Generación automática de la planificación de la entrega en desarrollo de software agil, asignación de historias de usuario a los desarrolladores usando algoritmos genéticos.

Automatic generation of sprint planning in agile software development, assignment of user stories to developers using genetic algorithms.

Fredy Humberto Vera-Rivera[1], Jose Luis Barbosa-Mora[2], Carlos Mauricio Gaona-Cuevas[3]
[1]Universidad Francisco de Paula Santander, Colombia
[2]Universidad Pontificia Bolivariana, Colombia
[3]Universidad del Valle, Colombia

Recibido: 09 marzo de 2020.
Aprobado: 24 de abril de 2020.

***Resumen*—** En el desarrollo de aplicaciones software usando metodologías ágiles, la asignación de las tareas de desarrollo es una actividad fundamental, de ella depende el éxito del desarrollo del proyecto, asignar las tareas de desarrollo a la persona correcta, en el tiempo adecuado y de forma óptima, puede traer una reducción en el tiempo de desarrollo y en los costos del proyecto. El problema de la planificación y asignación de tareas a recursos o personas, conocido como "scheduling" ha sido abordado desde diferentes enfoques y disciplinas, por ejemplo: la investigación de operaciones, la programación numérica y la programación lineal. En este trabajo se aborda este problema aplicado al desarrollo de software ágil, donde se busca asignar de forma automática y óptima las historias de usuario que deben implementar en una iteración (sprint) el equipo de desarrollo, teniendo en cuenta las características propias del equipo, por ejemplo, su experiencia (desarrollador junior, senior o novato). Se propone un algoritmo genético que genera la asignación de tareas para la iteración (sprint) de desarrollo. Se realizó la validación del algoritmo propuesto en un caso de estudio real, se pudo observar que el resultado obtenido mejora considerablemente al obtenido por el líder del proyecto. En el caso de estudio real se redujo el tiempo estimado de desarrollo de 99 horas a 87 horas (12%), siendo una diferencia importante que representa ahorros en los costos del proyecto. Como trabajo futuro se pretende usar casos reales más complejos, con más desarrolladores e historias para validar el método propuesto.

Palabras Claves: planificación ágil, prácticas agiles, algoritmo genético, spring backlog, desarrollo de software ágil

***Abstract*—** In the development of software applications, the assignment of development tasks is a fundamental activity, on it depends the success of the development of the project, assigning the development tasks to the right person, at the right time and optimally, can bring a reduction in development time and project costs. The problem of programming and assigning tasks to resources or people, known as "scheduling," has been approached from different approaches and disciplines, for example: operations research, numerical programming and linear programming. This work addresses this problem applied to the development of agile software, where it is sought to automatically and optimally assign the user stories that the development team must implement in an iteration or sprint, taking into account the characteristics of the team, for example, your experience (junior, senior or novice developer). A genetic algorithm is proposed that generates the assignment of tasks for the iteration or development sprint. The validation of the proposed algorithm was carried out in a real case study, it was observed that the obtained result improves considerably to that obtained by the project leader. In the case study, the estimated development time was reduced from 99 hours to 87 hours, being an important difference that represents savings in project costs. As future work it is intended to use more complex real-life cases, with more developers and stories to validate the proposed method.

Keywords: agile scheduling, agile practices, genetic algorithm, spring backlog, agile software development

*Autor para correspondencia.
Correo electrónico: fredyhumbertovera@ufps.edu.co (Fredy Humberto Vera Rivera).
La revisión por pares es responsabilidad de la Universidad de Santander.

Forma de citar: F. H. Vera-Rivera, J. L. Barbosa-Mora y C. M. Gaona-Cuevas, "Generación automática de la planificación de la entrega en desarrollo de software agil, asignación de historias de usuario a los desarrolladores usando algoritmos genéticos", Aibi revista de investigación, administración e ingeniería, vol. 8, no. 2, pp. 29-38, 2020.

I. INTRODUCCIÓN

El desarrollo de software se ha convertido en un factor importante para el desarrollo de las tecnologías modernas, en la última década el mercado de software ha tenido un crecimiento exponencial [1], el uso de software se da en casi todos los productos y servicios, el manejo y procesamiento de información, continua creciendo, cada vez se requiere software más especializado, el software que se construye debe ser seguro, confiable y robusto, interoperable con otros sistemas, con aplicaciones móviles y con un mundo inteligente que evoluciona cada día.

La complejidad que implica el desarrollo de software ha sido abordada con el uso de las metodologías y prácticas ágiles que nacieron a partir del manifiesto ágil y sus principios, en contraste a las formas tradicionales [2]. Los beneficios más importantes que se obtienen con el uso y adopción de las prácticas ágiles son: el incremento de la productividad del equipo de desarrollo, se mejora la visibilidad del producto software, se adquiere la habilidad para manejar y controlar el cambio en los requerimientos y en las prioridades del proyecto y se hacen entregas más rápidas del producto. Como las principales barreras que tienen las empresas para la adopción de las prácticas ágiles están: (1) la falta de planificación inicial, (2) la pérdida del control de la gestión y (3) la falta de previsibilidad [3]. Las prácticas o técnicas ágiles más utilizadas según el 13 estudio anual del estado del desarrollo ágil [3] son:

1. Planificación de la iteración (Sprint planning): Permite identificar las tareas que se deben realizar en la iteración, se realiza una reunión con el cliente par identificar las prioridades del proyecto, se define la lista priorizada de requisitos del producto. El equipo planifica la iteración, elabora la táctica que le permitirá conseguir el mejor resultado posible con el mínimo esfuerzo [4].

2. Reunión diaria de pie (Daily Standup): Es una reunión que realiza el equipo de desarrollo al iniciar el día, es una reunión de sincronización del equipo, en la cual se presentan los avances y dificultades que pueden tener los miembros del equipo, de tal forma que se puedan hacer ajustes a tiempo y se puedan cumplir con las tareas asignadas [5].

3. Retrospectivas: Es una reunión al final de la iteración, con el objetivo de mejorar de manera continua la productividad y la calidad del producto que está desarrollando, la motivación del equipo, cómo están engranando entre ellos, como fue la última iteración o cómo está yendo el proyecto; el equipo analiza cómo ha sido su manera de trabajar durante la iteración. Como resultado de una retrospectiva, se pueden encontrar los siguientes: plan de acciones de mejora, nuevas mejores practicas, acuerdos de equipo actualizados y los impedimentos o problemas a escalar [4].

4. Revisión de la iteración: Al terminar la iteración se realiza una revisión con todas las personas involucradas en el proyecto (equipo de desarrollo, cliente, patrocinadores) con el fin de identificar problemas, establecer acuerdos y si es el caso redireccionar el proyecto.

5. Iteraciones cortas: Se recomienda una duración de la iteración entre dos semanas a un mes. La idea es entregar valor al cliente lo más rápido posible a través de iteraciones cortas, entrega y despliegue continuos.

6. Planificación de la entrega (Release planning): La planificación de la entrega permite establecer la lista priorizada de objetivos/requisitos, la cual representa la visión y expectativas del cliente respecto a los objetivos y entregas del producto o proyecto, normalmente se expresan como historias de usuario. Se indican las posibles iteraciones y las entregas (releases) esperadas por el cliente, estas se pueden ir adaptando y modificando a medida que avanza el proyecto y según las necesidades del cliente [4].

El uso de las prácticas ágiles ha mejorado considerablemente el proceso de desarrollo de software, obteniendo productos software de mayor calidad y mejorando los tiempos de entrega. En Colombia, según un estudio realizado por J. Parada y otros, se evidencia que el 73% de los profesionales encuestados aplicaron una metodología ágil para el desarrollo de productos de software y del porcentaje obtenido, el 43.42% utilizó la metodología Scrum y el 19.7% aplicó el híbrido Scrum / XP [6]. Siendo esto un indicador de la importancia y del uso de las metodologías ágiles.

Las metodologías ágiles están dirigidas a reducir la probabilidad de fracaso por subestimación de costos, tiempos y funcionalidades en los proyectos de desarrollo de software [7]. La estimación del esfuerzo en desarrollar el producto software es una tarea fundamental e importante, Rojas M y otros [8] proponen una herramienta para estimar el proyecto software de acuerdo a los módulos de programación concebidos como un todo, proponen una métrica híbrida permite la comparación de estimaciones de funciones transaccionales y de datos. Mas'ad, Rafik y otros [9], proponen Blacksheep, una técnica novedosa para la estimación del esfuerzo del proyecto en contextos ágiles. Como método orientado al aprendizaje, Blacksheep se basa en datos para mejorar las predicciones de esfuerzo. La planificación de las tareas de desarrollo "Scheduling" es una parte fundamental del proceso de desarrollo de software y debe considerar métricas de estimación del esfuerzo como las enunciadas.

El "Scheduling" define cuándo y quién debe realizar los requerimientos en la iteración, en cambio el "Release planning" define qué debe ser hecho y los criterios de aceptación para el desarrollo del producto software [10], esta planificación puede cambiar a medida que avanza el desarrollo del software. El "Sprint plannig" define qué se debe hacer en una iteración, define las funcionalidades del incremento de producto a lograr. El sprint corresponde a una iteración a un incremento del producto, mientras que Release corresponde a todo el producto. El "Scheduling" se encarga de la programación y asignación de tareas en el Sprint, teniendo en cuenta las prioridades de los requerimientos o historias de usuario, el valor del negocio a realizar, el esfuerzo que implica cada requerimiento y las características del equipo de desarrollo como la velocidad y los conocimientos y destrezas individuales. El objetivo de la planificación de la iteración es dividir las funcionalidades o historias de usuario seleccionadas en tareas técnicas y asignarlas a los desarrolladores de tal forma que esa asignación sea optima [11]. En este trabajo se propone un método de programación genética para la planificación ágil que permite realizar la planificación de la iteración de forma automática, inteligente y optima.

La planificación de la versión de software (SRP) es el problema de seleccionar qué características o requisitos se incluirán en la próxima versión o versiones. Es un paso crucial en el desarrollo de software, que resulta ser extremadamente complejo dada la necesidad de conciliar múltiples criterios de toma de decisiones (por ejemplo, valor comercial, esfuerzo y costo), al tiempo que considera varias restricciones (por ejemplo, precedencia de características, disponibilidad de recursos) [12]. En este artículo se aborda parcialemente este problema, enfocado en la planificación en el contexto del desarrollo de software usando metodologías ágiles, teniendo en cuanta que los requisitos del software están planteados como historias de usuario, definidos para la iteración (Sprint) y se quieren asignar a los desarrolladores para ser implementados de forma optima, en el menor tiempo posible.

Esta planificación la realiza tradicionalmente el líder del proyecto, en ocasiones apoyado por herramientas tecnológicas que le permiten organizar las tareas y asignarlas al equipo de desarrollo. La mayoría de estas herramientas tecnológicas no tienen alguna forma automática, inteligente y optima de realizar esta asignación, son herramientas muy generales, no son especificas al desarrollo de software ágil e implican un trabajo tedioso y demorado para obtener una adecuada planificación, quedando a criterio del líder de proyecto. Por esos

motivos se plantea el siguiente problema de investigación ¿Por medio de programación genética se puede mejorar la planificación de la iteración teniendo en cuenta las características propias del equipo de desarrollo y de los requerimientos planteados como historias de usuario?

Como objetivo principal de este trabajo, se propone un método automático de planificación y asignación de las historias de usuario al equipo de desarrollo, establece qué se debe realizar en una iteración y quién lo debe realizar, teniendo en cuenta las características propias del equipo de desarrollo, la complejidad de las historias y la duración de la iteración, como resultado se genera el "Sprint backlog" de la iteración, usando un algoritmo genético, el cual permiten encontrar una solución optima al problema planteado a partir de individuos, reproducciones, mutaciones y selección del mejor.

Los algoritmos genéticos fueron establecidos por Holland [13], su dominio principal de aplicación es la optimización de funciones. Los algoritmos genéticos son procedimientos adaptativos para la búsqueda de soluciones en espacios complejos inspirados en la evolución biológica, con patrones de operaciones basadas en el principio Darwiniano de reproducción y supervivencia de los individuos que mejor se adapten al entorno en que viven [14].

Para abordar el problema de investigación se utilizó la metodología "Design Science Research" [15], en la cual el artefacto a crear y validar fue al algoritmo genético para la planificación de la iteración. Se realiza una demostración en un caso hipotético y se valida en un caso de estudio de la vida real de la empresa BITS Americas[1], empresa fundada en el año 2008 en Medellín, Colombia, bajo el concepto de proporcionar desarrollo de software a la medida y soluciones basadas en sitios Web para las empresas.

Algunos autores han abordado el tema de la planificación y de asignación de requisitos, por ejemplo, D. Greer and G. Ruhe, proponen un método para asignar de manera óptima los requisitos a los incrementos usando un algoritmo genético, genera un conjunto de las soluciones candidatas más prometedoras para respaldar la decisión final [16], este método no asigna las historias a los desarrolladores, siendo diferente al propuesto en este artículo. G. Ruhe y M. Sailu presentan dos enfoques para la planificación de la iteración. Primero, el arte del enfoque de "Release Planing" se basa en la intuición humana, la comunicación y las capacidades para negociar entre objetivos y limitaciones en conflicto. Y segundo, la ciencia del enfoque del "Release Planign" formaliza el problema y aplica algoritmos computacionales para generar las mejores soluciones. Proponen crear una sinergia entre los dos, integrando inteligencia humana y computacional para definir asignaciones óptimas de funciones de planificación [17].

Victor Hugo Escandon y otros, proponen un algoritmo genético multiobjetivo para actualizar o ajustar el plan original de desarrollo que por algún evento disruptivo se debe cambiar. Se consideran los objetivos de tiempo, costo, estabilidad, desaprovechamiento de la capacidad de desarrollo y valor de la liberación [18]. Karim Muhammad Rezaul y Ruhe Guenther modelan la planificación de lanzamientos basada en temas como un problema de optimización bi-objetivo (basado en búsquedas), adaptan el algoritmo genético de clasificación no dominado existente (NSGA-II) y proponen dos formas alternativas de seleccionar entre las soluciones óptimas de Pareto (un número potencialmente grande) la mejor planificación del lanzamiento [19]. Estos trabajos son referentes importantes de este trabajo de investigación, aunque los algortimos se enfocan en otros aspectos de la planificación.

D. Ameller y otros, realizaron en 2016 una revisión del estado del arte sobre los modelos planteados para la planificación de la versión del software (SRP), identficaron 17 métodos enfocados en diferentes aspectos, concluyen que se hace evidente que las propuestas científicas de SRP aún no han alcanzado la madurez requerida por los contextos industriales, también concluyen que el estado actual del arte reclama un esfuerzo creciente en acercar los modelos SRP a la industria [12]; por este motivo planteamos el presente trabajo de investigación, realizando una prueba de concepto de un algoritmo genético para la planificación ágil aplicado en un caso de estudio real de la empresa Bits Americas, teniendo un acercamiento y validación con la industria.

A continuación, en la sección II se presenta los conceptos y buenas prácticas del "Scheduling" o planificación de tareas, detallando la forma tradicional de realizar esta programación y la forma ágil propuesta en la literatura. Posteriormente en la sección III se explica el método llevado a cabo para la implementación y validación del algoritmo propuesto, la generación automática del "Agile Scheduling", posteriormente se discuten los resultados en la sección IV, por último, se dan las conclusiones obtenidas en el trabajo.

II. PLANIFICACIÓN AGIL – CONCEPTOS Y BUENAS PRÁCTICAS

El presente trabajo se enfoca en la práctica ágil más importante, la planificación de la iteración (Sprint planing), la cual consiste en [4]:

1. El cliente presenta al equipo la lista de requisitos priorizada del producto o proyecto, se da un nombre a la meta de la iteración y propone los requisitos más prioritarios a desarrollar.
2. El equipo examina la lista, resuelve las dudas con el cliente y añade condiciones de satisfacción, selecciona los objetivos o requisitos prioritarios que se compromete a completar en la iteración.
3. El equipo planifica la iteración, elabora la táctica que le permita conseguir el mejor resultado posible con el mínimo de esfuerzo.
4. El equipo define las tareas necesarias para completar el objetivo, define el sprint backlog.
5. El equipo realiza una estimación del esfuerzo necesario para realizar cada tarea.
6. Cada miembro del equipo se autoasigna a las tareas que puede realizar.

A continuación, se presenta una revisión de los conceptos y buenas prácticas para realizar la programación o asignación de tareas; en primer lugar, se analiza la forma tradicional y luego la forma ágil.

2.1.Forma tradicional

La asignación de tareas o "Scheduling" de un proyecto de desarrollo de software, generalmente la realiza el líder del proyecto o líder del equipo de desarrollo. En ocasiones esta asignación se realiza a su criterio, usando herramientas para gestión de proyectos como Microsoft Project o Google Schedule. El líder organiza las tareas y las asigna a los desarrolladores de forma empírica, usando las facilidades que le ofrecen estas herramientas. Al ser una asignación empírica en ocasiones esta asignación no es la óptima, teniendo como consecuencia fechas de entregas mal calculadas y retrasos en la entrega del producto. Con la práctica, con la experiencia de líder del proyecto y con el conocimiento de su equipo de desarrollo se pueden lograr asignaciones exitosas en tiempos, costos y esfuerzos.

El "Scheduling" ha sido estudiado ampliamente por el área de gestión de proyectos, principalmente por el Project Management Institute (PMI), el cual define en su PMBOK (Project Management Body of Knowledge) una serie de buenas prácticas, estándares y guías en la gestión de proyectos [20]. El "Scheduling" en la gestión de proyectos del PMI se relaciona con las áreas de conocimiento: gestión del alcance y gestión del tiempo principalmente, sus actividades se

[1] https://www.bitsamericas.com/inicio

ubican en los grupos de procesos de planificación y de control del proyecto.

En la gestión del alcance del proyecto se definen 4 actividades importantes: (1) Planificar la gestión del alcance, en la cual se indica cómo se va a planificar, gestionar y controlar el proyecto. (2) Recopilar requisitos, que consiste en documentar las necesidades, deseos y expectativas cuantificadas y documentadas de los interesados para convertirlas en requisitos del proyecto. (3) Definir el alcance, donde se desarrolla una descripción detallada del proyecto y del producto, se analizan los riesgos, los supuestos y las restricciones existentes. (4) Crear la EDT, La Estructura de Desglose de Trabajo (EDT) describe el alcance del proyecto según sus entregables, los cuales se dividen en componentes lo suficientemente pequeños y manejables que permitan planificar y controlar el proyecto. Estos componentes se denominan paquetes de trabajo que se programan, supervisan, controlan, estiman su costo y se asignan a un solo responsable de su ejecución [21]. Estos paquetes en el desarrollo ágil se podrían asimilar a las historias de usuario porque en las historias de usuario se plantean los requerimientos, las tareas técnicas de desarrollo, las prioridades, las estimaciones y los alcances del proyecto ágil.

En el área de gestión del tiempo, se controla que el proyecto termine dentro del plazo previsto. Sus resultados principales son la definición del cronograma del proyecto, junto con su línea base, calendarios, fechas de los entregables, actividades con su duración, dependencias y recursos requeridos. Las actividades propuestas son: (1) Planificar la gestión del cronograma, (2) Definir las actividades, (3) Secuenciar las actividades, (4) Estimar los recursos de las actividades, (5) Estimar la duración de las actividades y (6) Desarrollar el cronograma. Estas áreas, actividades y procesos se pueden adaptar para ser usadas en proyectos de desarrollo de software. Muchas empresas de diferentes áreas del conocimiento usan de forma exitosa el PMBok para la gestión de proyectos, así como las empresas de desarrollo de software.

Sheuly y Smolander presentan una revisión sistemática de literatura sobre el Agile Project Managament (APM), categorizan los resultados en cuatro áreas: (1) APM introducción y adopción, (2) APM métodos y enfoques, (3) Factor de equipo y (4) Estudios comparativos [22].

Malgwi and Blamah, proponen un enfoque basado en múltiples agentes para la programación del proceso de desarrollo de software, donde cada actividad se considera un proceso de agente autónomo y flexible. Un aspecto crucial para el sistema basado en múltiples agentes en la programación de proyectos es la disponibilidad de un modelo y un algoritmo eficaces para la programación de tareas. El modelo desarrollado (Sistema basado en múltiples agentes) proporciona una programación ágil y optimizada del proceso ágil y reduce los gastos generales en el proceso del software, ya que responde rápidamente a los requisitos cambiantes sin un exceso de trabajo en la programación del proyecto [23].

2.2. Forma ágil

La planificación ágil parte de la idea de planificar en función de objetivos de negocio en lugar de tareas (a diferencia de la planificación tradicional), priorizando los que aportan más valor, y esperando a dar detalle a objetivos y tareas conforme se va acercando el momento de construcción de estos objetivos, cuando la indeterminación se va reduciendo, de manera que se amortiza el esfuerzo de planificar de manera detallada [4].

En las metodologías ágiles, la planificación de la entrega “release planing” no sólo guía el desarrollo, sino que también guía la planificación de los proyectos y proporciona las estimaciones de esfuerzo necesario. Los equipos ágiles se centran en estimaciones "lo suficientemente buenas" para tratar de optimizar sus esfuerzos. Un enfoque de buena estimación toma poco tiempo y puede utilizarse para el seguimiento del proyecto [24].

Los modelos de estimación del esfuerzo en las metodologías ágiles, se basan en métricas de tamaño: Puntos de historias de usuario [25], puntos de características [26] y puntos de casos de uso [27]. Estas métricas son unidades abstractas, que expresan todo el tamaño del desarrollo y por lo general no son directamente convertibles en horas/días/meses por desarrollador. Estos modelos se basan en la selección y asignación de tareas de manera incremental, usando prioridades relativas en función de los objetivos empresariales y otros factores como las urgencias y los riesgos [16], también teniendo en cuenta las necesidades de las partes interesadas. Los valores son medidos en una escala de relaciones, lo que significa que los valores se ordenan y se pueden determinar la relacioón entre ellos [28].

La planificación ágil es iterativa, se va adaptando y mejorando a medida que el desarrollo avanza, se tiene en cuenta la velocidad del equipo de desarrollo y las características propias del equipo.

Para profundizar entre las similitudes y relaciones entre PMBOK y metodologías ágiles puede consultar a Clara Vidal y Patricio Letevier quienes analizan las similitudes y diferencias de estos enfoques en [29]. También P. Fitsilis compara un conjunto genérico de procesos de gestión de proyectos como se define en el PMBOK con una serie de procesos ágiles de gestión de proyectos. El resultado es que las metodologías ágiles de gestión de proyectos no pueden considerarse completas, desde el punto de vista tradicional de gestión de proyectos, ya que faltan varios procesos o no se describen explícitamente [30].

La idea central de este trabajo de investigación fue abordar la planificación ágil, se propuso un método automático de planificación de la iteración usando programación genética, de tal forma que esta asignación sea menos subjetiva al líder del proyecto y se haga basada en la experiencia del equipo de desarrollo y en la complejidad de cada historia de usuario. En este artículo se presenta la prueba de concepto realizada en la empresa Bits Americas, en la cual se usa un caso de estudio de la vida real para comparar la asignación realizada por el líder del proyecto contra la planificación realizada por el algoritmo genético, estos resultados se presentan en la sección V.

III. METODOLOGÍA

La metodología de este trabajo de investigación se basa en “Design Science Research” siguiendo los planteamientos hechos por Hevner y otros [15] para la investigación científica en el campo de los sistemas de información y el desarrollo de productos tecnológicos. El paradigma de la ciencia del diseño busca extender los límites de las capacidades humanas y organizacionales mediante la creación de artefactos nuevos e innovadores [15]. En la presente investigación el artefacto a crear es el método automático de planificación de la iteración.

La figura 1 presenta el modelo metodológico utilizado para diseñar, construir, probar y validar el método propuesto.

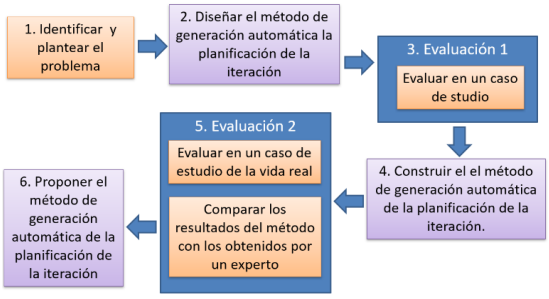

Figura 1: Modelo metodológico del trabajo de investigación.
Fuente: creación propia con base en aportes de [15].

Fase 1: Identificar y plantear el problema: Se realizó el planteamiento del problema que se quiere resolver usando "Design science research". Se definió como artefacto a desarrollar el algoritmo genético que permita optimizar la planificación de la iteración en el desarrollo de software ágil.

Fase 2: Diseñar el método de generación automática de la planificación de la iteración: Se realizó el diseño del artefacto a proponer, en este caso sería el método de generación automática de la planificación de la iteración, el diseño se realiza basados en teórias conocidas y aceptadas como son los algoritmos genéticos. El diseño del algoritmo genético se puede apreciar en la sección IV.

Fase3: Evaluación en un caso de estudio: Una vez diseñado el algoritmo inteligente se evalúa su uso en un caso de estudio académico e hipotético que permite ilustrar y verificar el funcionamiento del método. Los resultados de la aplicación del algoritmo genético en el caso de estudio se pueden apreciar en la sección V.

Fase 4: Construir el método de generación automática de la planificación de la iteración: Posteriormente se implementa y códifica la solución propuesta. Se construyó el algoritmo genético en Java, se hicieron pruebas y validaciones para verificar su correcto funcionamiento, primero con el caso de estudio hipotético y luego con el caso de la vida real.

Fase 5: Evaluar el método propuesto en un caso de la vida real: Una vez construido y verificado que el algoritmo funciona corrrectamento, se evaluó en un caso de estudio real en Bits Americas una empresa de desarrollo de software en el sector de las telecomunicaciones de Medellín, Colombia. En el caso de estudio real se tienen los datos de la planificación de una iteración realizada por el líder de proyecto, experto calificado y certificado en gestión de proyectos por PMI. Se comparan los resultados obtenidos por el método automático propuesto con los obtenidos por el experto. Se evalúan la solución obtenida en caso de requerir ajustes se cambia el diseño del método propuesto. Los resultados obtenidos en esta evaluación se pueden apreciar en la sección 5.1

Fase 6: Proponer el método de generación automática de la planificación de la iteración: Una vez realizada la prueba de concepto y obtenido un resultado satisfactorio se procede a socializar con el líder del proyecto de la empresa Bits Americas, de tal forma que en un trabajo futuro se pueda aplicar en más proyectos y obtener una validación mas completa, con casos de estudio reales y más complejos, con más historias de usuario y más desarrolladores. En este artículo se socializan y se hace la propuesta del método de planificación de la iteración usando algoritmos genéticos.

A continuación, en la sección IV se plantea el algoritmo genético para la planificación automática de la iteración en desarrollo de software ágil.

IV. GENERACIÓN AUTOMÁTICA DE LA PLANIFICACIÓN DE LA ITERACIÓN

El problema abordado consiste en encontrar la forma óptima de asignar las historias de usuario a los desarrolladores en una iteración. Teniendo en cuenta que se quiere minimizar el tiempo de desarrollo del conjunto de historias, o maximizar el número de puntos de historia que se pueden resolver en dicha iteración.

Los algoritmos genéticos fueron establecidos por Holland [13], su principal dominio de aplicación lo constituye la optimización de funciones, en este caso sería la optimización del tiempo o de los puntos de historia que se realizan en una iteración, de tal forma que se reduzca el tiempo de desarrollo o se maximice el número de puntos de historia realizados en el Sprint. El pseudocódigo del Algoritmo Genético abstracto es el siguiente:

```
1. BEGIN
2.     obtenerPoblacionInicial() // Sección 4.1
3.     WHILE NOT stop()
4.         seleccionarPadres() // Un padre aleatorio
5.         reproducción() // Sección 4.3
6.         mutacion() // Sección 4.4
7.         extenderPoblacion() // Agregar hijos y mutados
8.         seleccionarMejores() // Según Ecuación 3
9.     END
10. END
```

El proceso es iterativo, en cada iteración se seleccionan los mejores individuos, cada individuo tiene un cromosoma, el cual se cruza con otro individuo para generar la nueva población con el fin de encontrar la solución óptima al problema. Se ha demostrado matemáticamente que el algoritmo converge a la solución. Ahora se procede a plantear el algoritmo genético para el problema de este trabajo que consiste en distribuir o asignar las historias de usuario a los desarrolladores de forma automática. A continuación, se propone el diseño del método propuesto.

4.1. Definición de la población

Se tiene un conjunto de desarrolladores $D = \{D_1, D_2, D_3, \ldots, D_i\}$ que son los miembros del equipo de desarrollo que van a trabajar en la iteración. Cada desarrollador D tiene unas características propias, como son identificación, nombre, nivel de experiencia (1 – Senior, 2 – Junior, 3 - Novato) y el porcentaje de dedicación al proyecto (1 a 100%).

Se tiene un conjunto de historias de usuario $H = \{H_1, H_2, H_3, \ldots, H_j\}$, las cuales deben ser resueltas en la iteración, ordenadas por su prioridad. También con unas características propias: identificador, prioridad, dependencias con otras historias, tiempo estimado y los puntos de historia estimados.

Consideraciones: Se asume que los puntos y tiempos estimados se hicieron teniendo en cuenta cualquier técnica de estimación y que son genéricos para cualquier nivel de experticia del desarrollador. Esto implica que un desarrollador novato va a gastar más tiempo en implementar dicha historia de usuario que el estimado y también que un desarrollador senior. Para reducir la complejidad del algoritmo inicialmente no se van a tener en cuenta las dependencias entre historias de usuario, se asume que no existe dependencias entre ellas, en el caso que exista dependencias se podrían unir las historias de usuario en una sola.

Los individuos se definen a partir de la asignación de las historias a los desarrolladores, por lo tanto, el cromosoma de cada individuo se define a partir de una matriz de asignación de unos y ceros, donde en las columnas se tienen las historias de usuario y en las filas los desarrolladores y el cruce contiene un 1 cuando la tarea es asignada al desarrollador o cero en caso contrario. En la tabla 1, se presenta un ejemplo para 2 desarrolladores (D1, D2) y 5 historias de usuario (H1, H2, H3, H4, H5).

Tabla 1: Matriz de asignación para generar el cromosoma paras dos desarrolladores y 5 historias de usuario.

	H1	**H2**	**H3**	**H4**	**H5**
D1	1	0	0	1	1
D2	0	1	1	0	0

Fuente: Elaboración propia.

Por lo tanto, el desarrollador D1 tiene asignadas las historias {H1, H4 y H5} de este caso hipotético. El cromosoma resultante sería la unión de las asignaciones a cada desarrollador, fila 1 unida o concateneada con la fila 2 de la matriz de asignación, para este caso sería: Cromosoma: 10011 01100. A partir de este cromosoma es posible definir la función de adaptación o función objetivo.

4.2. Función de adaptación o función objetivo

La función objetivo estará dada en términos del tiempo, se busca minimizar el tiempo total en el cual se desarrollan todas las historias de usuario de la iteración. También se podría plantear a partir de los puntos de historia, estos serán tenidos en cuenta en un trabajo futuro. El cálculo del tiempo depende del tiempo que dura un desarrollador en terminar una historia de usuario y el conjunto de historias asociadas a él, teniendo en cuenta que:

1. El tiempo estimado para la historia denotado por *t*. Este tiempo se especifica como dato de entrada al algoritmo genético y hace parte de la información de la historia de usuario. Es un estimado que hace el equipo de desarrollo de acuerdo con su experiencia y a su velocidad de desarrollo.
2. El nivel de experticia del desarrollador denotado por *NE*, se asigna un peso a cada nivel asi: para un desarrollador Senior se asigna un nivel de experticia *NE=1*, a un desarrollador Junior un nivel de experticia *NE=1.5*, a un desarrollador Novato un nivel de experticia *NE=2*. Este peso afecta directamente al tiempo *t* estimado para implementar la historia de usuario.
3. La dedicación denotada por *d* del desarrollador en el proyecto, es el porcentaje del tiempo que el desarrollador va a trabajar en el proyecto. Es un número entre 0 y 1, corresponde al porcentaje de dedicación.
4. El tiempo T_{ij} corresponde al tiempo que gasta el i-esimo (*i*) desarrollador en implementar la j-esima (*j*) historia de usuario que tiene asignada.
5. El tiempo total T_i coresponde al tiempo total que el i-esimo desarrollador gasta en terminar el total de historias de usuario asignadas a él.

Por lo tanto, si la dedicación del desarrollador *i* al proyecto es del 100% el tiempo que gasta en terminar la historia de usuario *j* esta dado por:

$$T_{ij} = t_j * NE_i \quad (1)$$

Si la dedicación es menor al 100%, se divide entre el porcentaje de dedicación (d_i) asignado:

$$T_{ij} = t_j * NE_i / d_i \quad (2)$$

Donde *i* corresponde al i-esimo desarrollador, y *j* a la j-esima historia de usuario asignada al desarrollador. Así el tiempo total del desarrollador *i* sería:

$$T_i = \sum_{j=1}^{n} T_{ij} \quad (3)$$

Donde *n* corresponde al número de historias de usuario asignadas al desarrollador *i*. Ahora, el tiempo total sería el mayor valor *T* de los *Ti*, para los i=1 hasta el número total de desarrolladores *m*:

$$T = \text{mayor } [T_1, T_2, T_3, \ldots, T_m] \quad (4)$$

Con el algoritmo genético se busca encontrar la mejor combinación, la mejor asignación de las historias a los desarrolladores de tal forma que el tiempo T sea el menor.

Para explicar de mejor forma la aplicación de las ecuaciones planteadas se definió el ejemplo hipotético de la tabla 2.

Tabla 2: Ejemplo hipotético de historias de usuario a desarrollar en una iteración.

	Tiempo Estimado (*t*)	**Puntos**
H1	5	10
H2	7	14
H3	4	8
H4	6	12
H5	8	16
Total	30 horas	60

Fuente: Elaboración propia.

Para la matriz de asignación dada en la tabla 1, que tiene como resultado el cromosoma: 10011 01100. Asumiento que el desarrollador D1 tiene una dedicación al proyecto del 100%, es desarrollador junior y el desarrollador D2 del 80% y es desarrollador senior, entonces:

- El desarrollador D1 tiene asignadas las historias {H1, H4, H5}
- El desarrrollador D2 tiene asignadas las historias {H2, H3}
- Usando la ecuación (1) para cada historia de usuario y sumando el resultado como lo indica la ecuación (3) se obtiene para el desarrollador 1 el siguiente valor de $T_1 = (5 \times 1) + (6 \times 1) + (8 \times 1) = 19$ horas.
- De la misma forma para el desarrollador 2, usamos la ecuación (2) y la ecuación (3), $T_2 = (7 \times 1.5 / 0.8) + (4 \times 1.5 / 0.8) = 20.6$ horas.
- Entonces T = mayor [T_1, T_2] = mayor [19, 20.6] = 20.6
- Por lo tanto, el tiempo estimado de desarrollo de la iteración hipotética planteada serían de 20.6 horas.

Con el algoritmo propuesto se busca obtener una asignación más obtima de tal forma que el tiempo estimado de desarrollo se reduzca.

4.3. Función de reproducción

La reproducción consiste en generar un individuo nuevo, a partir de los padres, tomando la información de sus cromosomas, de los cuales se toma una parte de cada uno y se genera un cromosoma nuevo. Para este caso se generaría otra asignación diferente a partir de un padre seleccionado. Para generar el hijo se cambia el orden de asignación, es decir, se intercambia la asignación de un desarrollador con la de otro. Se debe tener en cuenta que una historia de usuario no puede ser asignada dos veces, esto quiere decir que en la matriz de asignación sólo puede aparecer un uno en cada columna.

Ejemplo: Dados los dos cromosomas:

1. 10011 01100
2. 01011 10100

El hijo de 1 sería: 10110 10011
El hijo de 2 sería: 10100 01011

Este proceso de realiza intercambiando una fila de la matriz de asignación, las filas a intercambiar se seleccionan de forma aleatoria. Este proceso de reproducción corresponde en la naturaleza a una reproducción asexual, donde un padre produce un hijo a partir de su información genética, de su cromozoma, siendo el hijo diferente, pero con información heredada de su padre.

4.4. Mutación

La mutación indica cambiar un bit aleatorio del cromosoma, al cambiar un bit del cromosoma de este problema de 1 a 0 o de 0 a 1, implica que una tarea es asignada o desasignada a una persona y esta debe ser asignada o desasignada a otra persona. Esto implica que la mutación se hace sobre dos bits.

Ejemplo: Mutar bit 7 del cromosoma obtenido.
Cromosoma: 01011 10100
Cromosoma mutado: 00011 11100

Los cromosomas mutados deben ser incluidos en la población. Este proceso se realiza de forma aleatoria, se selecciona de la población los individuos a mutar, se realiza la mutación de un bit también de forma aleatoria, para la mutación se calcula el valor de la función objetivo y se incluye en la población.

4.5. Selección

En los procesos de selección genética sobrevive el más fuerte, el individuo que mejor se adapta a las condiciones de su entorno, en el caso del problema de la generación automática de la planificación de la iteración sobreviven los n individuos que mejor se adapten a las condiciones del problema. Es decir, las asignaciones que impliquen un menor tiempo de desarrollo. La selección se realiza a partir de la función objetivo, esta se aplica a cada individuo y la población se ordena de forma ascendente, teniendo en cuenta los primeros lugares, los mejores individuos, correspondientes a las asignaciones que implican menos tiempo.

4.6. Convergencia

Para determinar la convergencia del método se define el número de iteraciones o generaciones de la población a procesar. También se especifica el número de hijos a tener en cada generación y el número de mutaciones a realizar. Al finalizar las iteraciones, se detiene el algoritmo y se selecciona el cromosoma ubicado en el primer lugar, que sería la mejor asignación de tareas.

Para los casos de estudio utilizados para evaluar el método propuesto se generó una población de 100 individuos, con 5 iteraciones o generaciones, con 50 hijos y 50 mutaciones. Estos valores se validaron realizando una evaluación experimental combinando diferentes valores, aumentando la población, las iteraciones, los hijos y las mutaciones, los resultados obtenidos fueron muy similares para los valores escogidos, las diferencias fueron despreciables.

V. RESULTADOS

La implementación del algoritmo genético se realizó en Java, obteniendo resultados interesantes. A partir de un caso de estudio sencillo y de uso académico se realizó la prueba de concepto que se detalla a continuación. Primero se realizó la validación en un caso hipotético, el cual se describe a continuación: se estableció un conjunto de desarrolladores con las características mostradas en la tabla 2.

Se definieron las historias de usuario descritas en la tabla 3, sin dependencias para un caso de prueba hipotetico. El tiempo de duración de la iteración o tiempo de desarrollo serian de 96 horas si cada historia se hiciera de forma consecutiva, terminando una y siguiendo con la otra, sin aplicar las restricciones de dedicación y experticia, este valor corresponde al valor máximo posible.

Tabla 2: Desarrolladores definidios para el caso de estudio hipotetico

Nombre	Idenficador	Dedicación	Nivel de Experticia
D1	D1	1	1- Senior
D2	D2	0.9	1- Senior
D3	D3	1	1.5 - Junior
D4	D4	1	2 – Novato

Fuente: Elaboración propia.

Tabla 3: Historias de usuario a implementar en la iteración

Id.	Descripción	Prioridad	Tiempo estimado (horas)	Estimación en puntos
H1	H1	1	10	2
H2	H2	2	8	1
H3	H3	3	24	3
H4	H4	4	8	1
H5	H5	5	10	2
H6	H6	6	30	4
H7	H7	7	6	1
Tiempo de duración de la iteración			96	14

Fuente: Elaboración propia.

Se generó una población de 100 individuos, en la figura 2 se pueden apreciar los 20 primeros individuos junto con su cromosoma y el valor de la función objetivo. Se generaron 50 hijos y 50 mutaciones aleatorias en 5 iteraciones, con este proceso se obtiene el resultado mostrado también en la figura 2. También se puede apreciar los 20 primero individuos de la población final. El algoritmo presenta convergencia y se obtiene un tiempo de desarrollo de 33 horas.

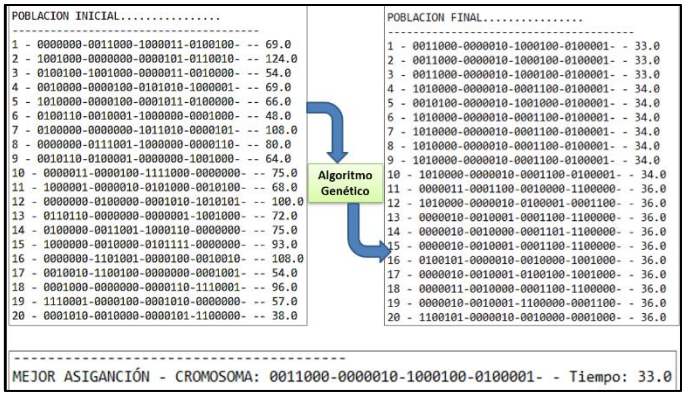

Figura 2: Resultados obtenidos para el caso de estudio hipotético.
Fuente: Elaboración propia.

La asignación para el cromosoma resultado se puede apreciar mejor en la tabla 4.

Tabla 4: Asignación optima obtenida con al algoritmo genético.

Nombre	Historias de Usuario	Tiempo (horas)
D1	H3 y H4	32
D2	H6	33
D3	H1 y H5	30
D4	H2 y H7	28
	Total	33

Fuente: Elaboración propia.

El tiempo se calculó teniendo en cuenta la dedicación y el nivel de experticia de los desarrolladores, usando las ecuaciones (1), (2), (3) y (4).

Un líder de proyecto o líder de desarrollo podría determinar y asignar las historias de usuario a los desarrolladores de acuerdo con su experiencia, dejando a los desarrolladores senior aquellas que son más complejas y a los junior o novatos las más sencillas. Teniendo en cuenta estas consideraciones la tabla 5 muestra una posible asignación para el caso hipotético de la tabla 2. La última columna de esta tabla muestra el tiempo estimado calculado para cada desarrollador.

Tabla 5: Asignación probable dada por un líder de proyecto

Nombre	Historias de Usuario	Tiempo estimado (horas)
D1	H6	30
D2	H3 y H7	33
D3	H1 y H5	30
D4	H2, H4	32
Tiempo de duración la iteración		33

Fuente: Elaboración propia.

Se puede apreciar que los resultados obtenidos por el algoritmo genético iguala a lo propuesto por un líder de desarrollo, siendo el propuesto en en este trabajo un método automático. Por lo tanto, es viable y acertado. Se deba ahora validar en un caso más complejo y de la vida real.

5.1. Evaluación del algoritmo en un caso de estudio de la vida real

El alcance de este trabajo de investigación fue realizar una prueba de concepto para veririfar que el algoritmo genético es viable para

optimizar la planificación de la iteración, comparando los resultados en unos casos de estudio hipotéticos y con un caso de estudio de la vida real, como trabajo futuro se pretende realizar un análisis comparativo con otros métodos similares resultados de la revisión de literatura, como también usar casos de la vida real más complejos con mas historias de usuario y desarrolladores.

Para validar el método de planificación de la iteración propuesto, se realizó un trabajo conjunto con la empresa de desarrollo de software Bits America que realiza proyectos para una de las empresas de telecomunicaciones más importantes de Colombia. Se tomó la información de la planificación de una iteración de uno de sus proyectos y se comparó con los resultados arrojados por el algoritmo genético propuesto.

La asignación de las historias de usuario a los desarrolladores las realiza el líder del proyecto, que en este caso es un profesional en Ingeniería de Sistemas, con especialización, estudiante de maestría y certificado en gestión de proyecto por el PMI, considerandose como una persona experta y con experiencia en proyectos de desarrollo de software. La tabla 6 muestra las características de los desarrolladores que hacen parte del proyecto.

Tabla 6: Desarrolladores definidios para el caso de estudio real.

Nombre	Idenficador	Dedicación	Nivel de Experticia
Raul Sanchez	D1	1	1.5 – Junior
Alejandro Villegas	D2	1	1 – Senior

Fuente: Elaboración propia.

Las hisotiras de usuario que se seleccionaron para el realizar este caso de estudio corresponden a una iteración real de un proyecto de desarrollo real de esta empresa, esas historias de usuario se pueden apreciar en la tabla 7.

La empresa Bits Americas tiene un mapeo de puntos de historia y horas de desarrollo y los definen de acuerdo con la siguiente relación:

2 puntos entre 1 y 3 horas de desarrollo
3 puntos entre 4 y 8 horas de desarrollo
5 puntos entre 9 y 15 horas de desarrollo
8 puntos entre 16 y 26 horas de desarrollo
13 puntos entre 27 y 40 horas de desarrollo
21 puntos más de 40 horas de desarrollo

Por este motivo se tiene una asignación diferente entre entre la estimación de puntos y el tiempo de desarrollo. Usando metodologías ágiles los equipos de desarrollo son libres de establecer este tipo de relaciones. El líder del proyecto corresponde al Scrum Master, la asignación de tareas y responsabilidades la realizan en conjunto con el equipo de desarrollo.

Tabla 7: Historias de usuario a implementar en la iteración.

Id.	Descripción	Prioridad	Tiempo estimado (horas)	Estimación en puntos
991	Integración SDK Braze	1	26	8
992	Deeplinks en registro de usuario	2	10	5
926	Mejoras de autocreación	3	24	8
998	Modificación diseño de bloque descarga de factura	4	16	8
857	Permitir cambio de wifi en equipos doblebanda	5	34	13
652	Consulta de visitas con usuario anómino	6	34	13
		Total	144	55

Fuente: Elaboración propia.

La tabla 8 presenta la asignación realizada por el líder del proyecto en conjunto con su equipo de desarrollo.

Tabla 8: Asignación realizada por el líder del proyecto

Nombre	Historias de Usuario	Tiempo (horas)
D1	991, 926 y 998	99
D2	992, 857 y 562	78
	Total	99

Fuente: Elaboración propia.

Se puede apreciar que cada desarrollador tiene asignadas 3 historias de usuario, el tiempo total de desarrollo de la iteracipon son 99 horas (66 horas multiplicadas por 1.5 del nivel de experticia), siendo este el mayor valor entre los dos desarrolladores.

Ahora en la figura 3, se pueden observar los resultados obtenidos por el método propuesto en este trabajo de investigación de forma automática usando algoritmos genéticos. Se muestra la población inicial, la población final y el resultado obtenido con el algoritmo genético.

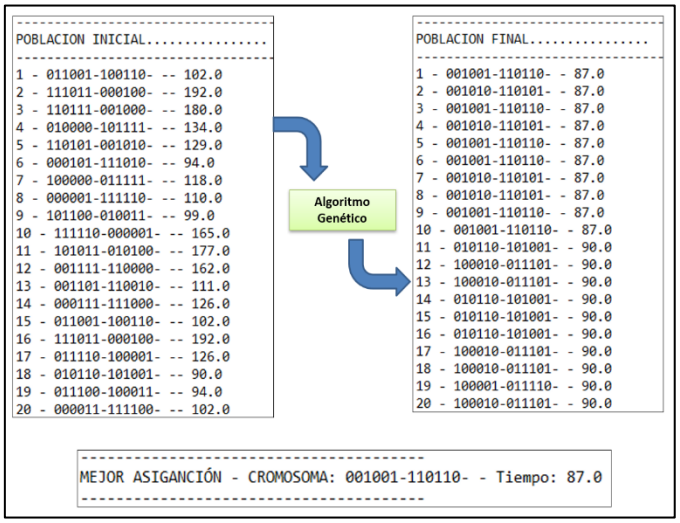

Figura 3: Resultados obtenidos para el caso de estudio real.
Fuente: Elaboración propia.

La asignación para el cromosoma resultado se puede apreciar mejor en la tabla 9.

Tabla 9: Asignación optima obtenida con al algoritmo genético.

Nombre	Historias de Usuario	Tiempo (horas)
D1	926 y 652	87
D2	991, 992, 998 y 857	86
	Total	87

Fuente: Elaboración propia.

Se puede apreciar que la distribución es más equitativa, la diferencia en horas de trabajo entre el desarrollador junior y el senior es sólo de una hora, teniendo en cuenta el nivel de experticia asignado. En la asignación realizada por el experto la diferencia es más amplia. En ocasiones la asignación de historias de usuarios complejas o que impliquen un grado de responsabilidad mayor se asignan a los desarrolladores con más experiencia y mejor capacitados, en este caso, según la estimación en puntos dadas por el equipo de desarrollo las historias mas complejas corresponden a las identificadas con los números 857 y 652. Estas fueron asignadas por el experto al desarrollador senior. El algoritmo genético propuesto las divide una para cada desarrollador y de esta forma se obtiene una mejor distribución obteniendo un menor tiempo estimado de desarrollo.

En cuanto al número de historias de usuario, la asignación realizada por el experto es más equitativa, porque asigna 3 historias a

cada desarrollador, mientras la asignación realizada por el algoritmo genético asigna 4 al desarrollador senior y 2 al desarrollador junior.

A simple vista se puede concluir que el algoritmo genético y el método propuesto es mejor o supera a la asignación realizada por el experto ya que obtiene un tiempo total de desarrollo menor. Este caso de estudio corresponde a un proyecto muy sencillo, sólo con dos desarrolladores y 6 historias de usuario, para poder generalizar y afirmar que el método automático de planificación de la iteración funciona mejor que la visión de un experto se deben realizar más pruebas con proyectos más complejos y con mayor número de desarrolladores, incluyendo también otras características de la historia de usuario como son la estimación en puntos y las dependencias.

Con este trabajo de investigación se pudo demostrar que los algoritmos genéticos pueden mejorar la asignación de las tareas de desarrollo en el contexto del desarrollo ágil, lo propuesto en este trabajo de investigación es una prueba de concepto positiva que permite seguir trabajando en este método para poder formalizarlo y validarlo experimentalmente en un trabajo futuro.

VI. CONCLUSIONES

Se demostró con dos casos de estudio uno hipotético y otro de la vida real que por medio de un algoritmo genético se puede mejorar significativamente la planificación de la iteración teniendo en cuenta las características propias del equipo de desarrollo (nivel de experticia y dedicación al proyecto) y de los requerimientos planteados como historias de usuario (tiempo de desarrollo y puntos estimados). Se compararon los resultados obtenidos por el algoritmo genético contra la planificación hecha por el líder del proyecto y/o equipo de desarrollo. En el caso de estudio hipotético, el resultado obtenido por el algoritmo fue el mismo que la planificación propuesta por el líder del proyecto; en el caso de estudio de la vida real el algoritmo genético encuentra una mejor planificación, reduciendo considerablemente el tiempo estimado de desarrollo en un 12% (de 99 horas a 87 horas).

La aplicación de técnicas automáticas y optimizadas para la asignación de tareas permite que esta se haga de una manera más efectiva, teniendo en cuenta las prioridades del proyecto, los recursos disponibles, las características de los desarrolladores y los requerimientos, en contraste con la asignación manual, que muchas veces puede ser imprecisa, a prueba y error, acomodada a la visión del líder del proyecto. El éxito del proyecto depende de la correcta asignación de las tareas de desarrollo, si se realizan en menor tiempo se pueden ahorrar costos y tener en el mercado la solución software en un menor tiempo. A mayor tiempo de desarrollo los costos son más altos, de ahí radica la importancia del método automático propuesto.

El uso de los algoritmos genéticos permiten optimizar tareas complejas automatizandolas, se demostró que el método propuesto funciona adecuadamente bajo las consideraciones establecidas, donde las historias de usuario no tienen dependencias, como trabajo futuro se pretende ampliar el algoritmo para contemplar las dependencias entre las historias de usuario, también se pretende utilizar otras técnicas de inteligencia artificial usadas para la optimización de funciones como son las colonias de hormigas, colonia de abejas o enjambres, haciendo un comparativo con el algoritmo propuesto.

Los casos de estudio propuestos son sencillos, permitieron hacer la prueba de concepto. Como trabajo futuro se pretende usar una base de datos de proyectos reales de la empresa Bits Americas, para comparar los resultados obtenidos en la planificación realizada por ellos, con el algoritmo genético, también se pretende evaluar de forma experimental con métodos similares.

VII. REFERENCIAS

[1] M. Tanveer, "Agile for large scale projects — A hybrid approach," in *2015 National Software Engineering Conference (NSEC)*, 2015, pp. 14–18.

[2] K. Beck *et al.*, "Manifiesto por el Desarrollo Ágil de Software," 2001. [Online]. Available: http://agilemanifesto.org/iso/es/manifesto.html. [Accessed: 29-Apr-2017].

[3] Versionone Enterprise, "13 Anual State of Agile Report," 2018. [Online]. Available: http://stateofagile.com/#ufh-i-521251909-13th-annual-state-of-agile-report/473508. [Accessed: 07-Jun-2019].

[4] proyectosagiles.org, "Proyectos Ágiles – La web de Scrum en español para la difusión de la gestión ágil de proyectos," 2017. [Online]. Available: https://proyectosagiles.org/. [Accessed: 13-Feb-2017].

[5] K. Schwaber and M. Beedle, *Agile Software Development with Scrum*. Prentice Hall, 2002.

[6] C. J. Parada, M. P. Rojas Puentes, and F. H. Vera-Rivera, "Study of the use of agile methodologies in the development of software construction projects in Colombia," *IOP Conf. Ser. J. Phys. Conf. Ser.*, vol. 1126, p. 12056, 2018.

[7] A. Navarro Cadavid, J. D. Fernández Martínez, and J. Morales Vélez, "Revisión de metodologías ágiles para el desarrollo de software," *Prospect. ISSN-e 2216-1368, Vol. 11, Nº. 2 (julio-diciembre), 2013, págs. 30-39*, vol. 11, no. 2, pp. 30–39, 2013.

[8] M. P. Rojas Puentes, M. F. Mora Méndez, B. Chacón, and S. M. Romero, "Estimation metrics in software projects," *J. Phys*, p. 12050, 2018.

[9] R. Mas'ad, R. Ñanculef, and H. Astudillo, "BlackSheep: Dynamic effort estimation in agile software development using machine learning," *XXII Ibero-American Conf. Softw. Eng. CIbSE 2019*, pp. 16–29, 2019.

[10] Á. Szke, "Conceptual scheduling model and optimized release scheduling for agile environments," *Inf. Softw. Technol.*, vol. 53, no. 6, pp. 574–591, 2011.

[11] M. Cohn, *Agile Estimating and Planning*. New York, NY, USA: Prentice Hall, 2005.

[12] D. Ameller, C. Farré, X. Franch, and G. Rufian, "A survey on software release planning models," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016, vol. 10027 LNCS, pp. 48–65.

[13] J. Holland, *Adaptation in natural and artificial systems*. Michigan: University of Michigan Press, 1975.

[14] F. Herrera, M. Lozano, and J. L. Verdegay, *Algoritmos Genéticos: Fundamentos, Extensiones y Aplicaciones*. ProQuest, 1995.

[15] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Q.*, vol. 28, no. 1, pp. 75–105, 2004.

[16] D. Greer and G. Ruhe, "Software release planning: an evolutionary and iterative approach," *Inf. Softw. Technol.*, vol. 46, no. 4, pp. 243–253, Mar. 2004.

[17] G. Ruhe and M. O. Saliu, "The art and science of software release planning," *IEEE Softw.*, vol. 22, no. 6, pp. 47–53, Nov. 2005.

[18] V. H. Escandon Bailon, H. Cervantes Maceda, and A. García Nájera, "Aplicación de un algoritmo genético multiobjetivo para la replaneación de liberaciones en proyectos ágiles de software Application of a Multi-Objective Genetic Algorithm to the Release Replanning in Software Agile Projects," *Res. Comput. Sci.*, vol. 148, no. 8, pp. 199–213, Apr. 2019.

[19] M. R. Karim and G. Ruhe, "Bi-objective genetic search for release planning in support of themes," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, vol. 8636 LNCS, pp. 123–137.

[20] PMI, *Guía de los fundamentos para la dirección de proyectos*

(guía del PMBOK®). 2013.

[21] G. Gbegnedji Castaño, "What is Project Management? | Project Manager's Essential Guide, by Gladys Gbegnedji," 2012. [Online]. Available: https://whatisprojectmanagement.wordpress.com/. [Accessed: 01-May-2017].

[22] S. Sheuly and K. : Smolander, *A SYSTEMATIC LITERATURE REVIEW ON AGILE PROJECT MANAGEMENT*. LAPPEENRANTA UNIVERSITY OF TECHNOLOGY, 2013.

[23] Y. M. Malgwi and N. V Blamah, "Multi-Agent Based Agile (XP) Software Development Process Scheduling Model," *Int. J. Pure Appl. Sci. Technol*, vol. 29, no. 2, pp. 54–63, 2015.

[24] B. Nuseibeh and S. Easterbrook, "Requirements engineering: a road map," in *Proceedings of the conference on The future of Software engineering - ICSE '00*, 2000, pp. 35–46.

[25] M. Cohn, *Agile Estimating and Planning*. New York, NY, USA, 2005.

[26] S. Palmer and M. Felsing, *A Practical Guide to Feature-Driven Development*. Person Education S.A., 2001.

[27] B. Anda, H. Dreiem, D. I. K. Sjøberg, and M. Jørgensen, "Estimating Software Development Effort Based on Use Cases — Experiences from Industry," vol. 2185, Springer Berlin / Heidelberg, 2001, pp. 487–502.

[28] N. Fenton and J. Bieman, *Software Metrics: A Rigorous and Practical Approach, Third Edition*, Third edit. Florida, USA: CRC Press Inc., 2014.

[29] C. Vidal Juan and P. O. Letelier Torres, "Gestión de proyectos de software desde una perspectiva tradicional y una ágil: contrastando PMBOK con los métodos ágiles," Universidad Politecnica de Valencia, Valencia, 2019.

[30] P. Fitsilis, "Comparing PMBOK and agile project management software development processes," in *Advances in Computer and Information Sciences and Engineering*, 2008, pp. 378–383.