario un sistema dinámico que pueda aceptar cambios de última hora. Los sistemas de planificación de horarios docentes estudiados no son adecuados para resolver el problema de planificación de exposiciones de los trabajos de diploma en la facultad, debido a que la mayoría de ellos no permiten: la planificación de gran cantidad de recursos, la realización de cambios imprevistos y la introducción de nuevas restricciones. Los sistemas mencionados anteriormente no son multiplataforma, son privativos, el código fuente no se encuentra disponible para los usuario y están basados en una planificación estática por lo que no pueden ser adaptados a las condiciones de la UCI. Por las razones antes planteadas se concluye que es necesaria la construcción de un sistema de planificación que satisfaga las necesidades de la facultad.

Como el espacio de soluciones para el problema de confección de horario es elevado es necesario el empleo de técnicas que operen aportando buenas soluciones con un bajo costo computacional [23].

1.3 Técnicas metaheurísticas para la planificación automática de horarios

Para dar solución a problemas de planificación es necesario incorporar métodos que garanticen una solución aceptable. La planificación de las exposiciones de los trabajos de diploma por las características que presente se clasifica como un problema de optimización combinatoria y puede modelarse como un problema de programación lineal. El espacio de soluciones que puede generar las combinaciones de salones, tribunales y horas disponibles para ambos es elevado. Dada la dificultad práctica para resolver de forma exacta el problema, es necesario comenzar un estudio de las principales técnicas metaheurísticas que proporcionan soluciones factibles. A continuación se explicarán las más se utilizadas en problemas de asignación de horarios.

1.3.1 Búsqueda Tabú

La Búsqueda Tabú (BT) pertenece a la clase de técnicas de búsqueda local y fue diseñada para obtener una aproximación a la solución óptima de un problema de optimización combinatoria. El rendimiento del método de búsqueda local aumenta mediante el uso de estructuras de memoria que se incorporan en el procedimiento. La configuración inicial de la búsqueda puede ser obtenida aleatoriamente, por medio de un algoritmo constructivo o alguna técnica heurística que utilice índices de sensibilidad. Una configuración aleatoria puede tener la ventaja de evitar una convergencia prematura, pero una configuración inicial de mala calidad conduce a un esfuerzo computacional mayor. La principal desventaja de este método es el hecho de que se puede mover a una solución peor [24].

1.3.2 Recocido Simulado

El Recocido Simulado (en inglés *Simulated Annealing*) es una técnica para resolver problemas de optimización combinatoria. Los algoritmos de recocido pueden combinarse con otras técnicas heurísticas como: los sistemas expertos, los algoritmos genéticos, las redes neuronales, entre otros, consiguiendo sistemas híbridos que pueden resultar de gran potencia en la resolución de problemas muy complejos. La principal desventaja que tiene esta técnica es que no guarda la información de movimientos previos para guiar los nuevos movimientos,

pudiendo encontrar varias veces la misma solución. Para resolver problemas de gran complejidad es necesario realizar muchas ejecuciones para encontrar una solución satisfactoria. La solución final suele encontrarse en un mínimo local[3] no explorando con suficiente amplitud el espacio de soluciones y una ejecución del problema puede requerir mucho tiempo de cálculo [25].

1.3.3 Método de la escalada máxima (en inglés *Hill Climbing)*

El método de la escalada máxima es un método iterativo de búsqueda local que es empleado en la solución de problemas de optimización. La técnica de *Hill Climbing* hace uso del cálculo de la función objetivo para la vecindad del punto actual y de esta forma determinar hacia donde crece la función, seleccionando el punto de mayor pendiente. Si el valor de la función objetivo es mejor en el punto nuevo, el punto anterior se reemplaza por el que fue hallado. Este proceso continúa hasta que no es posible encontrar ninguna mejora. Como se observa en la siguiente figura, una de las desventajas de este método es su incapacidad para escapar de óptimos locales. Si la exploración inicia en el punto "p", por ejemplo, el algoritmo encontrará el máximo local "a" sin llegar nunca al máximo global "b", ya que para hacerlo tendría que atravesar la región "r", que posee valores peores que "a" [26].

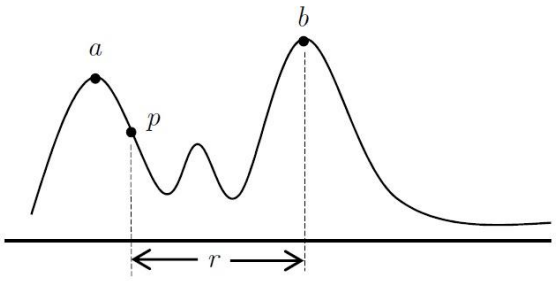

Figura 1: Representación del método Hill Climbing

1.3.4 Algoritmos Genéticos

Los métodos más utilizados para la solución de problemas de asignación de horario son los Algoritmos Genéticos (AG). Los AG simulan el proceso de evolución natural, usando un conjunto de términos de las ciencias biológicas para un mejor entendimiento. El algoritmo se basa en cambios aleatorios de soluciones candidatas, utilizando la función objetivo para

[3] Mínimo Local: valor de una función, que es menor que los valores de la función en puntos cercanos, pero que no es el menor de todos los valores.

determinar si esos cambios producen una mejora al proceso. Para la solución se basan en operadores probabilísticos, en vez de los típicos operadores determinísticos de las otras técnicas. La principal desventaja es que pueden converger prematuramente si se utilizan poblaciones pequeñas, donde una variación aleatoria en el ritmo de reproducción provoca que una solución sea dominante sobre las otras [27].

1.4 Propuesta del algoritmo a implementar

La metaheurística propuesta para obtener una solución automatizada que reduzca el tiempo y la ocurrencia de los errores es un Algoritmo Genético (AG). Se selecciona un algoritmo genético debido a que algoritmos intrínsecamente paralelos, realizan la búsqueda de la solución de forma simultánea con varios individuos. Son los algoritmos que hacen una barrida mayor del subespacio de posibles soluciones válidas y son los más exploratorios disponibles para los problemas de optimización [28].

Son algoritmos estocásticos[4] simples y potentes, que basan su funcionamiento en el uso de una función objetivo, una codificación de los parámetros y no los parámetros mismos. Pueden optimizar un parámetro hasta el punto en donde ese parámetro no puede mejorarse más sin causar una correspondiente pérdida de calidad en algún otro parámetro [28]. Estos algoritmos dependen de un buen número de parámetros: los operadores de cruce y mutación, sus respectivas probabilidades, el tamaño de la población y otras más que se explicarán en el epígrafe siguiente. En el desarrollo del sistema propuesto se hace necesario el uso de un algoritmo genético que garantice mejores resultados. Los AG son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Están basados en el proceso genético de los organismos vivos [29].

1.4.1 Composición de un Algoritmo Genético

Los algoritmos genético emplean de la biología varios términos para ejecutar las acciones que realizan [29]. En la siguiente tabla se resumen los conceptos utilizados y su respectiva interpretación en los algoritmos genéticos.

[4] Algoritmos Estocásticos: algoritmos que operan utilizando métodos probabilísticos para solucionar problemas.

Tabla 1: Conceptos fundamentales.

Población	Conjunto de individuos o cromosomas. Equivale a un conjunto de soluciones alternativas.
Cromosoma	Codificación de un individuo solución en una estructura manipulable computacionalmente.
Gen	Subestructura de un cromosoma.
Alelos	Conjunto de posibles valores que puede tener la subestructura gen.
Desempeño	Valor resultante del proceso de evaluación de un cromosoma que incide en el proceso de selección.
Selección	Mecanismo mediante el cual se eligen determinados cromosomas que formarán parte de la población que se cruzará.
Cruzamiento	Intercambio de genes entre dos estructuras cromosoma llamadas padres, que originan una estructura nueva llamada hijo.
Mutación	Cambio aleatorio de alelos dentro de una subestructura gen.
Generación	Ciclo selección-cruzamiento-mutación para una población determinada.

1.4.2 Funcionamiento de un Algoritmo Genético

Los genes son la unidad fundamental de los algoritmos genéticos y la composición de varios genes son conocidos como cromosomas. Inicialmente la población se genera de manera aleatoria, y se compone de un conjunto de cromosomas, que representan las posibles soluciones de un problema. A cada miembro de la población se le aplica una función de aptitud con el propósito de saber que tan buena es la solución que se está codificando. Posteriormente se procede a hacer la selección de los cromosomas que van a cruzarse en la próxima generación eligiendo una técnicas de selección. El cruzamiento es el principal operador genético y para su realización es necesario seleccionar el operador de cruzamiento que va a operar sobre los cromosomas. En el proceso de cruzamiento se combinan las características de los cromosomas padres. También existe un operador de mutación que

cambiará de manera aleatoria los alelos de un cromosoma. Para realizar el proceso es necesario conocer la probabilidad de mutación, la cual indica los individuos que pueden mutar. El proceso de selección para determinar que alelo del gen será cambiado se realiza aleatoriamente [29].

La siguiente imagen muestra de forma detallada el funcionamiento de un algoritmo genético empleando el tipo de codificación binaria, el operador de cruzamiento de la Rueda de la Ruleta y el método de cruzamiento en un punto. Los parámetros a seleccionar en los algoritmos genéticos se ajustan según las características del problema a resolver.

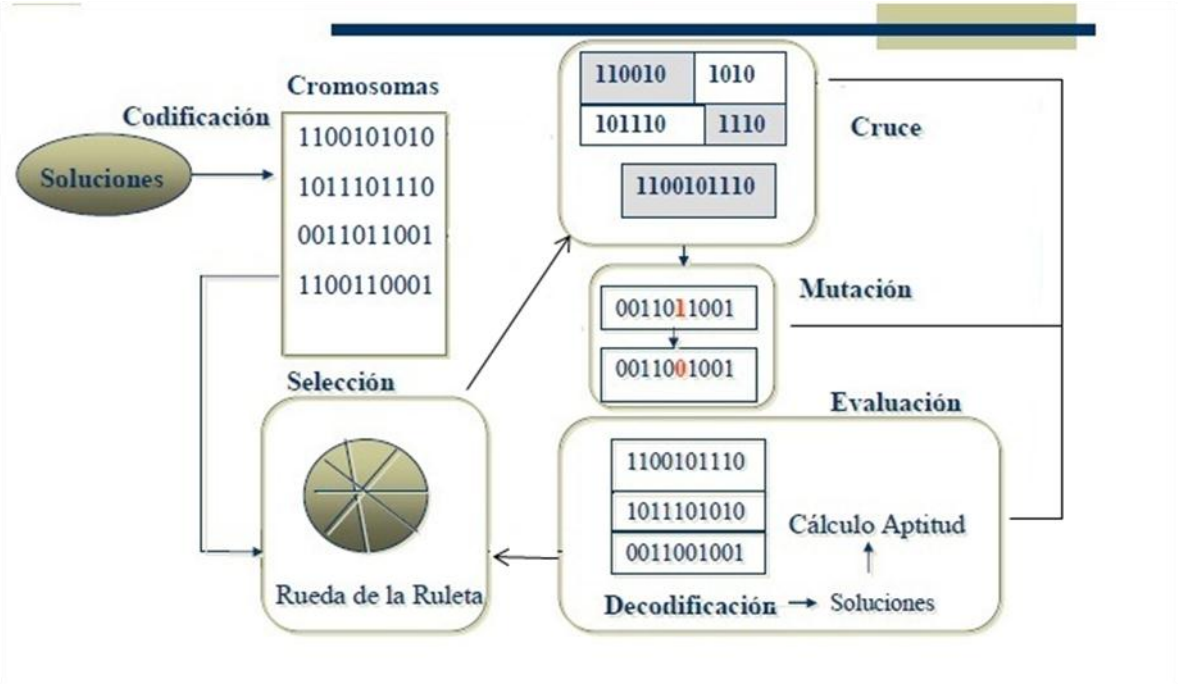

Figura 2: Representación de un Algoritmo Genético.

1.5 Metodologías y herramientas para el desarrollo de *software*

Las metodologías de desarrollo de *software* abarcan todo el ciclo de vida del *software* [30]. En la construcción del sistema es necesaria la selección de una metodología de desarrollo que documente detalladamente el proceso de desarrollo. El proceso de selección de las herramientas es fundamental en la modelación e implementación del sistema propuesto.

1.5.1 Proceso Unificado de Desarrollo (RUP)

Se propone utilizar como metodología de desarrollo RUP porque es necesario el uso de una metodología tradicional que permita definir una documentación detallada, para posibles versiones futuras que mejoren el producto final donde el equipo de desarrollo varíe. RUP se basa en procesos predefinidos con documentación muy precisa y una detallada planificación inicial que debe seguirse estrictamente. Esto se tiene en cuenta ya que se propone que el *software* quede completamente documentado y sea entregado en tiempo gracias a la planificación inicial que se realizó para su construcción. RUP no cuenta con la participación directa del cliente, evitando una fuerte dependencia entre el cliente y el proceso de desarrollo. Utilizando RUP se eliminan muchos de los problemas que son difíciles de superar; la ambigüedad, lo incompleto y las inconsistencias [31].

1.5.2 Lenguaje Unificado de Modelado

Lenguaje Unificado de Modelado (UML por sus siglas en inglés, *Unified Modeling Language*), es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Ofrece un estándar para describir un plano del sistema, incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. Los lenguajes de modelado son usados en combinación con las metodologías de desarrollo de *software* para avanzar de una descripción inicial a un plan de implementación y comunicar dicho plan a los desarrolladores [32].

1.5.3 Notación de Modelado de Proceso de Negocio (BPMN)

Se propone utilizar como notación de modelado BPMN (en inglés *Business Process Modeling Notation*) ya que permite coordinar la secuencia de los procesos y los mensajes que fluyen entre los participantes de las diferentes actividades de una forma entendible. Está orientada a procesos, que combinándolos entre sí se puede lograr una mayor exactitud a la hora de modelar la situación existente. BPMN es una notación expresiva, que cuenta con menos símbolos fundamentales, pero con más variaciones de estos, siendo fácilmente entendible por todos los usuarios del negocio, tanto para los analistas que crean el borrador inicial de los procesos como para los técnicos y desarrolladores responsables de la implementación tecnológica [33].

1.5.4 Herramienta CASE

Se propone utilizar como herramienta CASE (en inglés *Computer Aided Software Engineering*) Visual Paradigma en su versión 8.0 por ser una herramienta que sustenta UML y su licencia fue adquirida por la UCI. Es una herramienta que permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Es considerada una herramienta muy completa, fácil de usar y con soporte multiplataforma. Su uso es sencillo para la creación de todo tipo de diagramas UML, para los que dispone de un número considerable de estereotipos, permitiendo mayor entendimiento de los mismos. La herramienta también proporciona abundante documentación de UML, como tutoriales, demostraciones interactivas y proyectos, facilitándole el uso a personas inexpertas [34].

1.5.5 Lenguaje de programación

Se selecciona Java como lenguaje de programación para la construcción del *software* porque permite la incorporación de librerías que facilitan la implementación del algoritmo genético propuesto. Es un lenguaje robusto que contiene varias funciones integradas y los errores se detectan en el momento de producirse, lo que facilita la depuración, contribuyendo a mejorar la fiabilidad de su uso. Permite a los desarrolladores de aplicaciones escribir un programa una vez y luego ser capaz de ejecutarlo en cualquier lugar. La aplicación a implementar es de escritorio y se debe ejecutar independientemente del sistema operativo, necesitando utilizar un lenguaje multiplataforma [35].

1.5.6 Entorno de Desarrollo Integrado

Se selecciona como IDE[5] (en inglés *Integrated Development Environment*) NetBeans en su versión 7.3 debido a que está disponible en múltiples plataformas y permite a los desarrolladores escribir, compilar, depurar y ejecutar programas en lenguaje java. *"Es un IDE de código abierto escrito completamente en Java permitiendo crear aplicaciones de escritorio".*

[5]IDE: conjunto de programas que corren desde una interfaz de usuario. Por ejemplo: los lenguajes de programación que frecuentemente incluyen un editor de texto, un compilador y depurador que son activados y funcionan desde un menú común.

Netbeans permite la integración con *frameworks*[6] que permiten la implementación de un algoritmo genético de una forma más dinámica [35].

1.5.7 Framework JGAP

Para la implementación del algoritmo genético se propone la utilización del *framework* JGAP (en inglés *Java Genetic Algorithms Package*) el cual brinda un conjunto de mecanismos genéticos programados y permite la implementación propia de una función de optimización para enmarcar la mejor solución. Se distribuye bajo licencia GPL (en inglés *General Public License*) [36]. Las clases del *framework* utilizadas para la construcción del algoritmo son:

Configuración: Esta clase contiene una configuración por defecto para la ejecución del algoritmo genético y puede ser modificada en cualquier momento en dependencia de las necesidades del programador. Dentro de la configuración se encuentran los operadores genéticos como son el operador de selección, el operador de mutación y el de cruzamiento, además cuenta con la dimensión de la población. Esta clase tiene asociada un método llamado evolución (en inglés *evolve*), destinado a realizar el proceso de selección, cruzamiento y mutación de los individuos de la población.

FintnessFuntion: Clase que permite evaluar la adaptación en el entorno a cada individuo.

Interface GeneticOperator*:* Interfaz que contiene los distintos tipos de operadores genéticos que posee el *framework* los cuales son: Operador de cruzamiento y operador de mutación.

NaturalSelector: Clase abstracta que posibilita utilizar un método de selección para los individuos. En la estructura de clases tiene como hijos las clases SelectorTorneo, Selector Ruleta y el SelectorMejorCromosoma, las cuales son los distintos tipos de selección a utilizar.

BaseGene: Este clase contiene métodos para el trabajo con los genes, permitiendo codificar y descodificar la información contenida en los mismos.

Conclusiones del capítulo

[6] Frameworks: conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

En el capítulo se definieron las características que intervienen en el proceso de planificación de exposiciones de pre-defensas y defensas de los trabajos de diploma en la Facultad 2, permitiendo clasificarlo como un problema de asignación de horario. Además se realizó un análisis de las inconsistencias que presenta la planificación de las exposiciones de los trabajos de diploma permitiendo identificar la necesidad inmediata de la automatización del proceso. También fueron analizadas varias técnicas para dar solución al problema, seleccionando un algoritmo genético que permita la confección de un horario automático. Fueron seleccionadas las metodologías, herramientas y tecnologías que serán utilizadas en el proceso de construcción del *softwar*e.

Capítulo 2: Análisis de la Solución

El presente capítulo aborda temas relacionados con la modelación del negocio y diseño del sistema informático. Se definen los requisitos funcionales y los no funcionales que debe cumplir el *software*. Se modela el proceso de negocio

2.1 Descripción de los procesos de negocio

El proceso de planificar la exposición de defensas y pre-defensas de tesis en la Facultad 2 inicia cuando la Vicerrectoría de Formación de la UCI define la fecha para la realización de las exposiciones. El planificador procede a ubicar el período de tiempo en el calendario, consultando la planificación docente de los profesores por el horario, para determinar la disponibilidad de cada profesor. Luego el planificador busca los locales con los que cuenta la facultad y analiza los locales disponibles en algún momento dentro del período de días especificado para su selección, priorizando los locales que la tecnología se encuentra en buen estado. Existen dos tecnologías necesarias a tener en cuenta para la discusión de las tesis; televisor y computadora donde pueden funcionar indistintamente y en función de esto se asigna una prioridad al local. A continuación se definen las prioridades de los locales:

- Local con tecnología óptima: cuenta con televisor y computadora en perfecto estado funcional.
- Local con tecnología alta y con tecnología media: se establece en dependencia de las condiciones que proponga el planificador.
- Local con tecnología baja: no funcionan televisor ni computadora.

El estado de la tecnología de un local determinado puede variar en cualquier momento. El resultado de los locales seleccionados es almacenado en un registro que consta del número del local, número del edificio docente y los rangos de tiempo desocupado dentro del período de discusiones de pre-defensas y defensas de los trabajos de diploma. Luego el planificador procede a buscar en el horario los rangos de tiempo en que tiene clase cada miembro de los tribunales de tesis, con el objetivo de encontrar los diferentes momentos en que todos los miembros de un tribunal están disponibles. El proceso se repite por cada tribunal, obteniendo como resultado las coincidencias existentes entre las disponibilidades de sus miembros.

Capítulo 2: Análisis de la solución

A partir de la información del acabado de las tesis y de los resultados alcanzados en los cortes evaluativos según al centro al que pertenezcan las tesis (Centro de Informatización de la Seguridad Ciudadana (ISEC), Centro Telemáticas (TLM) o departamento docente) se asigna un estado a la misma (avanzado, medio, atrasado o discutido).

El estado asignado influye directamente en el orden en que serán planificadas las tesis. Ubicando primero las avanzadas, después las de estados medio y por último las que tienen estado atrasado. El estado discutido será asignado a las tesis que ya fueron discutidas para que no sean incluidas en la planificación.
Una vez que se tienen las disponibilidades de los locales en cuanto al tiempo y las coincidencias de los miembros de los tribunales el planificador procede a asignar los tribunales a los locales teniendo en cuenta los siguientes factores:

- Cantidad de tesis por tribunal, procurando poner las tesis de un tribunal en el mismo día para que no tenga que volver a reunirse.
- Procurar que todas las tesis de un tribunal se discutan en un mismo local si son en la misma sesión del mismo día.
- Tener en cuenta que se deben poner primero las tesis que estén más adelantadas.
- Priorizar en la planificación los locales donde la tecnología sea óptima (en perfecto estado televisor y computadora).

El planificador al concluir con la planificación a realizar una revisión de la misma para asegurarse que no existen tesis sin planificar. En el caso de que alguna tesis se encuentre fuera de planificación, se buscan las posibles causas del incidente y se planifica en un local que disponible en cualquier horario. La planificación final puede ser modificada según las necesidades existentes con los miembros del tribunal y locales:

- Necesidad de un período de tiempo más largo que el asignado en la planificación.
- Coincidencias de la planificación con actividades docentes, productivas o reuniones.
- Algún miembro del tribunal o tesista presente problemas personales.
- Asignación de local planificado a otra actividad docente.

- Tecnología afectada sin posibilidad de reposición de algún local donde se haya planificado una exposición.

2.2 Modelación del negocio

Con el objetivo de realizar una representación previa del sistema a implementar se realiza una modelación del negocio.

2.2.1 Diagramas de Procesos de Negocio (BPMN)

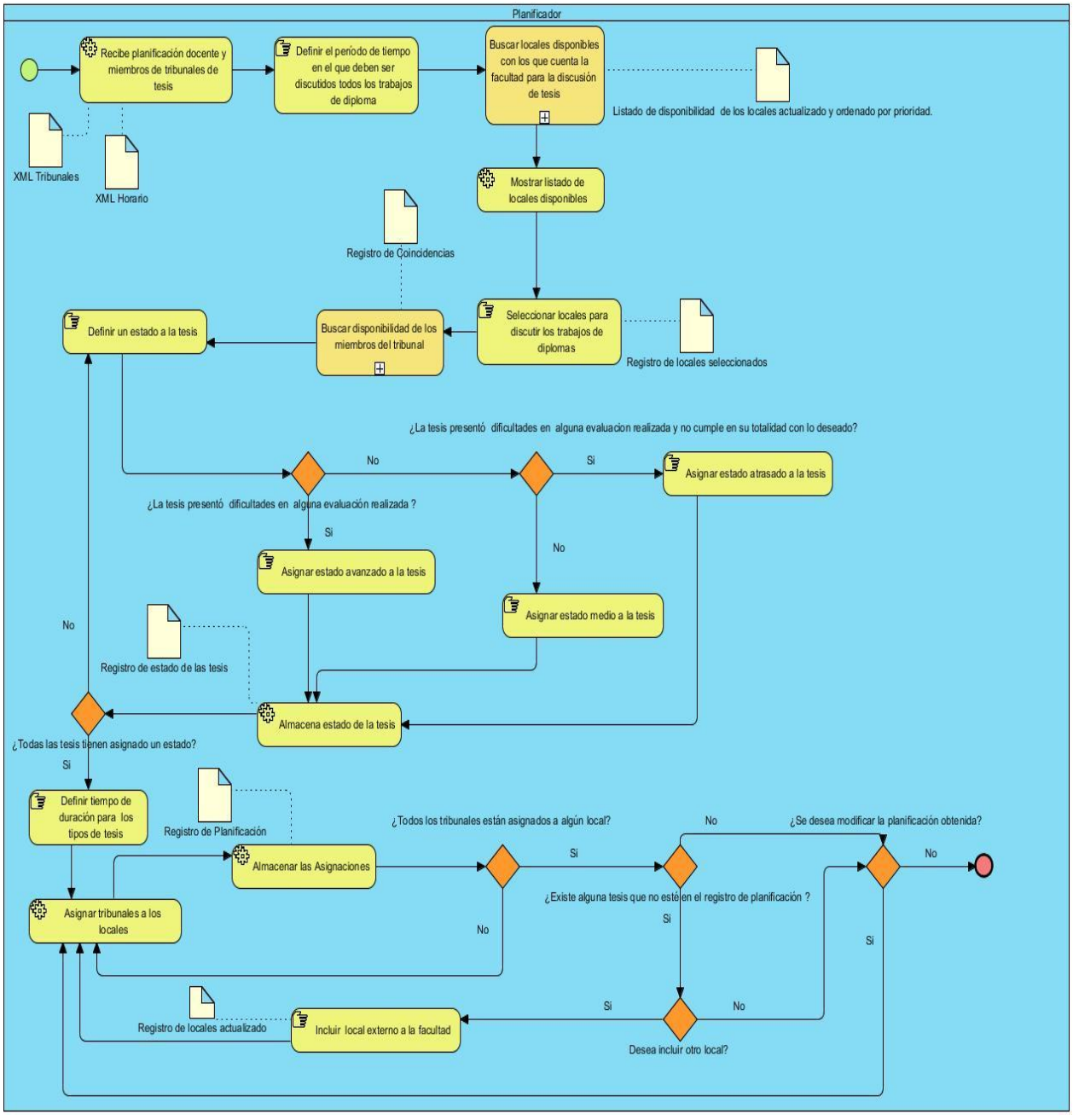

Figura 3: Planificación de exposición de pre-defensas y defensas de tesis.

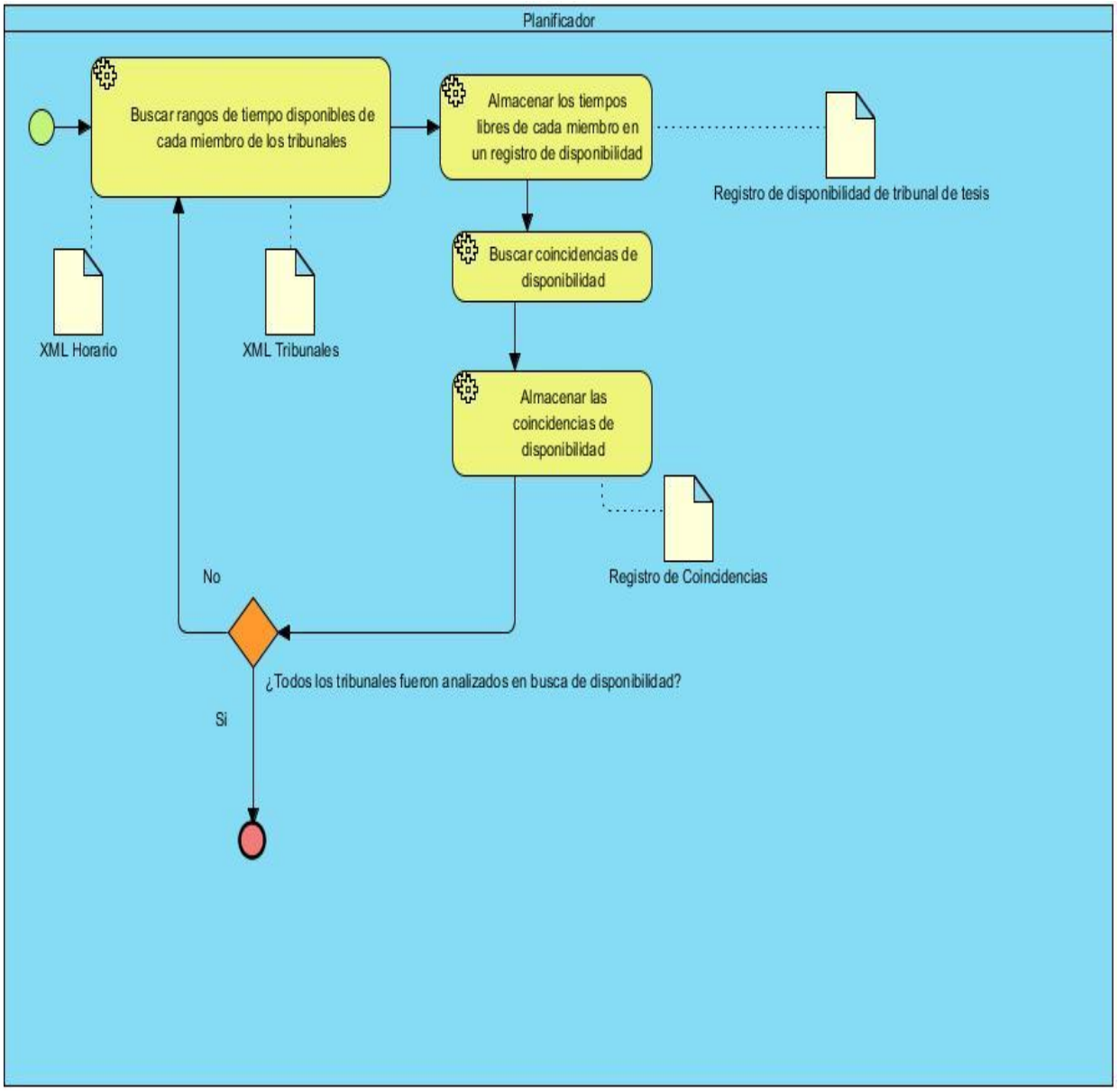

Figura 4: Buscar disponibilidad de los miembros del tribunal.

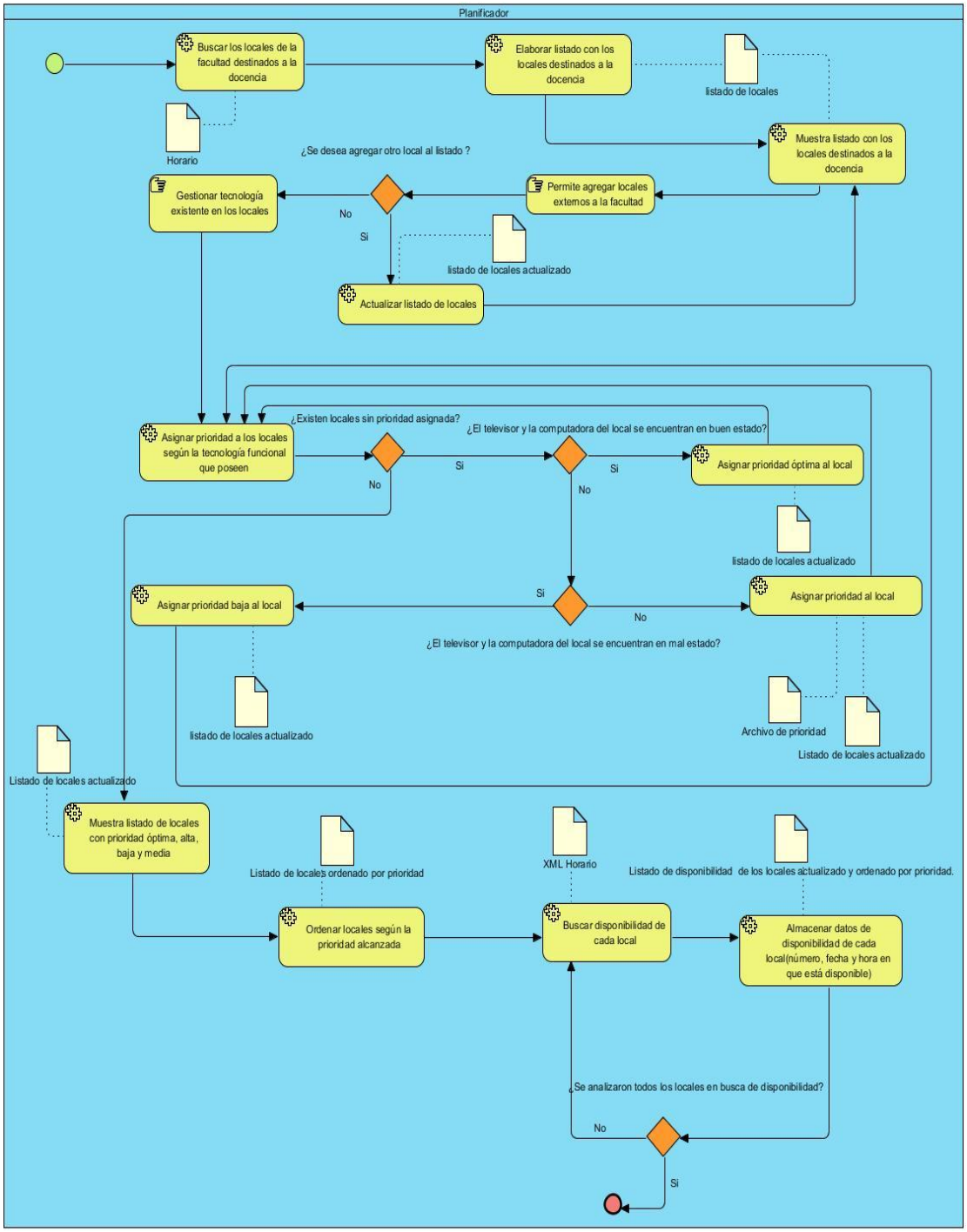

Figura 5: Buscar locales disponibles con los que cuenta la facultad para la discusión de tesis.

2.3 Especificación de los requisitos de *software*

Los requisitos constituyen un elemento fundamental en el proceso de producción de cualquier aplicación informática. Los requisitos se pueden clasificar en dos categorías, requisitos funcionales y requisitos no funcionales [37].

2.3.1 Requisitos funcionales

Los requisitos funcionales para la realización del sistema posibilitan definir el alcance del sistema, teniendo en cuenta las acciones a realizar y el intercambio de datos entre sus diferentes funciones [38].

Los requisitos funcionales del sistema propuesto son:

RF1. Gestionar Local.

- **RF1.1.** Registrar Local
- **RF1.2.** Buscar Local
- **RF1.3.** Modificar Local
- **RF1.4.** Cargar Fichero XML Horario

RF2. Cargar Fichero XML Tribunales

RF3. Cargar Configuración de Usuarios

RF4. Importar Fichero XML Horario

RF5. Importar Fichero Tribunales

RF6. Gestionar Usuarios

- **RF 6.1.**Registrar Usuario
- **RF6.2.** Buscar Usuario
- **RF6.3.** Modificar Usuario

RF6.4. Eliminar Usuario

RF7. Autenticar Usuario

RF8. Cambiar Contraseña

RF9. Buscar Tesis por Criterio

RF10. Ver Detalles de Tesis

RF11. Buscar Tesis

RF12. Modificar Avance de Tesis

RF14. Mostrar Tesis

RF15. Buscar Disponibilidad de Miembros de Tribunal

RF16. Realizar Planificación de Tesis

RF17. Exportar Planificación de Tesis

RF18. Importar Planificación de Tesis

RF19. Modificar Planificación de Tesis

2.3.2 Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Son las características que hacen al producto atractivo, usable, rápido y confiable [38].

RNF 1.Usabilidad:

- Facilidad de uso: el sistema debe tener una interfaz amigable que permita el uso sencillo de la aplicación, además de acceder a la información de manera efectiva y rápida.

- Se garantiza para el administrador final un adiestramiento básico en el uso de la aplicación.

RNF 2. Apariencia o interfaz

- Las ventanas del sistema: contienen claro y bien estructurados los datos, además de permitir la interpretación correcta de la información.
- Interfaz: Se emplearon colores cómodos a la vista, sin acumulación de imágenes u objetos que distraigan al usuario de su objetivo. Permite a los usuarios interactuar con el sistema de forma fácil.

RNF 3. Seguridad

- El sistema requiere de la autenticación como primera acción, con un nombre de usuario único y una contraseña que debe ser de conocimiento exclusivo de la persona que se autentica.

RNF 4. Restricciones de diseño

- Como Entorno Integrado de Desarrollo (IDE) se emplea NetBeans7.3
- El modelado UML se realiza con Visual Paradigm 8.0
- Se utiliza como metodología de desarrollo de *software* RUP.

RNF 5. Requisitos de licencia

- El proyecto utiliza la política de *software* libre donde todas las herramientas que utilizan son libres. Para la herramienta Visual Paradigm se utiliza la licencia que la UCI adquirió.

RNF 6. Requisitos hardware

- Para garantizar el correcto funcionamiento de la aplicación, se necesita una computadora con un procesador Pentium IV o superior y una memoria RAM (en inglés *Random Access Memory*) de 512 megabytes o más.

RNF 7. Requisitos de *software*

- Se necesita tener instalada el JDK (en inglés *Java Development Kit*) de Java en las máquinas clientes.

2.4 Modelo de casos de uso

El modelo de caso de uso está compuesto por actores, casos de uso y las relaciones entre ellos. Un caso de uso describe las acciones que brinda algún resultado a los actores del sistema.

2.4.1 Diagrama de paquetes del sistema

El siguiente diagrama de paquetes agrupa los casos de uso del sistema de la aplicación que se modela. Los casos de uso se encuentran agrupados en paquetes teniendo en cuenta la funcionalidad que representa cada uno [39].

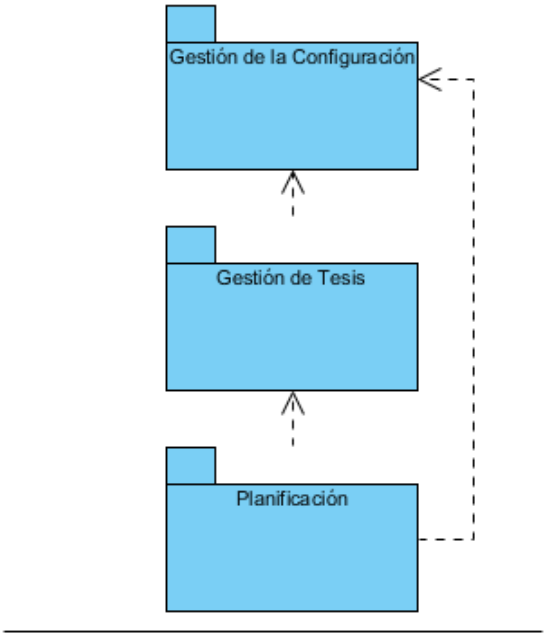

Figura 6: Diagrama de Paquetes.

A continuación se describen los paquetes utilizados:

Gestión de Configuración: contiene las funcionalidades de autenticar usuario, cargar la configuración de usuario y de ficheros XML, cambiar contraseña e importar manualmente el fichero XML de los tribunales.

Gestión de tesis: contiene las funcionalidades para la gestión de las tesis. Modificar, mostrar y buscar y ver detalles de las tesis.

Planificación: contiene todas las acciones relacionadas con la planificación. Planificar exposición de tesis y modificar planificación.

Los paquetes Gestión de tesis y Planificación dependen del paquete Gestión de la Configuración porque este paquete contiene la funcionalidad autenticar usuario. Las funcionalidades contenidas en los paquetes Gestión de tesis y Planificación, se ejecutan a partir de la autenticación. A su vez el paquete Planificación depende de Gestión de tesis

porque debe gestionarse primero la funcionalidad Modificar Estado de la Tesis del paquete Gestión de tesis para después ejecutar las funcionalidades contenidas en el paquete Planificación.

2.4.2 Diagrama de casos de uso del sistema

Un diagrama de un caso de uso ilustra un conjunto de casos de uso para un sistema, los actores y la relación entre los actores y los casos de usos. A continuación se muestran los diagramas de casos de uso del sistema que se modela.

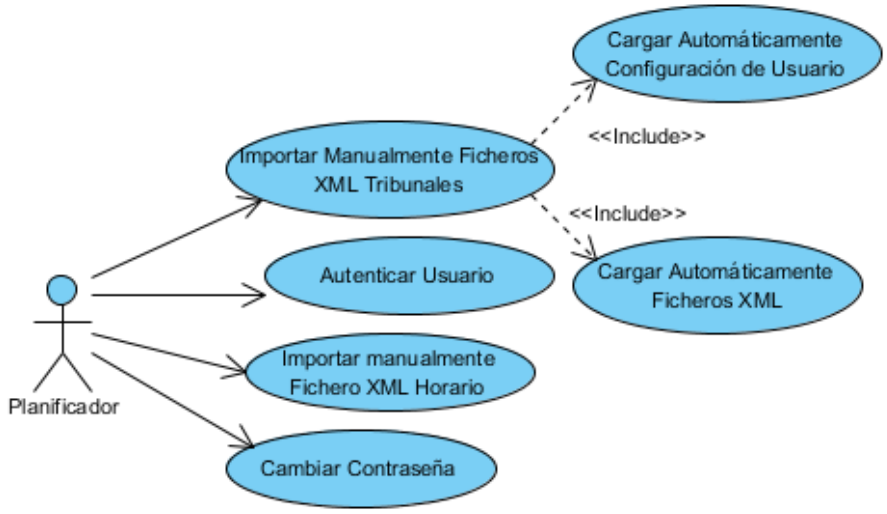

Figura 7: Diagrama de casos de uso del paquete Gestión de Configuración.

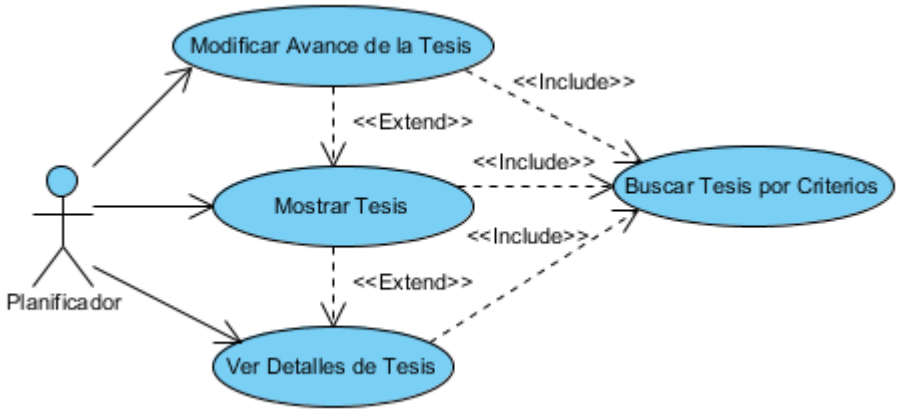

Figura 8: Diagrama de casos de uso del paquete Gestión de Tesis.

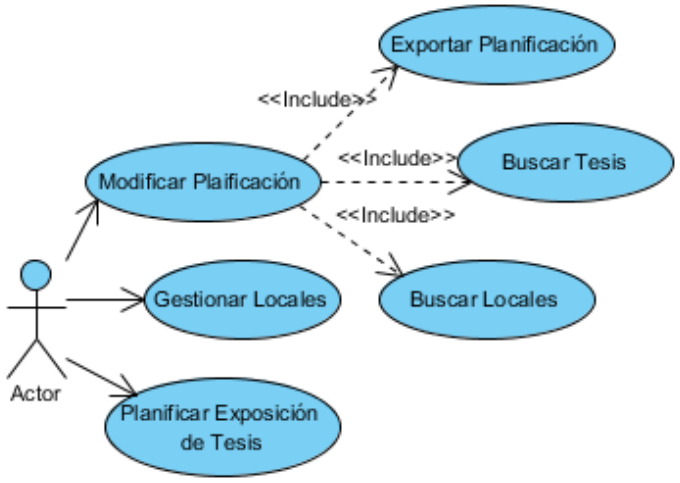

Figura 9: Diagrama de casos de uso del paquete Planificación.

2.4.2 Descripción de casos de uso

En la descripción de Caso de Uso del Sistema (CUS) se detalla paso a paso la interacción del sistema con el usuario. A continuación se describe el **CUS Planificar exposiciones de Tesis**. El resto de las descripciones se de los CUS se encuentran en el Anexo1.

Tabla 2: Descripción del CUS Planificar Exposición de Tesis.

Objetivo	Realizar la planificación de tesis de la Facultad 2.
Actores	Planificador
Resumen	El CUS se inicia cuando el planificador selecciona la opción “Planificar” en el menú principal. Selecciona los locales, las tesis a planificar y define los períodos de tiempo en que se va a desarrollar la planificación, así como el tiempo de duración de cada tipo de tesis. Devuelve el resultado de la planificación. Termina el CU.
Complejidad	Alta
Prioridad	Secundario

Precondiciones	Debe seleccionar los locales, las tesis y buscar la disponibilidad de los miembros del tribunal.
Pos condiciones	Planificación de tesis por local.
Flujo de eventos	
Flujo básico Planificar Exposición de Tesis	
Actor	**Sistema**
1. Selecciona la opción “Planificar Exposición de Tesis” en el menú principal.	2. Invoca al **CU “Buscar Locales”.**
	3. Brinda la opción de seleccionar los locales mediante el campo: • Seleccionado
4. Selecciona los locales a utilizar en la planificación.	
5. Oprime el botón “Siguiente”.	6. Valida que exista al menos un local seleccionado. Si no hay local seleccionado, ver el flujo alterno 6a. “Debe Seleccionar Local”.
	7. Invoca el **CU “Buscar Tesis”.**
	8. Brinda la opción de seleccionar la tesis mediante el campo: • Seleccionado
9. Selecciona las tesis a planificar.	
10. Oprime el botón “Siguiente”.	11. Valida que exista al menos una tesis seleccionada. Si no hay tesis seleccionada, ver el flujo alterno 11a. “Debe Seleccionar Tesis”.
	12. Muestra los datos a llenar para definir el período de duración de las pre-

	defensas y defensas • Inicio • Fin Muestra los datos a llenar: • Tiempo de receso. • Número máximo de sesiones por día. Muestra los datos a llenar para definir el tiempo de duración de las pre-defensas y defensas según el estado de las tesis: Estado Avanzado: • Horas • Minutos Estado Medio: • Horas • Minutos Estado Atrasado: • Horas • Minutos
13. Oprime el botón “Siguiente”.	14. Valida los datos introducidos Si faltan datos obligatorios, ver el flujo alterno 14a. “Faltan datos obligatorios”.
	14. Muestra la planificación resultante.
	15. Invoca al **CU Exportar Planificación.**
	16. Termina el CU.
Flujos alternos	
***Cancelar Planificación**	

Actor	Sistema
1. Oprime el botón “Cancelar”.	2. Limpia los campos y regresa al menú principal.
***Interfaz Anterior**	
Actor	**Sistema**
1. Oprime el botón ¨Anterior¨	2. Limpia los campos y muestra la interfaz anterior.
***Finalizar Planificación**	
Actor	**Sistema**
1. Oprime el botón ¨Finalizar¨	2. Almacena la planificación resultante y regresa al menú principal.
6a. Debe Seleccionar Local	
Actor	**Sistema**
	6a.1 Muestra el mensaje “Debe escoger algún local para continuar”, regresa al paso 4 del flujo básico.
11a. Debe Seleccionar Local	
Actor	**Sistema**
1.	Muestra el mensaje “Debe escoger alguna tesis para continuar”, regresa al paso 9 del flujo básico.
14a. Faltan datos obligatorios.	
Actor	**Sistema**
	1. Muestra el mensaje “No debe dejar ningún campo en blanco”, regresa al paso 12 del flujo básico.

Relaciones	**CU Incluidos**	Buscar Disponibilidad de Miembros de Tribunal. Buscar Locales. Buscar Tesis. Exportar Planificación.

Conclusiones del capítulo

En el capítulo se obtuvo una mejor comprensión de las de las características del sistema que se presenta, lo cual contribuye a una correcta documentación y diseño de la aplicación. Fueron modelados los procesos que intervienen en el negocio de la entidad obteniendo los requisitos funcionales y no funcionales para el desarrollo del sistema. La elaboración del diagrama de casos de uso del sistema permitió una mejor comprensión del problema por parte del cliente y los desarrolladores.

Capítulo 3: Diseño del Sistema

En el presente capítulo se define y describe la arquitectura propuesta para el diseño del sistema. Además se especifica el uso de patrones de diseño a utilizar para la correcta implementación del sistema. También realizan los diagramas de clases y de componente para guiar el proceso de desarrollo de software.

3.1 Arquitectura del sistema.

La arquitectura de *software* es una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes desde la visión del resto del sistema y las formas en que los componentes interactúa [39].

3.1.2 Estilo arquitectónico

La arquitectura del *software* guía al equipo de desarrollo a través del ciclo de vida del sistema. Mediante los estilos y patrones arquitectónicos se logra representar el sistema de forma estructurada y organizada [40].

Arquitectura en capas

Durante el diseño de la presente investigación se decidió utilizar una arquitectura en capas. El patrón pertenece al estilo arquitectónico de llamada y retorno donde cada capa proporciona servicios a la capa superior y se sirve de las prestaciones que le brinda la inferior. El sistema fue dividido en tres capas, cada capa puede tratarse de forma independiente, sin tener que conocer los detalles de las demás. La división del sistema en capas facilita el diseño modular, donde cada capa encapsula un aspecto concreto del sistema. Además permite la construcción de un sistema débilmente acoplado, minimizando las dependencias entre capas con el objetivo de sustituir la implementación de una capa sin afectar al resto del sistema.

3.1.3 Descripción de la arquitectura

A continuación se muestra una imagen del diseño de la arquitectura del sistema a implementar y a continuación se detalla el uso de las capas.

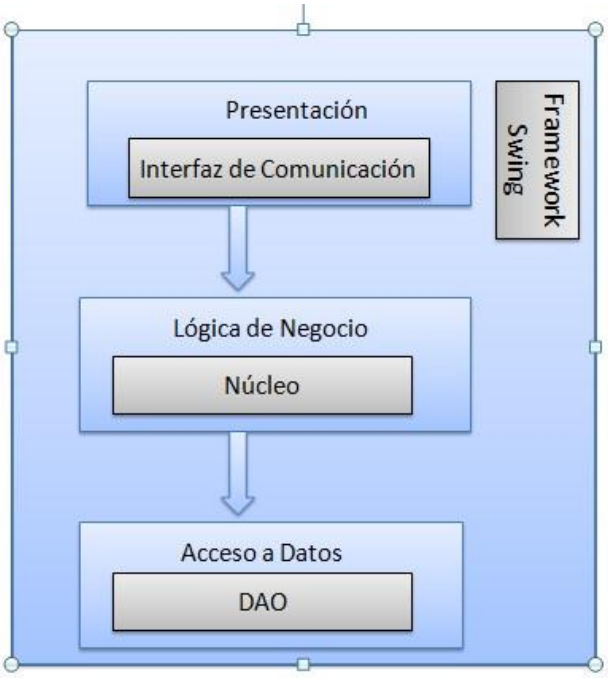

Figura 10: Estructura de la arquitectura en 3 capas.

Capa de presentación: capa donde se encuentran todas las interfaces gráficas que se utilizan para interactuar con el usuario y donde se visualiza el funcionamiento de las acciones que se llevan a cabo por parte de sistema. Su comunicación se establece con la capa lógica de negocio. Se utilizó en esta capa la librería *Swing* para personalizar el diseño de las interfaces, utilizando varios elementos que existen por defecto y que los componentes de *Swing* modifican para un mejor estilo en la interfaz.

Lógica de negocio: capa que agrupa las clases encargadas de implementar la lógica de negocio de la aplicación.

Acceso a Datos: capa que permite la interacción con el fichero que contiene los datos del usuario del sistema. Los DAOs (del inglés *Data Access Object*) encapsulan la persistencia de los objetos del negocio, proveen la persistencia de los objetos transitorios y las actualizaciones de los objetos existentes en la Base de Datos.

3.2 Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular [42]. Para el diseño de

la aplicación se hizo uso de los Patrones Generales de *Software* para Asignar Responsabilidades (GRASP). A continuación, se listan algunos de los utilizados durante el desarrollo de la aplicación.

Experto: consiste fundamentalmente en asignar una responsabilidad al experto en información, clase que cuenta con la información necesaria para cumplir la responsabilidad. Su uso se evidencia en las clases:

1. Clase Planificación: responsables de realizar la planificación ya que cuenta con la información necesaria para ejecutar esta acción.
2. Clase Local: responsable de asignar la prioridad a cada local ya que cuenta con la información necesaria para ejecutar la acción.

Alta Cohesión: caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Su uso se evidencia en las clases Tesis, Tribunales y Planificación donde todas sus responsabilidades están relacionadas pero cada una realiza una un número relativamente pequeño de responsabilidades.

Bajo Acoplamiento: cada clase en el sistema depende solo de las clases necesarias para su implementación. No existe una sobrecarga de dependencia entre las clases.

Los patrones de diseño del GoF(en inglés *Gang of Four*) son una herramienta fundamental para la implementación del sistema. A continuación se hace referencia a los que fueron utilizados.

Singleton: garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Su uso se evidencia en la capa de Negocio más específicamente en la clase Servicios Implementación la cual implementa a la Interface Servicios. En dicha clase se hace un llamado a una instancia de la clase principal Planificación, permitiendo que solo se cree un objeto de esta clase y que pueda ser utilizado en varios métodos.

```
public class Planificacion {

    /**
     * Uso del patrón singletón
     */
    private static Planificacion instance = null;

    public static Planificacion getInstance() {
        if (instance == null) {
            instance = new Planificacion();
        }
        return instance;
    }
}
```

Figura 11: Se muestra como se crea la instancia del método Planificación.

```
public class ServiciosImpl implements Servicios {
  private Planificacion planificacion;

   public ServiciosImpl() {
    this.planificacion = Planificacion.getInstance();
     }

   public LinkedList<Profesor> ObtenerListaPersona() {
       return planificacion.getListaPersona();
   }

   public void SetListaTribunales(LinkedList<Tribunales> tribunales) {
       planificacion.setListaTribunales(tribunales);
   }

}
```

Figura 12: Se muestra el llamado de la instancia y como se utiliza en varios métodos.

Iterator: Se evidencia tanto en los métodos contenidos en las interfaces de usuario como en las clases del dominio. Se utiliza para acceder al contenido de una colección de datos.

```
public void initDisponibilidad(int dias) {
    this.matriz_disponibilidadLocales = new int[5][dias];
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < dias; j++) {
            matriz_disponibilidadLocales[i][j] = Planificacion.LIBRE;
        }
    }
}
```

Figura 13: Uso del patrón en el recorrido del listado de miembros del tribunal en busca de la disponibilidad de los profesores miembros de dicho tribunal existente.

3.3 Diagrama de paquetes del diseño

El siguiente diagrama de paquetes del diseño tiene el objetivo de estructurar una visión general del sistema. Las clases del sistema se encuentran agrupadas en cada paquete según el objetivo con que fueron creadas. La estructura organizativa de los paquetes responde a la arquitectura definida para la implementación del sistema. Se definió un color para diferenciar la capa de la arquitectura a la que pertenecen. Los paquetes de color amarillo representan la capa de presentación, los de color azul la capa de negocio y los de color verde la capa de acceso a datos.

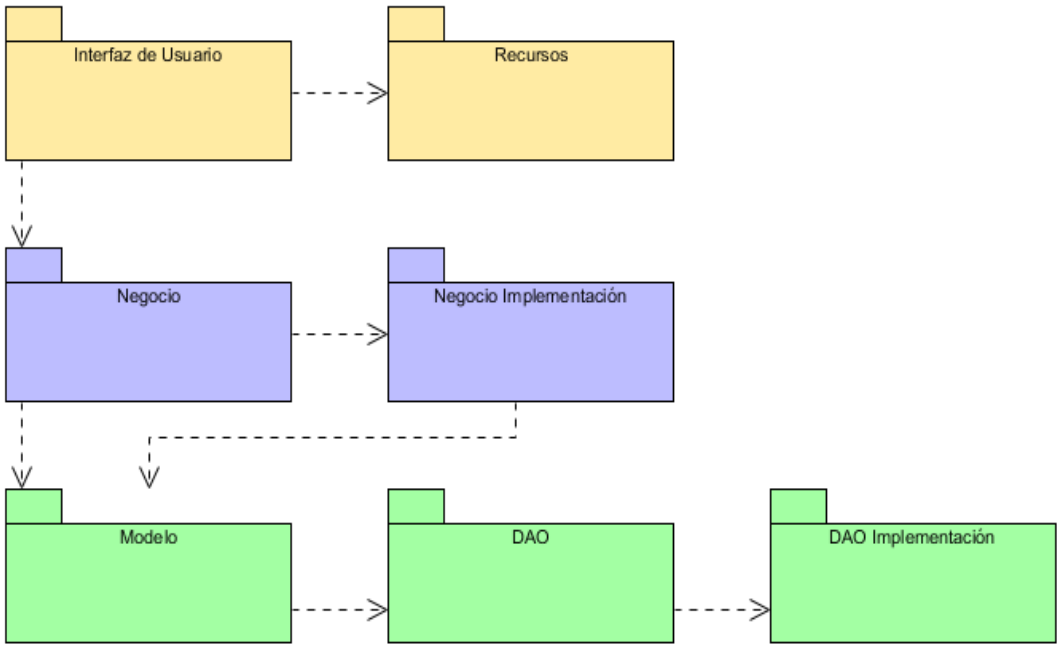

Figura 14: Estructura de los paquetes del diseño.

La estructura organizativa de los paquetes es la siguiente:

Paquete Interfaz de Usuario: contiene todas las interfaces de usuario.

Paquete Recursos: contiene el mapa de configuración, encargado de gestionar las imágenes.

Paquete Negocio: contiene la interfaz Servicio, encargada de la comunicación entre las capas presentación y negocio.

Paquete Negocio Implementación: contiene la implementación de la interfaz Servicio.

Paquete Modelo: contiene todas las clases correspondientes al negocio del sistema.

Paquete DAO: contiene la interfaz usuarioDao, encargada de la comunicación entre las capas negocio y acceso a dato.

Paquete DAO Implementación: contiene la implementación de la interfaz usuarioDao.

3.4 Diagrama de clases del diseño

Los diagramas de clases del diseño se encuentran enfocados en la arquitectura definida para el sistema. Los diagramas se modelaron de forma similar porque intervienen las mismas clases contenidas en los paquetes que forman la arquitectura. Por tal motivo en cada diagrama solo varían las clases que son interfaces de usuario y los métodos a implementar sobre la interfaz. El siguiente diagrama de clases del diseño correspondientes al CUS Planificar Exposición de Tesis:

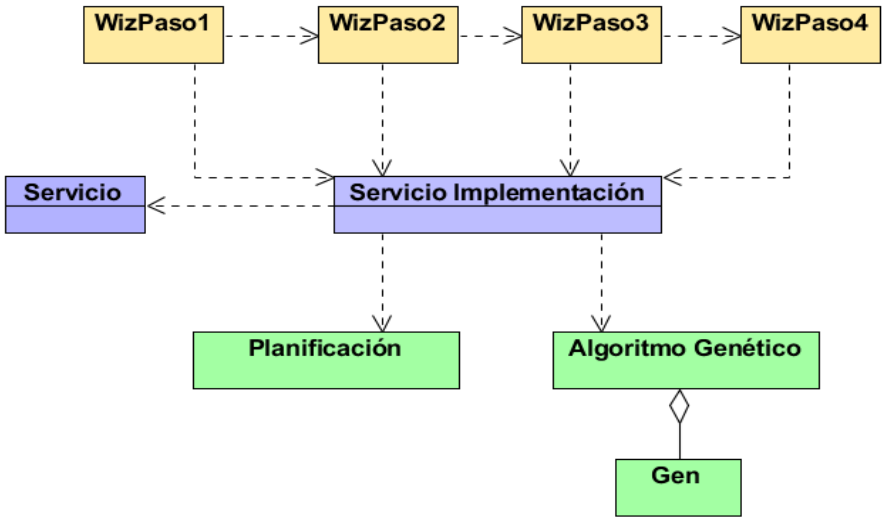

Figura 15: Diagrama de clase del diseño del CUS Planificar Exposición de Tesis.

A continuación se explica la relación que se establece en cada una de las clases del diseño.

WizPaso1: Se seleccionan los locales para planificar.

WizPaso2: Se seleccionan las tesis para planificar.

WizPaso 3: Se definen los períodos de tiempos.

WizPaso 4: Se encarga de mostrar la planificación.

Servicio: Se utiliza para conectar los *wizsard* con las clases que necesita.

ServicioImplementación: Implementa la interfaz servicio.

Planificación: Contiene los listados con la tesis y los locales.

Conclusiones del capítulo

En el capítulo se definió la arquitectura en tres capas para ser utilizada como base fundamental en la implementación del sistema. La modelación de los diagramas de paquetes y clases del diseño permitió organizar estructuralmente el sistema.

.

Capítulo 4: Implementación y Prueba

En el presente capítulo se realiza el diagrama de componentes evidenciando la integración de cada uno. Se diseñan los parámetros del algoritmo genético para su posterior implementación utilizando el *framework* JGAP. También se realizan las pruebas de caja negra para verificar que el *software* cumple con los requisitos establecidos.

4.1 Diagrama de componentes

El diagrama de componentes que a continuación se muestra refleja la integración de cada uno de los componentes. El componente Planificación.jar contiene todos los paquetes que conforman el sistema. Los componentes Tribunal.XML y Horario.XML son los archivos de entrada para realizar la planificación mientras que Planificación Final.XML es el archivo de salida una vez culminada la planificación. El sistema carga el componente Swing library para mejorar el diseño de las interfaces y el *framework* JGAP para la implementación del Algoritmo Genético.

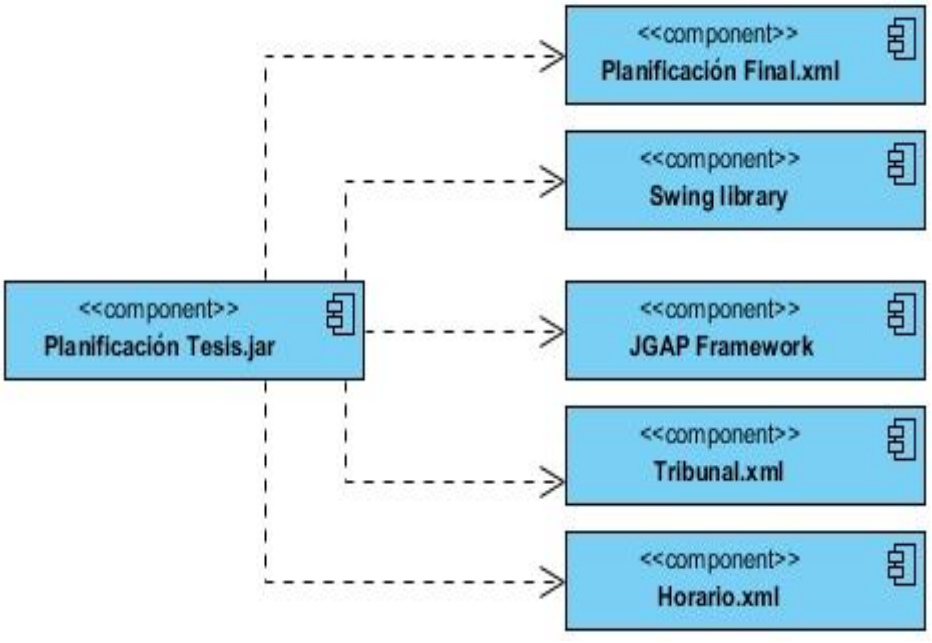

Figura 16: Diagrama de Componentes.

4.2 Modelo de optimización

Como quedó planteado en el epígrafe 1.1, la solución del problema puede ser modelada como un problema de optimización combinatoria. Para la realización del modelo de optimización es necesario determinar el conjunto de restricciones que intervienen en el problema. Además se necesita construir la función objetivo, para evaluar el desempeño de las decisiones tomadas, sujetas a las restricciones que a continuación se definirán.

4.2.1 Restricciones

El problema de planificación de un horario para las exposiciones de los trabajos de diploma está sometido a varias restricciones con diferentes niveles de importancia. Algunas son inviolables y ningún horario será aceptado si no las cumple considerándolas como restricciones fuertes. El cumplimiento de otro conjunto de restricciones es conveniente o deseable y servirán para medir la calidad del horario obtenido, denominadas restricciones débiles.

Restricciones Fuertes

- Un profesor miembro de un tribunal no puede estar asignado a una discusión de tesis si debe impartir clases en el mismo horario.
- Un tribunal no puede ser asignado a más de una discusión de tesis en el mismo horario.
- Un local no debe ser asignado a más de una discusión de tesis en el mismo horario.

Restricciones Débiles

- Todas las tesis pertenecientes a un tribunal deben ser planificadas en el mismo día para evitar volver a reunirse.
- Todas las tesis de un tribunal deben discutirse en un mismo local.
- Tener en cuenta que se deben poner primero las tesis que presenten estado Avanzado.

- Priorizar en la planificación los locales donde la tecnología sea óptima (en perfecto estado televisor y computadora).

4.2.2 Función objetivo

La función objetivo se utiliza para seleccionar los individuo de la población que van a cruzarse, determinando los más adaptados para continuar con la evolución de la población. El problema de asignación de horarios para la exposición de pre-defensa y defensa de los trabajos de diploma queda formulado de la siguiente manera:

- Encontrar un horario h que pertenezca al conjunto de posibles horarios H que cumpla con todas las restricciones fuertes y con el mayor número de restricciones débiles.

$$\text{Maximizar } F(h) = \begin{cases} 1 + \sum_{i=1}^{M} D_i(h) & \text{Si} \quad \sum_{i=1}^{N} F_i(h) = N \\ 0 & \text{Si} \quad \sum_{i=1}^{N} F_i(h) \neq N \end{cases}$$

$$F_i(h) = \begin{cases} 1 \text{ si cumple con la restricción fuerte } i \\ 0 \text{ si no cumple con la restricción fuerte } i \end{cases}$$

$$D_i(h) = \begin{cases} 1 \text{ si cumple con la restricción débil } i \\ 0 \text{ si no cumple con la restricción débil } i \end{cases}$$

- Donde F(h)es la función objetivo, N es la cantidad de restricciones fuertes y M es la cantidad de restricciones débiles.
- Di(h) es la evaluación de la restricción débil i para el horario h.
- Fi(h) es la evaluación de la restricción fuerte i para el horario h.

4.3 Diseño e implementación del Algoritmo Genético

Para el diseño del Algoritmo Genético (AG) se procede a seleccionar en cada paso los operadores a aplicar según las condiciones del problema. Para comprender como fue estructurado el diseño del algoritmo es necesario conocer los conceptos fundamentales a los que se hace referencia. La siguiente tabla resume la terminología que deben tenerse en cuenta:

Tabla 3: Conceptos Específicos del algoritmo genético.

Población	Conjunto de horarios.
Cromosoma	Está compuesto por la ubicación de todas las tesis ya planificadas y es considerado como un horario.
Gen	Contiene la información de la ubicación de una tesis: día, turno, posición del turno, medio turno, identificador del local, identificador de la tesis.
Alelo	Valor asociado a cada elemento de la ubicación de la tesis.

El diseño está compuesto por la codificación de los parámetros a tratar, el tamaño de la población inicial, la selección de los mejores individuos (cromosomas) y su posterior cruzamiento. Además se incluye la operación de mutación, seleccionando los individuos que realizan el proceso. Una vez concluido el diseño del algoritmo se procede a la implementación, utilizando el *framework* JGAP. El *framework* consta de una estructura de clases e interfaces que implementan el funcionamiento de los operadores genéticos.

4.3.1 Codificación

Se propone para codificar el problema el tipo de codificación entera, debido a que es muy utilizada en problemas de optimización combinatoria (la búsqueda de la mejor permutación de elementos sujetos a determinadas restricciones). Un cromosoma es definido como una solución del problema y está compuesto por N cantidad de genes en dependencia de la cantidad de tesis a planificar. Los genes están compuestos a su vez por alelos que representan la ubicación para determinada tesis. Los primeros cinco alelos de un gen representan la ubicación que puede tener la tesis ubicada en la sexta posición del gen.

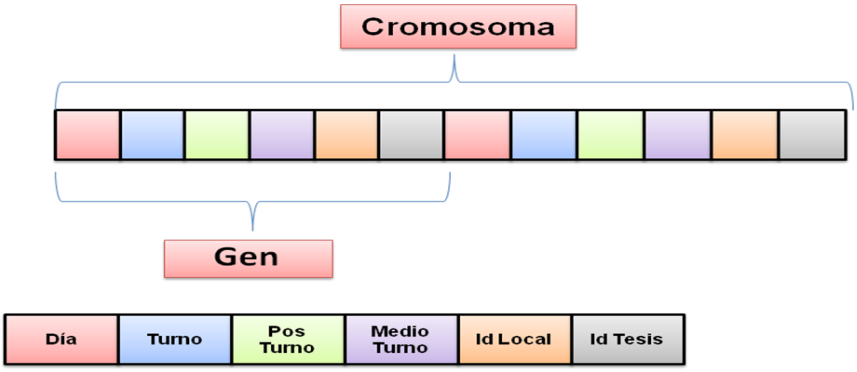

Figura 17: Codificación de los Genes.

La codificación de los genes se describe a continuación:

Día: Es un número entero en el intervalo (0…N), que representa el día donde inicia la planificación.

Turno: Es un número entero en el intervalo (0…4), la hora de inicio de cada turno se corresponde con la definida en el horario docente. La cantidad de tunos representa los destinados a las exposiciones en un día.

Pos Turno: Indica si la exposición de la tesis está en la primera mitad del turno tomando valor 0 o en la segunda mitad con valor 1. Para determinar la mitad del turno la duración del mismo fue dividida en dos, asignando 45 minutos a cada mitad.

Medio Turno: Indica si la exposición acaba en los primeros 45 minutos del turno o no, tomando valores (0,1) respectivamente.

IdLocal: Es un número entero en el intervalo (1…N), que representa el identificador del local donde se realiza la planificación.

IdTesis: Es un número entero en el intervalo (1…N), que representa el identificador de la tesis a planificar.

Haciendo uso del *framework* JGAP la codificación entera fue definida como se muestra en la siguiente figura:

```
public class Gen extends BaseGene {

    /** Dias desde el dia de inicio elegido. */
    private int dia;
    /** Turno dentro del día, 1er, 2do, etc. */
    private int turno;
    /** Indica si la tesis esta en el inicio o mediado del turno. */
    private int posTurno;
    /** Indica donde acaba la tesis, en los primeros 45 mins o no. */
    private boolean medioTurno;
    /** Identificador del local. */
    private String idLocal;
    /** Identificador de la tesis. */
    private int idTesis;
    //private Ubicacion ub;
```

Figura 18: Codificación

4.3.2 Población

Para generar la primera población se emplea un algoritmo voraz [7]permitiendo que cumpla con las restricciones fuertes que fueron definidas. Posteriormente se crean 100 copias exactas del individuo inicial, las cuales van evolucionando por cada iteración del algoritmo. Cada individuo es un horario y representa una posible solución al problema. A continuación se muestra una imagen con el tamaño de la población inicial.

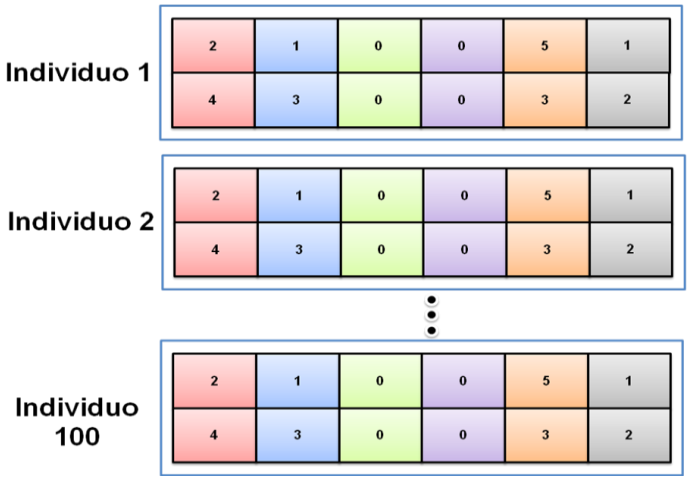

Figura 19: Población inicial.

[7] Algoritmo voraz: Es un algoritmo que para resolver un determinado problema, sigue una heurística consistente en elegir la opción óptima en cada paso local con la esperanza de llegar a una solución general óptima.

El *framework* JGAP permite realizar copias de un individuo previamente creado. En este caso se definió un total de 100 individuos los cuales representan el tamaño de la población.

```
Chromosome chrom = new Chromosome(conf, g);
conf.setSampleChromosome(chrom);
conf.setPopulationSize(100);
```

Figura 20: Creación de la población inicial.

4.3.3 Operador de selección

El operador de selección empleado es la técnica del Torneo probabilístico. Permite que los mejores individuos sean escogidos con una mayor probabilidad, pero al mismo tiempo admite a los peores individuos. Este proceso ayuda a mantener la diversidad de la población. Para aplicar la técnica del torneo es necesario definir el tamaño del torneo, en este caso se seleccionaron dos individuos.

Para cada iteración se evalúan los individuos por pares de dos haciendo uso de la función objetivo, almacenado en una lista el que adquiera mejor valor. En la primera iteración del algoritmo todos los individuos obtienen el mismo valor de función objetivo por ser iguales. Para este caso se almacena en la lista el primer individuo de los que van a realizar el torneo. Este proceso se realiza tantas veces como individuos tenga la población. La siguiente imagen muestra los individuos que fueron seleccionados al aplicar la técnica del Torneo.

Población	Tamaño Lista	Individuos	Función Objetivo
100	0	Individuo 1 Individuo 2	50 50
100	1	Individuo 2 Individuo 100	50 50
. .	. .	. .	. .
100	100	Individuo 100 Individuo 1	50 50

	Pos 0	Pos 1	...	Pos 99
Lista de seleccionados	Individuo 1	Individuo 2	...	Individuo 100

Figura 21: Proceso de Selección.

Haciendo uso del framework JGAP se muestra el fragmento de código donde se utiliza la clase NaturalSelector para seleccionar como operador de selección la técnica del Torneo probabilístico. Con una probabilidad de selección de 80% y especificando que los individuos serán evaluados por pares.

```
conf.addNaturalSelector(new TournamentSelector(this.conf, 2, 0.8), false);
```

Figura 22: Operador de selección.

4.3.4 Operador de cruzamiento

El operador de cruzamiento seleccionado es el operador de cruce en un punto, manteniendo la tesis y cruzando la ubicación correspondiente a la misma. La probabilidad de cruzamiento es de 80%. Para decidir qué individuos pueden cruzarse se asigna un número aleatorio por individuo y se compara con la probabilidad de cruzamiento. Los individuos que su número aleatorio sea menor o igual que la probabilidad de cruzamiento podrán cruzarse. La siguiente figura indica que solo se cruzan las ubicaciones de las tesis del individuo uno.

Individuos	Número Aleatorio	Probabilidad
Individuo 1	0.25	0.8
Individuo 2	0.86	0.8
. . .	. . .	. . .
Individuo 100	0.74	0.8

Figura 23: Selección de los individuos a cruzar.

El proceso de cruzamiento se realiza en los individuo por separado, manteniendo la tesis y cruzando la ubicación que fue asignada a la misma. Para realizar el cruzamiento entre las tesis se decidió seleccionar aleatoriamente cuales podrían cruzar. El proceso de selección coincide con el anteriormente descrito para la selección de los individuos a cruzar. La siguiente figura ilustra cómo se realiza el proceso de cruzamiento para la ubicación de dos tesis seleccionadas:

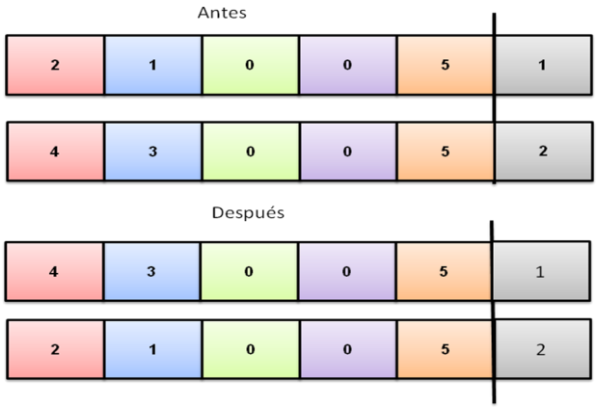

Figura 24: Proceso de cruzamiento.

Haciendo uso del *framework* JGAP se muestra el fragmento del código donde se configura el operador de cruzamiento y se indica la probabilidad de cruzamiento.

```
conf.addGeneticOperator(new CrossoverOperator(this.conf, 0.80f));
```

Figura 25: Operador de cruzamiento.

4.3.5 Operador de Mutación

El operador de mutación seleccionado fue mutación por intercambio (en inglés *SWAP Mutation*), con una probabilidad de mutación de 10%. La siguiente figura corresponde al proceso de seleccionar los individuos que van a realizar el proceso de mutación. Cada individuo recibe un número aleatorio y los que presenten un número aleatorio menor que la probabilidad de cruzamiento pueden realizar la mutación. En este caso fue seleccionado el Individuo 1.

Individuos	Número Aleatorio	Probabilidad
Individuo 1	0.15	0.2
Individuo 2	0.85	0.2
. .	. .	. .

Individuo 100	0.67	0.2

Figura 26: Selección de los individuos a mutar.

Con el fin de realizar el proceso de mutación fue creada una lista con ubicaciones disponibles, en caso de existir. La mutación es realizada para cada gen, asignado una nueva ubicación definida en la lista de ubicaciones. Para asignar una ubicación a un gen se debe verificar que el alelo tesis permite la ubicación que le será asignada, teniendo en cuenta en este proceso las restricciones fuertes. En la siguiente figura se muestra el proceso de mutación que se ejecuta una vez seleccionado el individuo a mutar, el cual contiene para este caso una sola tesis planificada.

Figura 27: Proceso de mutación.

Haciendo uso del *framework* JGAP se muestra el fragmento del código donde se configura el operador de mutación y es asignada la probabilidad de cruzamiento.

```
conf.addGeneticOperator(new MutationOperator(this.conf, 10));
```

Figura 28: Operador de mutación.

4.3.6 Evolución

El método *evolve*() del *framework* JGAP es el encargado de realizar los procesos de selección, cruzamiento y mutación entre los individuos. Devuelve la población que mejor se

encuentre adaptada al medio. Fue definida como condición de parada la no mejoría de la función en tres iteraciones.

```
while (mejora) {
    poblacion.evolve();
    double tmpValr = poblacion.getFittestChromosome().getFitnessValue();
    if (tmpValr <= valor) {
        contEvol++;
    } else {
        valor = tmpValr;
        contEvol = 0;
    }
    if (contEvol == 3) {
        mejora = false;
    }
}
return poblacion.getFittestChromosome();
```

Figura 29: Método evolución

4.4 Pruebas

Las pruebas de *software* son utilizadas para identificar posibles fallos de implementación, calidad o usabilidad en un sistema informático.

4.4.1 Método de prueba

Prueba de Caja Negra: Pruebas que se llevan a cabo sobre la interfaz del *software*, se centra principalmente en los requisitos funcionales del *software*. Pretenden demostrar que las funciones del *software* son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto. Las pruebas de caja negra verifican las especificaciones funcionales y no consideran la estructura interna del programa, además se hacen sin el conocimiento interno del producto y no se validan funciones ocultas.

4.4.2 Diseño de casos de pruebas

El diseño de los casos de pruebas verifica si el producto satisface los requerimientos del usuario y si se comporta como se desea. Para comprobar el funcionamiento del sistema se realizó el caso de prueba Planificar Exposición de Tesis:

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1: Realizar Planificación	Se realiza una planificación satisfactoriamente.	Realiza la Planificación. Ver DCP_Exportar Ficheros XML	1. Seleccionar los Locales. 2. Seleccionar las Tesis. 3. Seleccionar la fecha inicial. 4. Seleccionar la fecha final. 5. Especificar la cantidad de sesiones. 6. Definir el tiempo de receso. 7. Definir Tiempo para tesis con estado avanzado. 8. Definir Tiempo para tesis con estado medio. 9. Definir Tiempo para tesis con estado atrasado. 10. Selecciona la opción "Finalizar".
EC 1.2: Cancelar Planificación	Se cancelan las operaciones que se estén realizando.	Cancela la operación que se esté realizando.	1. Seleccionar la opción “Cancelar”.
EC 1.3: Datos de Selección Incorrectos.	No permite realizar la planificación.	No permite ejecutar la siguiente acción.	1. Seleccionar los Locales. 2. Seleccionar las Tesis. 3. Seleccionar la fecha inicial. 4. Seleccionar la fecha final. 5. Especificar la cantidad de sesiones. 6. Definir el tiempo de receso. 7. Definir Tiempo para tesis con estado avanzado. 8. Definir Tiempo para tesis con estado medio. 9. Definir Tiempo para tesis con

			estado atrasado. 10. Selecciona la opción "Finalizar".
EC 1.4: Faltan Datos Obligatorios	No permite realizar la planificación.	No permite ejecutar la siguiente acción. Muestra un mensaje especificando que la fecha de fin debe ser mayor que la fecha de inicio.	1. Seleccionar los Locales. 2. Seleccionar las Tesis. 3. Seleccionar la fecha inicial. 4. Seleccionar la fecha final. 5. Especificar la cantidad de sesiones. 6. Definir el tiempo de receso. 7. Definir Tiempo para tesis con estado avanzado. 8. Definir Tiempo para tesis con estado medio. 9. Definir Tiempo para tesis con estado atrasado. 10. Selecciona la opción "Finalizar".

4.4.3 Resultados de las pruebas

En la primera etapa de pruebas se le aplicaron 3 iteraciones de pruebas a los casos de uso Planificar Exposiciones de Tesis y como resultado se detectaron 20 no conformidades en la primera ,4 en la segunda y 2 en la última.

Primera Etapa

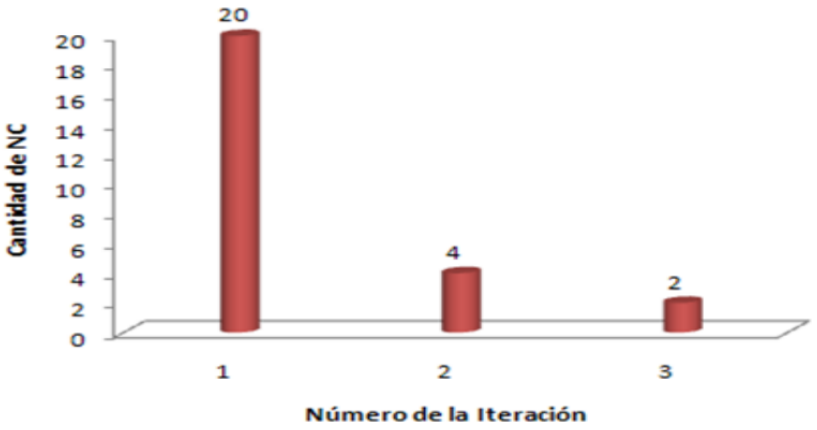

Figura 30: Resultado de las pruebas por iteración.

En la segunda etapa se probaron todas las funcionalidades del sistema, los resultados obtenidos fueron: 7 no conformidades para la primera iteración, 11 en la segunda y 1 en la última iteración.

Segunda Etapa

Figura 31: Resultado de las pruebas por iteración.

Conclusiones del capítulo

Con el diseño del diagrama de componente se estableció la relación que existe entre los principales componentes del sistema. El diseño e implementación del algoritmo genético utilizando las funcionalidades que brinda el *framework* JGAP, permitió la confección

automática de un horario para las exposiciones de los trabajos de diploma. Con la aplicación de pruebas de caja negra se verificó que el sistema cumple con todos los requisitos funcionales.

Conclusiones Generales

La realización de este trabajo responde a la necesidad de de buscar una solución al problema planteado en la Facultad 2 de la Universidad de las Ciencias Informáticas. Una vez culminada la investigación se da cumplimiento al objetivo general planteado, evidenciándose las siguientes conclusiones generales:

- ✓ La modelación del problema de asignación de horarios para las exposiciones de los trabajos de diploma permitió definir las restricciones fuertes y débiles que deben tenerse en cuenta para realizar la planificación de los trabajos de diploma.
- ✓ El diseño del Algoritmo Genético contribuyó a la generación de un horario automático para darle solución al problema de planificación de horarios.
- ✓ Fue implementado un sistema automatizado capaz de darle solución al objetivo de la presente investigación.
- ✓ Fue validado el resultado a través de casos de pruebas comprobando que la planificación propuesta por el sistema informático es adecuada con los datos especificados.

Recomendaciones

1. Adicionar nuevas restricciones donde se tengan en cuenta las afectaciones que puedan presentar los profesores.

2. Una futura integración con el sistema de confección de tribunales de tesis en la Facultad 2.

3. La implementación de una aplicación web que se encargue de mostrar a los estudiantes de la facultad el horario final de pre-defensas y defensas de los trabajos de diploma.

4. Extender la utilización del software a todas las facultades de la universidad

.

Referencias Bilbliográficas

1. Solis, P.D., Torres; Eimyn, Rizo, *El horizonte de la planificación: vinculación del proceso presupuestario con la planificación a largo plazo.* Contribuciones a la Económia., junio 2009.
2. MARTINEZ , R. *Importancia de la Planificación en la Organización Turistística 2da parte.* Sábado, 28 de noviembre del 2009 [cited; Available from: http://rincondeluniversitario.blogspot.com/2009/11/importancia-de-la-planificacion-en-la.html.
3. QU, R., *A survey of search methodologies and automated system development for examination timetabling.* Journal of scheduling, 2009. **12**(1): p. 55-89.
4. Mohammed Azmir, A.-B., Ahamad Tajudin Khader, *University Course Timetabling Using a Hybrid Harmony Search Metaheuristic Algorithm*, in *IEEE Transactions on Systems, Man, and Cybernetics.* 2012.
5. Lewis, R., *"A survey of metaheuristic-based techniques for university timetabling problems".* OR Spectrum, 2008. **30**(1): p. 167-190.
6. *Reglamento Trabajo Docente y metodológico*, M.D.L.E. Superior, Editor. 2010.
7. ALFONSO, H., *Resolviendo problemas de optimización con técnicas inteligentes.* En XIII Workshop de Investigadores en Ciencias de la Computación, 2011.
8. ETÁN OLIVÁN, J., *BOJ DEL VAL.* Eva. Programación lineal, 2012.
9. SARMIENTO-LEPESQUEUR, A., *Programación y asignación de horarios de clases universitarias: un enfoque de programación entera.*
10. PIEDRAHITA, E., *Programación lineal.* Revista Universidad EAFIT, 2012. **1**(1): p. 41-62.
11. PASTOR MORENO, R., *Metalgoritmo de optimización combinatoria mediante la exploración de grafos.* 2012.
12. MARTÍ, R., *Algoritmos heurísticos en optimización combinatoria.*, in *Departament d'Estadística i Investigació Operativa,Facultat de Matemàtiques*. 2005, Universitat de Valèencia.

13. MARTI, R., *Procedimientos metaheurısticos en optimización combinatoria.* Matemátiques, 2003. **1**(1): p. 3-62.
14. GENDREAU, M.P., Jean-Yves., *Metaheuristics in combinatorial optimization.* Annals of Operations Research, 2005. **140**(1): p. 189-213.
15. OSMAN, I.H.K., James P., *Meta-heuristics: theory and applications.* 1996: Springer.
16. WREN, A.S., *timetabling and rostering—a special relationship?* En Practice and theory of automated timetabling., 1996(Springer Berlin Heidelberg): p. 46-75.
17. Schaerf, A., *A survey of automated timetabling.* Artificial Intelligence Review archive, April 1999. **13**(2): p. 87-127.
18. MEJÍA CABALLERO, J.M., *Asignación de horarios de clases universitarias mediante algoritmos evolutivos*, in *Ingenieria Industrial*. 2009, Universidad del Norte.
19. *Generador de Horarios para Centros de Enseñanza*. [cited 2013 31 enero]; Available from: http://www.penalara.com.
20. *Timetab*.[cited 2009 21 febrero]; Available from: http://www.abcdatos.com/programas/programa/l1382.html.
21. *TimeTabler Lantiv.* [cited 2013 20 febrero]; Available from: http://www.lantiv.com.
22. *TimeTabler Mimosa Software*. [cited 2013 20 febrero]; Available from: http://www.mimosasoftware.com.
23. Faculty of Engineering, C.a.C.I., School of Computing, *Computing Research*. 2002.
24. Hocaoglu, C.H.A.a.G., *A TABU SEARCH ALGORITHM TO SOLVE A COURSE TIMETABLING PROBLEM.* ,Journal of Mathematics and Statistics, 2007. **36(1)**.
25. Moray . Nandhini , D.S.K., *A Survey of Simulated Annealing Methodology for University Course Timetabling* International Journal of Recent Trends in Engineering, May 2009. **1**(2).
26. Grosan C., y.A.A., *Intelligent Systems: A Modern Approach. Intelligent Systems Reference Library*. 2011, Springer.
27. Mejia Caballero, I.J.M., *Asignacion de horarios de clases universitarias mediante algoritmos evolutivos* in *Division de Postgrados e investigaciones en ingeniería Maestria en ingeniería Industrial*. Junio del 2008, Universidad del norte Barranquilla-Atlantico.
28. Rushil Raghavjee, Nelishia Pillay,. *A Comparison of Genetic Algorithms and Genetic Programming in Solving School Timetabling Problems.* IEEE Press, 2012.

29. Holland, H.J., *Adaptation in Natural and Artificial Systems*. 1975: University of Michigan Press. 211.
30. Patón, D.E.F.-M., *Alarcos*. 2006.
31. Kamilosol1. *Metodologías Tradicionales vs. Metodologías Ágiles*. 2007 [cited 2007 Julio 8].
32. Omg.org. *Introduction to OMG UML*. 2009 [cited 2 Abril, 2010]; Available from: http://www.omg.org/gettingstarted/what_is_uml.htm.
33. (BPMI), B.P.M.I., *Business Process Modeling Notation (BPMN)*
Vol. 1. 2004.
34. Informática, I.N.d.E.e., *Herramientas Case.El mejor soporte para el proceso de desarrollo de software*. Colección Cultura Informática. 2010.
35. Ordax, J.M., *Java Básico.Introducción a Java*. 2004.
36. CHAMBA, L.A. *JGAP-y-Algoritmos-Geneticos*. 2012 [cited; Available from: http://es.scribd.com/jimaca%C3%B1a/d/59637214-JGAP-y-Algoritmos-Geneticos.
37. Méndez, G., *Requisitos Funcionales*. 2008: Madrid,España.
38. PRESSMAN, R.S., *Ingeniería del software: Un enfoque práctico.*
39. Christine Hofmeister, R.N.y.D., *Describing Software Architecture with UML*. IEEE Computer Society Press ed. 1999.
40. Reynoso, C.B., *Introducción a la Arquitectura deSoftware.* Marzo de 2004.
41. Ivar Jacobson, G.B., James Rumbaugh.,. *El proceso unificado de desarrollo de software*. 2000.
42. Patrones de Diseño. Prieto, Félix.2009.

Bibliografía

1. Solis, P.D., Torres; Eimyn, Rizo, *El horizonte de la planificación: vinculación del proceso presupuestario con la planificación a largo plazo.* Contribuciones a la Económia., junio 2009.
2. MARTINEZ , R. *Importancia de la Planificación en la Organización Turistística 2da parte.*\Sábado, 28 de noviembre del 2009 [cited; Available from: http://rincondeluniversitario.blogspot.com/2009/11/importancia-de-la-planificacion-en-la.html.
3. QU, R., *A survey of search methodologies and automated system development for examination timetabling.* Journal of scheduling, 2009. **12**(1): p. 55-89.
4. Mohammed Azmir, A.-B., Ahamad Tajudin Khader, *University Course Timetabling Using a Hybrid Harmony Search Metaheuristic Algorithm*, in *IEEE Transactions on Systems, Man, and Cybernetics.* 2012.
5. Lewis, R., *"A survey of metaheuristic-based techniques for university timetabling problems".* OR Spectrum, 2008. **30**(1): p. 167-190.
6. *Reglamento Trabajo Docente y metodológico*, M.D.L.E. Superior, Editor. 2010.
7. ALFONSO, H., *Resolviendo problemas de optimización con técnicas inteligentes.* En XIII Workshop de Investigadores en Ciencias de la Computación, 2011.
8. ETÁN OLIVÁN, J., *BOJ DEL VAL.* Eva. Programación lineal, 2012.
9. SARMIENTO-LEPESQUEUR, A., *Programación y asignación de horarios de clases universitarias: un enfoque de programación entera.*
10. PIEDRAHITA, E., *Programación lineal.* Revista Universidad EAFIT, 2012. **1**(1): p. 41-62.
11. PASTOR MORENO, R., *Metalgoritmo de optimización combinatoria mediante la exploración de grafos.* 2012.
12. MARTÍ, R., *Algoritmos heurísticos en optimización combinatoria.*, in *Departament d'Estadística i Investigació Operativa,Facultat de Matemàtiques*. 2005, Universitat de Valèencia.
13. MARTI, R., *Procedimientos metaheurısticos en optimización combinatoria.* Matemátiques, 2003. **1**(1): p. 3-62.

14. GENDREAU, M.P., Jean-Yves., *Metaheuristics in combinatorial optimization.* Annals of Operations Research, 2005. **140**(1): p. 189-213.
15. OSMAN, I.H.K., James P., *Meta-heuristics: theory and applications.* 1996: Springer.
16. WREN, A.S., *timetabling and rostering—a special relationship?* En Practice and theory of automated timetabling., 1996(Springer Berlin Heidelberg): p. 46-75.
17. Schaerf, A., *A survey of automated timetabling.* Artificial Intelligence Review archive, April 1999. **13**(2): p. 87-127.
18. MEJÍA CABALLERO, J.M., *Asignación de horarios de clases universitarias mediante algoritmos evolutivos*, in *Ingenieria Industrial*. 2009, Universidad del Norte.
19. *Generador de Horarios para Centros de Enseñanza*. [cited 2013 31 enero]; Available from: http://www.penalara.com.
20. *Timetab*.[cited 2009 21 febrero]; Available from: http://www.abcdatos.com/programas/programa/l1382.html.
21. *TimeTabler Lantiv.* [cited 2013 20 febrero]; Available from: http://www.lantiv.com.
22. *TimeTabler Mimosa Software*. [cited 2013 20 febrero]; Available from: http://www.mimosasoftware.com.
23. Faculty of Engineering, C.a.C.I., School of Computing, *Computing Research*. 2002.
24. Hocaoglu, C.H.A.a.G., *A TABU SEARCH ALGORITHM TO SOLVE A COURSE TIMETABLING PROBLEM.* ,Journal of Mathematics and Statistics, 2007. **36(1)**.
25. Moray . Nandhini , D.S.K., *A Survey of Simulated Annealing Methodology for University Course Timetabling* International Journal of Recent Trends in Engineering, May 2009. **1**(2).
26. Grosan C., y.A.A., *Intelligent Systems: A Modern Approach. Intelligent Systems Reference Library*. 2011, Springer.
27. Mejia Caballero, I.J.M., *Asignacion de horarios de clases universitarias mediante algoritmos evolutivos* in *Division de Postgrados e investigaciones en ingeniería Maestria en ingeniería Industrial*. Junio del 2008, Universidad del norte Barranquilla-Atlantico.
28. Rushil Raghavjee, Nelishia Pillay,. *A Comparison of Genetic Algorithms and Genetic Programming in Solving School Timetabling Problems.* IEEE Press, 2012.
29. Holland, H.J., *Adaptation in Natural and Artificial Systems*. 1975: University of Michigan Press. 211.
30. Patón, D.E.F.-M., *Alarcos*. 2006.

31. Kamilosol1. *Metodologías Tradicionales vs. Metodologías Ágiles*. 2007 [cited 2007 Julio 8].
32. Omg.org. *Introduction to OMG UML*. 2009 [cited 2 Abril, 2010]; Available from: http://www.omg.org/gettingstarted/what_is_uml.htm.
33. (BPMI), B.P.M.I., *Business Process Modeling Notation (BPMN)*
Vol. 1. 2004.
34. Informática, I.N.d.E.e., *Herramientas Case.El mejor soporte para el proceso de desarrollo de software*. Colección Cultura Informática. 2010.
35. Ordax, J.M., *Java Básico.Introducción a Java*. 2004.
36. CHAMBA, L.A. *JGAP-y-Algoritmos-Geneticos*. 2012 [cited; Available from: http://es.scribd.com/jimaca%C3%B1a/d/59637214-JGAP-y-Algoritmos-Geneticos.
37. Méndez, G., *Requisitos Funcionales*. 2008: Madrid,España.
38. PRESSMAN, R.S., *Ingeniería del software: Un enfoque práctico.*
39. Christine Hofmeister, R.N.y.D., *Describing Software Architecture with UML*. IEEE Computer Society Press ed. 1999.
40. Reynoso, C.B., *Introducción a la Arquitectura deSoftware.* Marzo de 2004.
41. Ivar Jacobson, G.B., James Rumbaugh.,. *El proceso unificado de desarrollo de software*. 2000.
42. Patrones de Diseño. Prieto, Félix.2009.
43. Rich, E., Inteligencia Artificial. Segunda Edición. 1994, Madrid.
44. José H. Canós, P.L.y.C.P., Métodologías Ágiles en el Desarrollo de Software: Valencia.
45. Ingeniería de Software. Septima edición. Sommerville, Ian.Madrid : Pearson Educación S.A.: s.n., 2005
46. Jesús M. González Barahona, J.S.y.G.R., Introducción al software libre, UOC, Editor, 2007.

Glosario de Términos

Algoritmo genético: Es un método adaptativo que puede usarse para resolver problemas de búsqueda y optimización. Está basado en el proceso genético de los organismos vivos.

BPMN: Es el acrónimo de *Business Process Modeling Notation* y es una notación gráfica estandarizada para el modelado de los procesos de negocio.

Exploración: Es realizada en el espacio de soluciones, realiza una búsqueda en amplitud, localizando así zonas prometedoras.

Explotación: Es realizada en el espacio de búsqueda, realiza una búsqueda en profundidad en dichas zonas, obteniendo así las mejores soluciones.

Framework: Marco de trabajo que define en términos generales un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular. Sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

JDK: El *Java Development Kit*, por sus siglas en inglés, es un grupo de herramientas para el desarrollo de software. Incluye las herramientas necesarias para escribir, testear y depurar aplicaciones y *applets* de Java.

Metaheurísticas: Las metaheurísticas generalmente se aplican a problemas que no tienen un algoritmo o heurística específica que dé una solución satisfactoria

Patrones GRASP: Son patrones generales de software para asignación de responsabilidades. Son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software.

Patrones GoF: Son patrones de diseño enfocados en la comunicación de clases y objetos. Son adaptados para resolver un problema general de diseño en un contexto particular.

POC: La optimización combinatoria es una rama de la optimización en matemáticas aplicadas y en ciencias de la computación, relacionada a la investigación de operaciones, teoría de algoritmos y teoría de la complejidad computacional. También está relacionada con otros campos, como la inteligencia artificial e ingeniería de software. Los algoritmos de optimización combinatoria resuelven instancias de problemas que se creen ser difíciles en general, explorando el espacio de soluciones (usualmente grande) para estas instancias.