Universidad de las Ciencias Informáticas

ALGORITMO PARA LA OPTIMIZACIÓN DE ESTADOS EN LA MEJORA DE PROCESOS DE SOFTWARE

Trabajo final presentado en opción al título de Ingeniero en Ciencias Informáticas

Autor: Alejandro Perdomo Vergara

Tutora: MSc. Ana Marys Garcia Rodríguez

Co-tutor: Ing. Yordanis Milanés Zamora

La Habana, Cuba

2016

Declaración de autoría

Declaro ser autor del presente trabajo de diploma y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste se firma la presente a los 23 días del mes de junio del año 2016.

Alejandro Perdomo Vergara

_____

Firma del autor

MSc. Ana Marys Garcia Rodríguez

_____

Firma de la tutora

Ing. Yordanis Milanés Zamora

_____

Firma del co-tutor

Agradecimientos

Agradezco primeramente a la Revolución y a la educación cubana por la oportunidad de llegar hasta la educación superior.

A mi familia y amigos por darme el apoyo necesario y el aliento para seguir siempre adelante. Especialmente a mi madre porque a ella le debo no solo mi vida sino todo lo que soy y todo lo que un día puedo llegar a ser.

A todos aquellos que de una manera u otra contribuyeron a la investigación y desarrollo de este trabajo de diploma. Especialmente a la MSc. Ana Marys Garcia Rodríguez por el arduo trabajo y por la amistad.

Por último y no menos importante a la Universidad de las Ciencias Informáticas por todos los cambios que hizo en mi ser.

Resumen

Actualmente se refleja una tendencia a la definición y mejora continua de los procesos en el desarrollo de software, con el propósito de alcanzar una organización estratégica para el logro de los objetivos. En este sentido la Mejora de Procesos de Software ha jugado un papel fundamental, no obstante, se refleja un número considerable de fracasos, lo cual se atribuye a que no se considera el estado de la organización integralmente respecto a su entorno socio-cultural. Diversas investigaciones han dirigido sus esfuerzos a definir una evaluación integral de las organizaciones, considerando los factores críticos que influyen en el éxito, previo a la mejora de procesos; sin embargo, no se provee a las organizaciones de posibles estados de mejora a alcanzar para obtener mejores resultados sobre la base de la reutilización de las experiencias adquiridas en torno a los factores que influyen en la mejora. Lo antes expuesto conlleva a que se dediquen recursos y tiempo sin conocimiento previo de hacia dónde dirigir los esfuerzos. La presente investigación tiene como objetivo desarrollar un algoritmo genético, considerando las experiencias adquiridas entorno a las características organizacionales, para contribuir a la toma de decisiones en las iniciativas de Mejora de Procesos de Software. La implementación del mismo provee de escenarios de mejora a alcanzar por las organizaciones como apoyo a la toma de decisiones a partir de su estado inicial y las buenas prácticas que puede aplicar para mejorarlo.

Palabras claves: buenas prácticas, Factores Críticos de Éxito, Mejora de Procesos de Software, optimización de estados, toma de decisiones.

Abstract

A tendency in software development to process definition and continuous improvement, with the aim of achieving a strategic organization for the achievement of the objectives is currently reflected. In this way, the Software Process Improvement has played a main role, however, a considerable number of failures is reflected, which is attributed to not take in fully consideration the state of the organization regarding its socio-cultural environment. Several studies have directed its efforts to define a comprehensive assessment of the organizations, considering the critical factors that influence success prior to process improvement; however, organizations aren't provided of possible improvement states to achieve in order to aim best results based on the reuse of learned lessons about factors that influence in the improvement. The foregoing brings to devote time and resources without prior knowledge of where to direct efforts. This research aims to develop a genetic algorithm, considering the experiences acquired around organizational characteristics, to contribute to decision making in initiatives of Software Process Improvement. Its implementation provides improved states to achieve by organizations as a support to decision-making from its initial state and the good practices which can apply to improve it.

Keywords: critical success factors, decisions making, good practices, Software Process Improvement, states optimization.

Índice

Introducción

En la actualidad se visualiza un incremento en la agilidad de ejecución de los procesos sociales y facilidades en el trabajo de las personas, lo cual constituye consecuencia directa de los cambios significativos sufridos por la sociedad como resultado del creciente desarrollo de las Tecnologías de la Información y las Comunicaciones. El avance de dichas tecnologías, ha promovido cambios y la creación de nuevos conocimientos, a un ritmo tal, que las organizaciones que no cuentan con suficiente sabiduría para entenderlos, preservarlos e incrementarlos, así como la tenacidad para emprenderlos, pueden quedar irreversiblemente excluidas del desarrollo (CASTRO 2013). En este ámbito, el desarrollo de la industria del software se ha convertido en un factor dominante en la economía del mundo industrializado (PRESSMAN 2010). Es por ello que diversas instituciones y países han enfocado sus esfuerzos en desarrollar esta industria con el objetivo de elevar la calidad de sus procesos y de esta manera proporcionar oportunidades de mejoras en el ámbito social, incrementar en retorno sus ingresos y posicionarse en un mercado internacional que exige cada vez más, productos de mejor calidad (GARCIA *et al.* 2016).

Estudios realizados por Standish Group y publicados en el Chaos Manifiesto 2015 (STANDISH-GROUP 2015), evidencian que a pesar de los beneficios que proporciona la industria del software, no existe una correspondencia entre la demanda continua de informatización de la sociedad y la ejecución eficiente y eficaz de los procesos. Reflejando que entre los años 2011 y 2015 aproximadamente el 29% de los proyectos resultó exitoso, mientras que el 52% presentó problemas de retrasos y el 19% resultó fallido. Durante varias décadas se han concebido diversas investigaciones referentes a los problemas en el desarrollo de software (BAUER 1972; BOEHM 1976; HUMPHREY 1995; PRESSMAN 2010; SOMMERVILLE 2007), las cuales convergen en la necesidad de aplicación de métodos más efectivos de ingeniería de software, siguiendo una perspectiva sistemática, disciplinada y cuantificable, teniendo como centro el proceso y como base un enfoque de calidad (IEEE 1990).

Las investigaciones previamente mencionadas, reflejan la necesidad de establecer un desarrollo dirigido por procesos para incrementar la eficiencia en el desempeño de las organizaciones desarrolladoras de software; así como lograr que los procesos implantados se desarrollen con la calidad requerida. Además, se establece que para el cumplimiento de los objetivos de la organización desde un enfoque de calidad, no solo es importante establecer una perspectiva dirigida a procesos, sino también la definición de iniciativas que

contribuyan a la mejora continua de los mismos, en aras de alcanzar una madurez estratégica en la organización y en consecuencia proporcionar servicios de calidad en un mercado competitivo. Las investigaciones poseen un auge destacado, tanto desde el punto de vista teórico como en su ejercicio práctico, enfocando la ingeniería y la calidad de software en la integración y mejora de los procesos de software (BOAS *et al.* 2010) con el fin de fortalecer esta industria.

Varios autores han reflejado en sus investigaciones (ASHRAFI 2003; BASILI *et al.* 2002; GARCIA *et al.* 2016; PINO, FRANCISCO J. *et al.* 2008; TRUJILLO *et al.* 2013; ZAHRAN 1998) (PINO, FRANCISCO J. *et al.* 2008) la importancia del papel que juega la MPS al elevar la madurez y capacidad de los procesos mediante la introducción de buenas prácticas. Los proyectos de MPS se centran en mejorar el rendimiento, la utilidad y la efectividad de los procesos de una manera disciplinada (ASHRAFI 2003); ventaja que no siempre puede ser observada a corto o mediano plazo, pues la implantación de buenas prácticas es una faena cuyos resultados pueden tardar años en visualizarse.

Instituciones y comunidades científicas han elegido la aplicación de modelos, normas, guías y estándares en función de la MPS. En desventaja, los modelos, guías y estándares explican qué hacer para establecer una mejora de los procesos en la organización, pero no establecen el cómo ejecutarla.

Estudios señalan que los modelos, en general, no consideran las situaciones que se pueden encontrar en organizaciones que desean implementar una MPS (ALLISON 2010; BOAS *et al.* 2010; CATTANEO *et al.* 2001; DOUNOS and BOHORIS 2010; LAPORTE and TRUDEL 1998; MOITRA 1998; NGWENYAMA 2003; STELZER and MELLIS 1999). Diversas publicaciones (BABAR and NIAZI 2008; MÜLLER *et al.* 2010; NIAZI *et al.* 2010; PINO, FRANCISCO J. *et al.* 2008) derivan la aplicación de cambios culturales y organizativos como una necesidad para el éxito de las iniciativas MPS, los cuales son complejos de abordar y demandan grandes inversiones de recursos y tiempo. En tal sentido algunos países se han aventurado en adaptaciones propias de estos modelos a su contexto, ejemplo de ello son: MoProSoft en México (NYCE 2011), MPS. Br en Brasil (SOFTEX 2012), la propuesta Iberoamericana Competisoft (COMPETISOFT 2008) y en Cuba el Modelo de Calidad para Desarrollo de Aplicaciones Informáticas (MCDAI) (RAMÍREZ and SANTANA 2013) creado por el Centro Nacional de Calidad de Software (CALISOFT) de conjunto con la Universidad de las Ciencias Informáticas (UCI).

Se considera de manera general que los programas de mejora definidos en las organizaciones, no organizan sus objetivos estratégicos en base a las necesidades reales y los aspectos sociales de las mismas. Estudios sobre los resultados de la aplicación de los programas de MPS en las organizaciones (FORRADELLAS *et al.* 2005; NGWENYAMA 2003; TRUJILLO *et al.* 2013), muestran un gran número de fracasos que ascienden hasta un 70%, lo cual es consecuencia en parte de que no se contemplen en los programas de mejora el estado real de las organizaciones y sus particularidades, lo cual determina un punto de partida distinto para el programa y condiciona sus resultados (TRUJILLO *et al.* 2014). En pos de adaptar los programas de MPS a las características particulares de las organizaciones, varios autores han realizado investigaciones que tributan a la propuesta de factores influyentes en el diseño y la ejecución de la MPS. Dichos factores se refieren a los aspectos sociales y han sido obtenidos a partir de: literatura consultada (DOUNOS and BOHORIS 2010), entrevistas y encuestas a consultores de MPS (BOAS *et al.* 2010; SANTOS *et al.* 2010), datos y experiencias acumuladas de la ejecución de MPS (BOAS *et al.* 2010; SANTOS *et al.* 2010), estudio de casos (ALLISON 2010) y diagnósticos al iniciar la MPS (TRUJILLO *et al.* 2014). Los Factores Críticos de Éxito (FCE) son influyentes de forma negativa o positiva en la MPS y se ha podido inferir que su uso en función de los contextos organizacionales contribuye al éxito de los programas de MPS (DOUNOS and BOHORIS 2010; MONTONI and ROCHA 2010). Para ello es importante considerar que el contexto varía en dependencia de la organización, desde los objetivos estratégicos y los aspectos sociales hasta las necesidades reales en función de los recursos con que cuenta (GARCIA *et al.* 2016).

Existe una visualización de los avances sobre el tratamiento de los FCE para valorar el estado de las organizaciones frente a la MPS (GARCIA *et al.* 2016; NIAZI *et al.* 2010; NIAZI *et al.* 2006; TRUJILLO *et al.* 2013). A pesar de ello, persisten insuficiencias asociadas a la reutilización del conocimiento adquirido en este entorno para establecer propuestas de escenarios a las organizaciones que reflejen una mejora cualitativa respecto a su estado inicial para enfrentar la MPS. Además, no se realiza un análisis del grado en que las buenas prácticas inciden sobre el comportamiento de los FCE. Resulta engorroso realizar el procesamiento de la información dado el gran cúmulo de elementos a considerar con vista a la propuesta de escenarios de mejora, por ello es esencial la transformación de la información resultante de las experiencias en la MPS, en conocimiento útil para la toma de decisiones respecto hacia dónde dirigir los esfuerzos mediante la aplicación de técnicas de inteligencia artificial (IA).

El análisis anterior conduce al siguiente diseño de investigación:

Problema de la investigación:

¿Cómo determinar escenarios de mejora para contribuir a la toma de decisiones en las iniciativas de mejora de procesos de software?

Objeto de estudio de la investigación:

Mejora de procesos de software.

Campo de acción:

Optimización de estados en la mejora de procesos de software.

Objetivo general de la investigación:

Desarrollar un algoritmo genético, considerando las experiencias adquiridas entorno a las características organizacionales, para contribuir a la toma de decisiones en las iniciativas de mejora de procesos de software.

Objetivos específicos:

1. Definir el marco teórico de la investigación mediante el análisis de los principales referentes teóricos para el desarrollo de la solución.
2. Diseñar e implementar el proceso y el componente de optimización de estados para contribuir a la toma de decisiones en las iniciativas de mejora de procesos de software.
3. Validar la solución propuesta mediante la realización de un pre-experimento.
4. Verificar la satisfacción del cliente mediante la aplicación de la técnica Iadov.

Entre los ***métodos de trabajo científico*** utilizados se destacan los siguientes:

Métodos teóricos:

- El método **histórico-lógico** para el análisis crítico de los trabajos previos con el objetivo de establecer un punto de referencia para la propuesta resultante.
- El método **inducción-deducción** para la identificación de la problemática y sus variantes de solución.
- El método **hipotético-deductivo** para la elaboración de la idea a defender y la propuesta de la línea de trabajo de la investigación.

- El **analítico-sintético** para la descomposición del problema de investigación en elementos que permitan su profundización, para luego sintetizarlos en la solución propuesta.

Métodos empíricos:

- El **experimental** para comprobar la utilidad de los resultados obtenidos a partir de la implementación del algoritmo.

Métodos cualitativos y cuantitativos:

- **Encuestas** y **técnica Iadov** para conocer el nivel de satisfacción de los clientes con el algoritmo desarrollado.
- **Pre-experimento** para verificar la solución propuesta.

La **estructura del trabajo** queda constituida en tres capítulos:

En el ***Capítulo 1 - Marco teórico referencial*** se presentan elementos teóricos-conceptuales vinculados con la problemática y los nuevos retos en la MPS, los FCE que influyen en la misma y las técnicas de IA, algoritmos genéticos para el procesamiento y reutilización de experiencias.

En el ***Capítulo 2 - Algoritmo de optimización de estados en la MPS*** se presentan los artefactos resultantes del diseño e implementación del algoritmo genético para la optimización de estados en la MPS, así como el pseudocódigo que describe al algoritmo.

En el ***Capítulo 3 - Validación de la solución*** se reflejan los resultados constatados de la aplicación de la técnica Iadov para validar la satisfacción del cliente respecto a la aplicabilidad y utilidad de la solución; así como de la realización de un pre-experimento para corroborar la funcionalidad del algoritmo desarrollado.

Capítulo I - Marco teórico referencial

Una necesidad esencial del perfeccionamiento de la producción de software en las organizaciones, es la reutilización de las buenas experiencias resultantes del proceso de desarrollo de software. Se persigue como objetivo minimizar en gran medida los indicadores de costo, tiempo y esfuerzo en la mejora continua de los procesos, lo cual favorece la obtención de productos y servicios con un grado de calidad que satisfaga las necesidades del cliente.

En el presente capítulo se describen la naturaleza y los conceptos asociados al problema para posteriormente abordar los campos asociados al objeto de estudio con el fin de determinar la técnica más adecuada para resolver la problemática propuesta. Por último, se profundiza en la posición asumida en consideración a las técnicas existentes para el desarrollo de la solución.

Naturaleza del problema

Mejora de procesos de software

En la actualidad el crecimiento de la industria del software destaca entre los sectores del mercado (PINO, FRANCISCO J. *et al.* 2009), debido al alza del papel de sus productos en las disímiles esferas de la sociedad. A pesar de ello, se plantea que no existe correspondencia de dichos resultados con la ejecución eficiente y eficaz de los mismos (PRESSMAN 2002; VIRTANEN *et al.* 2013). Standish Group publica en su informe de 2015 (ver Figura 1) que el número de proyectos exitosos constituye aproximadamente solo un 29%, mientras que el 52% presentan retrasos, y el 19% de los proyectos fracasan (STANDISH-GROUP 2015).

	2011	2012	2013	2014	2015
ÉXTITOS	29%	27%	31%	28%	29%
RETRASOS	49%	56%	50%	55%	52%
FRACASOS	22%	17%	19%	17%	19%

Figura 1: Chaos Manifesto 2015
Fuente: (STANDISH-GROUP 2015)

Del análisis de estos datos, se puede afirmar que, a pesar del incremento en la demanda de informatización de la sociedad, la producción de software aún contempla insuficiencias

que inciden sobre el desarrollo de productos (TRUJILLO 2014). Varios autores (MATHIASSEN and POUYA 2003; MATURRO 2010; PRESSMAN 2002) visualizan la MPS como un enfoque estructurado que posibilita a una organización de software mejorar continuamente sus capacidades para proporcionar servicios de calidad en forma competitiva. Por su parte, Ashrafi (ASHRAFI 2003) precisa de forma más certera que la MPS se centra en mejorar el rendimiento, la utilidad y la efectividad de los procesos de una manera disciplinada.

Uno de los principios de la MPS se basa en mejorar la madurez del proceso de desarrollo del software (LI *et al.* 2006) reflejado en su utilidad, rendimiento y efectividad, y como consecuencia de ello la calidad del producto de software (ASHRAFI 2003). Esta mejora consiste en modificar o depurar las prácticas que causan problemas y en contraposición incentivar las que proveen de buenos resultados (ZAHRAN 1998).

Según Trujillo significativas ventajas son arrojadas sobre la madurez de la organización cuando se institucionaliza una mejora de procesos de software. Sin embargo, resulta complejo y "requiere cambios culturales y organizativos que a veces son difíciles de abordar, dependiendo del contexto y afectando el resultado de la MPS". Es por ello que la autora al referirse a la MPS aporta el siguiente concepto, el cuál será adoptado en la presente investigación (TRUJILLO 2014):

> *La **Mejora de Procesos de software** es un proceso sistémico que con independencia del enfoque adoptado, requiere de cierto tiempo, recursos, medidas y las iteraciones para su aplicación efectiva y exitosa. Su objetivo es mejorar el rendimiento del proceso de desarrollo de software, a partir de desarrollar acciones que se manifiestan en modificaciones al proceso de desarrollo de software.*

Modelos de Mejora de Procesos de Software

El establecimiento de modelos, normas, guías y estándares es un factor importante en el aumento de la probabilidad de éxito a la hora de ejecutar los programas de mejora. Siendo reconocido entre ellos el Modelo IDEAL, llamado así por las fases que lo componen (en inglés): Iniciar, Diagnosticar, Establecer, Actuar y Aprender (KAUTZ *et al.* 2000) (MCFEELEY 1996). El Modelo IDEAL ha sido empleado por los países que hoy poseen más organizaciones certificadas por CMMI como Estados Unidos, China, India, Japón, Francia, Korea, Taiwán, además en Irlanda, la India e Israel, los cuales poseen una amplia reputación en la industria del software a escala global (GARCIA 2013).

Por otra parte destacan las normas ISO 15504 (ISO 2004), ISO 9000 (ISO 2005a) e ISO 25000 (ISO 2005b) desarrolladas por la Organización Internacional para la Estandarización (ISO por sus siglas en inglés) (ISO 1997), la Guía de Fundamentos para la Dirección de Proyectos (PMBok) (PMI 2013a) propuesto por el Instituto de Gestión de Proyectos (PMI por sus siglas en inglés) (PMI 2013b), la propuesta del Instituto de Ingenieros de Software (SEI por sus siglas en inglés) de la Universidad Carnegie Mellon (CARNEGIE-MELLON 2013) ampliamente difunda como Modelo de Capacidad de Madurez Integrada (CMMI por sus siglas en inglés) (CMMI 2016), entre otros.

Es válido destacar que estos modelos, guías y estándares especifican qué hacer para establecer una mejora de los procesos, pero no establecen el cómo ejecutarla. Lo que ha llevado a algunos países a tomar iniciativas en la creación de adaptaciones de estos modelos a su contexto particular (GARCIA 2013). Entre las adaptaciones destacan: MoProSoft de México (NYCE 2011), Competisoft, una propuesta Iberoamericana (COMPETISOFT 2008), MPS. Br en Brasil (CAVALCANTI and CHAVES 2008) y la creación del Instituto de Software Europeo (ESI por sus siglas en inglés) IT Mark (SINERTIC 2011).

Estudios sobre los resultados de la aplicación de los programas de MPS en las organizaciones (FORRADELLAS *et al.* 2005; NGWENYAMA 2003; TRUJILLO *et al.* 2013), muestran un gran número de fracasos que ascienden hasta un 70%. En pos de mejorar dichos resultados se han realizado varias investigaciones (ALLISON 2010; BOAS *et al.* 2010; DOUNOS and BOHORIS 2010; SANTOS *et al.* 2010; TRUJILLO *et al.* 2014) que de conjunto llevaron a la identificación de factores que influyen en el éxito o fracasos de los programas de mejora. Dichos factores son conocidos como FCE.

Conceptos asociados al problema

FCE

En muchas ocasiones la ejecución exitosa de los programas de MPS se ve obstaculizada por riesgos y barreras que resultan difíciles de erradicar o combatir. Esto se debe a lo engorroso de la realización de un plan de acción que pueda contrarrestar cualquier influencia negativa. Varias organizaciones han mostrado lecciones aprendidas, que ayudan a nutrirse de sus experiencias e identificar a las acciones que acortan la influencia negativa de un factor. Estas acciones se han convertido en buenas prácticas (NIAZI *et al.* 2010). Teniendo en cuenta lo anterior Trujillo (TRUJILLO 2014) define un conjunto de FCE junto a sus medidas base (ver anexo 1) y planeta las definiciones siguientes:

*Los **factores críticos de éxito**, son los factores que se consideran determinantes en el éxito del programa de MPS en una organización.*

*Las **buenas prácticas**, son acciones que disminuyen la influencia negativa de un factor.*

Se hace necesario señalar que el no establecimiento de una alineación entre la evaluación de los FCE y la reutilización de experiencias obtenidas por las organizaciones como reflejo de los resultados obtenidos al iniciarse en la MPS; ha traído como consecuencia que las organizaciones empleen un mayor número de recursos humanos y financieros, de igual manera mayor esfuerzo y tiempo para iniciarse en la MPS (GARCIA 2013). A decir de García Rodríguez “esto se refleja en que no se tienen en cuenta las buenas prácticas, lecciones aprendidas y resultados alcanzados por otras organizaciones donde los FCE pudieran comportarse de manera similar debido a la semejanza de sus culturas”.

Al analizar el planteamiento anterior se hace notable la necesidad de medir el impacto de los FCE en las organizaciones a partir de la reutilización de experiencias. Esta actividad aporta una influencia positiva sobre la actividad gerencial durante el proceso de toma de decisiones, pues ayuda a mitigar el número de recursos humanos, financieros, de esfuerzo y de tiempo a la hora de iniciarse en la MPS.

Toma de decisiones en las organizaciones

Tomar decisiones se encuentra entre las obligaciones que impone el oficio gerencial. Respecto a lo cual Schein, plantea: "la toma de decisiones es el proceso de identificación de un problema u oportunidad y la selección de una alternativa de acción entre varias existentes, es una actividad clave en todo tipo de organización" (DÍAZ DUARTE 2005; SCHEIN 1988).

La identificación de todas las alternativas posibles, el pronóstico de las consecuencias de las mismas y su evaluación según los objetivos y metas trazados son postulados intrínsecos dentro del proceso de toma de decisiones. Para ello, se requiere primeramente tener información actualizada sobre las alternativas aprovechables en el presente o sobre las que se deben considerar. Además, es necesario poseer información sobre el futuro: cuáles son las consecuencias de actuar según cada una de las diversas opciones. Luego, es indispensable la información sobre cómo llevar el estado actual al futuro; conocer los valores y las preferencias que se deben utilizar para elegir, entre las alternativas, que según

los criterios establecidos, llevan de forma más factible a los resultados deseados (CASE 2012; CHOO and DÍAZ 1999).

Teniendo en cuenta lo antes expuesto, puede definirse la toma de decisiones como una actividad imprescindible en las organizaciones, con un significado especial para todos los niveles, pues es parte fundamental e inherente a todas las demás actividades de la empresa. En ocasiones, en la realización de esta actividad, no se toman en cuenta las tradiciones, hábitos, costumbres, la propia intuición y experiencia de un directivo en la forma de solucionar los problemas.

El uso de modelos para evaluar las organizaciones al iniciar la MPS, contribuye a esta actividad al proveer de recomendaciones a los directivos para la toma de decisiones. La experiencia de ello dentro de la UCI es el modelo Si.MPS.Cu, el cual permite obtener los criterios para valorar una organización al iniciar la MPS. Para ello usa indicadores, métricas y un razonamiento basado en casos. Además, toma en cuenta el análisis de la influencia de los FCE, la identificación de riesgos y las barreras que impiden la mejora. Este modelo recomienda buenas prácticas para disminuir la influencia negativa de los FCE, siendo sus salidas, recomendaciones de gran importancia para los directivos (TRUJILLO 2014).

Un proceso clásico en la toma de decisiones es la optimización, el cual comprende la formulación del problema, el modelado del mismo, la optimización del modelo y la implementación de la solución (TALBI 2009).

Campos asociados al objeto de estudio

Optimización

Según la Real Academia de la Lengua Española (RAE) optimización es “acción y efecto de optimizar”, mientras que optimizar es “buscar la mejor manera de realizar una actividad” (RAE 2016). Por otra parte, en el contexto científico la optimización es el proceso de tratar de encontrar la mejor solución posible para un determinado problema(VÉLEZ and MONTOYA 2007).

Para la presente investigación es importante considerar la optimización como un proceso. El mismo será un refinamiento continuo que si bien no retorna un resultado óptimo, tendrá como fin la obtención del mejor o mejores resultados alcanzables teniendo en cuenta las variables iniciales del problema en cuestión, logrando un resultado más cercano a lo realizable y no a lo idealizado.

Este proceso pude ser aplicado a diferentes campos, fundamentalmente al soporte de la toma de decisiones en diferentes entornos, entre ellos, a las organizaciones que se enfrentan a la MPS. A dichas organizaciones les resulta de gran utilidad, al iniciarse en el proceso de mejora, conocer hacia dónde dirigir los esfuerzos para la mejora, teniendo en cuenta su estado inicial, para lo cual pueden apoyarse en modelos como Si.MPS.Cu (TRUJILLO 2014) o en herramientas como Hefexto (GARCIA 2013).

En ocasiones los estados iniciales no permiten alcanzar el éxito en la MPS, sin embargo, aplicando un conjunto de buenas prácticas en las organizaciones se hace posible optimizar sus estados, lo cual lógicamente incrementa la probabilidad de éxito en el proceso de mejora.

Optimización de estados

Entre las definiciones que ofrece la RAE acerca del término “estado” refleja que es “situación o modo de estar …, en especial la situación temporal … cuya condición está sujeta a cambios” (RAE 2016). Por su parte una organización es una “asociación de personas regulada por un conjunto de normas en función de determinados fines” (RAE 2016).

Al analizar los conceptos anteriores, se asumen los siguientes conocimientos para la presente investigación:

> *Un **estado inicial** como: la situación o modo de estar de una organización en un momento de tiempo dado, cuya condición está sujeta a cambios.*
>
> *La **optimización de estados** como: el proceso mediante el cual se obtienen escenarios de mejora, tomando como punto de partida el estado inicial de la organización y las buenas prácticas que puede aplicar.*
>
> *Un **escenario de mejora** como: la situación o modo de estar que puede alcanzar una organización a partir de su estado inicial y el conjunto de buenas prácticas que puede aplicar para mejorarlo.*

Técnicas de optimización

La optimización combinatoria es una rama de la optimización cuyo dominio se compone de problemas de donde el conjunto de posibles soluciones es discreto o se puede reducir a un conjunto discreto. Su objetivo consiste en encontrar buenas soluciones (no necesariamente la óptima) al problema en un tiempo razonable (HERRERA 2006; OROZCO 2007).

Los problemas de optimización combinatoria son de difícil solución ya que suelen requerir agrupamientos, ordenaciones o asignaciones de un conjunto discreto de objetos que satisfagan ciertas restricciones. Además, se encuentran en muchas áreas de aplicación, y presentan una gran complejidad computacional, así, los algoritmos exactos son ineficientes o simplemente imposibles de aplicar. Suelen ser solubles mediante las meta-heurísticas que son una familia de algoritmos aproximados de propósito general. Estas suelen ser procedimientos iterativos que guían una heurística subordinada de búsqueda, combinando de forma inteligente distintos conceptos para explorar y explotar adecuadamente el espacio de búsqueda. Dentro de sus principales ventajas se encuentran las siguientes (HERRERA 2006):

- Son algoritmos de propósito general.
- Poseen gran éxito en la práctica.
- Son fácilmente implementables.
- Son fácilmente paralelizables.

Una posible clasificación de las meta-heurísticas es la siguiente (HERRERA 2006):

- Meta-heurísticas basados en métodos constructivos, los cuales parten de una solución inicial vacía y van añadiéndole componentes hasta construir una solución. Ejemplo de ello es la optimización basada en Colonias de Hormigas.
- Meta-heurísticas basadas en trayectorias, donde la heurística subordinada es un algoritmo de búsqueda local que sigue una trayectoria en el espacio de búsqueda. Parten de una solución inicial e iterativamente tratan de reemplazarla por otra solución de con mejor calidad. Algunos ejemplos de ello es la Búsqueda Local, el Enfriamiento Simulado, la Búsqueda Tabú, y el Búsqueda Local Iterativa.
- Meta-heurísticas basadas en poblaciones, donde el proceso considera múltiples puntos de búsqueda en el espacio que evolucionan en paralelo. Algunos ejemplos de ello son: Algoritmos Genéticos, Scatter Search y Algoritmos Meméticos.

Según Hertz y Kobler (HERTZ and KOBLER 2000), las técnicas utilizadas para resolver problemas complejos de optimización combinatoria han evolucionado progresivamente de métodos constructivos a métodos de búsqueda local y finalmente a algoritmos basados en poblaciones de los cuales se han desarrollado varios métodos. La Computación Evolutiva (CE) representa un amplio grupo de estas técnicas, dividida básicamente en dos ramas: los Algoritmos Evolutivos (AE) y la Inteligencia de Enjambres (IE).

Las técnicas de Computación Evolutiva, pertenecen a la categoría de métodos heurísticos basadas en los principios de la evolución natural (TOMASSINI 1995). Estas técnicas heurísticas podrían clasificarse en 4 categorías principales: los Algoritmos Genéticos (AG) (HOLLAND 1975), las Estrategias Evolutivas (EE) (RECHENBERG 1965), Programación Evolutiva (PE) (FOGEL 1966) y Programación Genética (PG) (KOSA 1992). Estas son una alternativa potente cuando el espacio de búsqueda es demasiado grande para realizar una búsqueda exhaustiva (PURIS and BELLO 2009). También se han denominado modelos computacionales bioinspirados, pues fueron desarrollados a partir de modelos existentes en la naturaleza. Otra técnica de optimización bioinspirada es la Optimización basada en Colonias de Hormigas (OCH) (MUÑOZ *et al.* 2008).

Al examinar las diferentes características de cada rama de la CE, se puede establecer una serie de diferencias entre las técnicas, siendo la más notoria la naturaleza centralizada de los AE, frente al comportamiento distribuido en IE. Otra diferencia se encuentra en la interacción con la función objetivo, donde en los AE corresponde al nivel de "desempeño" y en la IE por lo general corresponde al ambiente a ser explorado. A su vez, los agentes en los AE no pueden modificar su ambiente, mientras que en IE esta característica es fundamental (MUÑOZ *et al.* 2008).

Teniendo en cuenta que la problemática planteada no está asociada a búsquedas espaciales sino a la optimización de un estado inicial a partir de las experiencias almacenadas en una base de conocimientos, se considera relevante la aplicación de técnicas meta-heurísticas basadas en poblaciones. Específicamente el uso de algoritmos evolutivos donde el estado inicial representa un individuo y las experiencias almacenadas la población sobre la cuál realizar las búsquedas. Se hace necesario entonces establecer una comparación entre los diferentes AE con el fin de seleccionar el más adecuado, acorde a las necesidades planteadas por la problemática. En tal sentido la Tabla 1 establece una comparación entre PG, PE, EE y AG.

	Programación Genética	Programación Evolutiva	Estrategias Evolutivas	Algoritmos Genéticos
Representación del problema	Expresiones de LISP representadas	Números reales	Vector de reales + desviaciones estándar	Cadena binaria, Árboles, Valor

	habitualmente como árboles			entero, Valor Real
Operadores de Variación	Crossover, algo de mutación	Mutación	Mutación gaussiana y crossover aritmético (diferentes tipos)	Mutación y crossover
Métodos de selección	Diversos tipos de Selección	Diversos tipos de Selección	Diversos tipos de Selección	Diversos tipos de selección
Tipo de selección	Preservativa	Estática y extintiva	Estática y extintiva	Dinámica, preservativa y al vuelo
Campo natural de aplicación	Optimización estocástica	Optimización evolutiva	Optimización paramétrica	Optimización de atributos
Clave de la búsqueda	Operadores genéticos de recombinación (crossover)	Operadores genéticos de alteración (mutación)	Operadores genéticos de alteración (mutación)	Operadores genéticos de recombinación (crossover)

Tabla 1: Comparación de los Algoritmos Evolutivos
Fuente: (COELLO and ZACATENCO 2004; MATEOS ANDALUZ 2004; PETTIS 2005; ROSAS et al. *2000)*

Para el problema en cuestión se hace necesaria la optimización de los atributos de un individuo. Haciendo uso de la reutilización del conocimiento, se pretende que el resultado mantenga un margen de cercanía con el individuo original, lo que conlleva a la necesidad de una selección en la base de conocimientos proporcional al fitness de cada individuo. Basado en lo descrito en la Tabla 1 se considera para la presente investigación el uso de AG como técnica de optimización más apropiada para solucionar la problemática planteada.

Alternativa para solucionar el problema

Algoritmos genéticos

A principio de los 1960's John H. Holland plantea bajo el nombre de "planes reproductivos genéticos" (HOLLAND 1962) las bases de lo que se conoce hoy como AG. Los algoritmos evolutivos son métodos de búsqueda de soluciones inspirados en los procesos y mecanismos de la evolución biológica propuesta por Darwin, teniendo en cuenta la variedad

de los organismos vivos y su adaptación al medio (PADRÓN and MORA 2014). Estos usan operadores de cruzamiento, mutación y la selección, siendo el cruzamiento el operador principal (COELLO and ZACATENCO 2004). La Figura 2 describe el esquema general de un algoritmo evolutivo.

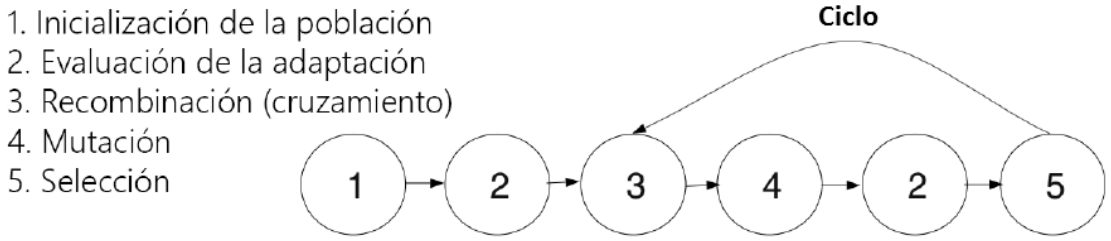

Figura 2: Esquema general de un algoritmo evolutivo
Fuente: (COELLO et al. 2002)

La mención de los AG para resolver problemas de optimización comienza desde 1967 con Rosenberg (GLOVER and KOCHENBERGER 2006) . Luego, el investigador David Schaffer plantea en 1983 un algoritmo llamado VEGA (SCHAFFER 1985) uno de los primeros predecesores de los AG que conocemos hoy en día. En la década de los 90's surgieron muchas metodologías que buscaban soluciones de una manera más eficiente, tales como MOGA (FONSECA and FLEMING 1993) NPGA (HORN *et al.* 1994), NSGA (SRINIVAS and DEB 1994), SPEA (ZITZLER and THIELE 1999), entre otras. Sin embargo en el 2000 y el 2001 NSGAII (DEB *et al.* 2002) y SPEA2 (ZITZLER *et al.* 2001) aumentaron la efectividad de este tipo de algoritmos, a partir de entonces, se han propuesto diferentes metodologías, aunque estas dos últimas han sido las más reconocidas por la comunidad científica (GLOVER and KOCHENBERGER 2006; TORRES, BA CUARTAS 2009).

Los AG en la práctica resultan particularmente adecuados para resolver problemas de optimización, debido a que consideran simultáneamente varias soluciones. Sin embargo, es necesario tener en cuenta dos requisitos previos para utilizarlos en cualquier problema (PADRÓN and MORA 2014):

1. Debe ser posible codificar las soluciones candidatas del problema para poder realizar las operaciones de cruzamiento y mutación; para la evaluación se necesita decodificar la solución.
2. Debe ser posible evaluar las soluciones parciales del problema de manera cuantitativa con el objetivo de guiar el proceso evolutivo.

Los AG son intrínsecamente paralelos, lo que quiere decir que no trabajan de forma secuencial como las técnicas tradicionales, sino que operan de forma simultánea con varias soluciones. Como consecuencia de esto si la solución que se descubre resulta sub-óptima, los AG simplemente desechan esta solución y siguen por otros caminos, mientras que con los algoritmos tradicionales solo se puede renunciar a todo el trabajo hecho y empezar de nuevo (ARRANZ DE LA PEÑA and PARRA TRUYOL 2014).

Los algoritmos resultan menos afectados por los máximos locales (falsas soluciones) que las técnicas tradicionales, siendo esta una clara superioridad a tener en cuenta para su utilización. Además, poseen habilidad para manipular muchos parámetros simultáneamente y no necesitan conocimientos específicos sobre el problema que intentan resolver pues son capaces de realizar cambios aleatorios en sus soluciones candidatas para luego utilizar la función objetivo y determinar si esos cambios producen o no una mejora. Los AG usan operadores probabilísticos, en lugar de los típicos operadores determinísticos de las otras técnicas (ARRANZ DE LA PEÑA and PARRA TRUYOL 2014).

Variadas son las ventajas de la aplicación de AG para dar solución al problema planteado. El uso en la resolución de la problemática está dado por la posibilidad de codificar las soluciones candidatas del problema, lo que permite realizar operaciones de cruzamiento y mutación. Además, a través de una función objetivo es posible determinar si los cambios realizados producen o no una mejora.

Modelos de algoritmos genéticos

Existen dos modelos fundamentales de algoritmos: Modelo Generacional y Modelo Estacionario. En el Modelo Generacional la nueva población reemplaza directamente a la antigua durante cada iteración, pues se crea una población completa con nuevos individuos. Por su parte en el Modelo Estacionario durante cada iteración se le aplican los operadores genéticos a dos padres de la población que se escogen mediante mecanismos de muestreo. El modelo estacionario es elitista y cuando se reemplazan los peores cromosomas de la población produce una presión selectiva alta (convergencia rápida) (GOLDBERG 1989).

Operadores de los algoritmos genéticos

Codificación

Dentro de los AG existe un conjunto de mecanismos de representación – también conocido como codificación- que se emplea para solucionar distintos problemas de optimización. Consiste en codificar en un cromosoma, las variables previamente parametrizadas pues

cada uno tiene genes que corresponden a los respectivos parámetros de un problema (ARRANZ DE LA PEÑA and PARRA TRUYOL 2014). Existen diferentes formas y aunque la más utilizada es mediante una cadena de números binarios, también se puede realizar la codificación mediante números enteros o cadenas de caracteres (ARRANZ DE LA PEÑA and PARRA TRUYOL 2014; CONCEPCIÓN *et al.* 2004).

Una codificación correcta es base clave de una buena resolución del problema, por ello la importancia de la adecuada selección de la forma de representación (CONCEPCIÓN *et al.* 2004). Dicha selección dependerá de del problema a resolver, pues para determinada situación es mejor una codificación basada en números reales, mientras que esa misma codificación complique la solución en otro caso (ARRANZ DE LA PEÑA and PARRA TRUYOL 2014; CASE 2012).

Codificación binaria

La codificación binaria es el método más extendido entre todos pues los primeros AG utilizaban la misma. Concibe cada cromosoma como una cadena de bits (valores de 1 o 0). A su favor tiene que puede abarcar muchos cromosomas incluso con un número reducido de genes. Sin embargo, por otro lado esta opción no es la idónea para muchos problemas y en algunas ocasiones es necesario realizar correcciones después de la reproducción y/o mutación (ARRANZ DE LA PEÑA and PARRA TRUYOL 2014).

Codificación numérica

La codificación numérica utiliza cadenas de números que representan un número en una secuencia. Resulta muy útil en problemas de ordenamiento, para optimizar la elección de una secuencia. Además es muy aplicable en problemas donde se hace necesario elegir ruta, minimizando la distancia a recorrer y evitando repetir puntos en la misma (CASE 2012).

Codificación en árbol

Este procedimiento codifica cada cromosoma como un árbol con ciertos objetos. Los cambios aleatorios pueden generarse cambiando el operador, alterando el valor de un cierto nodo del árbol o simplemente sustituyendo un subárbol por otro. Se utiliza principalmente en el desarrollo de programas o expresiones para programación genética (ARRANZ DE LA PEÑA and PARRA TRUYOL 2014).

Codificación por valor directo

La codificación por valor directo define cada cromosoma como una cadena de valores relacionados con el problema a estudiar, pudiendo ser desde números decimales, cadenas

de caracteres o incluso una combinación de varios de ellos (ARRANZ DE LA PEÑA and PARRA TRUYOL 2014).

Operadores de selección

De la Peña & Parra Truyol afirman en (ARRANZ DE LA PEÑA and PARRA TRUYOL 2014) que: "…es necesario hacer una selección con los individuos más capacitados para que éstos sean los que se reproduzcan con más probabilidad de acuerdo con la teoría de Darwin en la cual los más capacitados son los que deben sobrevivir y crear una nueva descendencia más facultada".

La selección comprende un subproceso de evaluación, en el cuál se decodifica el gen, convirtiéndose en una serie de parámetros de un problema. Se halla la solución del problema a partir de esos parámetros, y se le da una puntuación a la solución en función de su proximidad a la mejor solución. Dicha puntuación conocida por fitness determina siempre los cromosomas que se van a reproducir, y aquellos que se van a eliminar. Una vez evaluado cada cromosoma y obtenida su puntuación, se tiene que crear la nueva población teniendo en cuenta que los buenos rasgos de los mejores se transmitan a esta. Es necesario tener en consideración que hay varias formas de seleccionar la población de la siguiente generación (ARRANZ DE LA PEÑA and PARRA TRUYOL 2014; CONCEPCIÓN *et al.* 2004).

Selección por rueda de ruleta

El operador de selección por Rueda de Ruleta propone crear un patrimonio genético con los cromosomas presentes en una generación. Cada cromosoma tendrá una parte en esa ruleta mayor o menor en función de a la puntuación que tenga cada uno en su fitness. Dentro de la ruleta se escogen pares aleatorios y se emparejan par posteriormente cruzarlos y crear nuevas generaciones (ARRANZ DE LA PEÑA and PARRA TRUYOL 2014; CONCEPCIÓN *et al.* 2004). Es importante señalar que obviamente el cromosoma con mayor puntuación saldrá con mayor probabilidad (ARRANZ DE LA PEÑA and PARRA TRUYOL 2014; OROZCO 2007).

Selección por torneo

La selección por torneo propone escoger aleatoriamente un número de individuos de la población, y el que tiene puntuación mayor se reproduce, sustituyendo su descendencia al que tiene menor puntuación (VALENCIA 1997).

De la Barrera (DE LA BARRERA 2013) precisa que cuando el problema involucra pesos en los parámetros de la función, la selección por torneo es una alternativa eficiente porque permite regular de manera dinámica la presión que se ejerce en la selección.

Selección jerárquica

El operador de selección jerárquica define que los individuos atraviesen múltiples rondas de selección en cada generación. Las evaluaciones de los primeros niveles son más rápidas y menos discriminatorias, mientras que los que sobreviven hasta niveles más altos son evaluados más rigurosamente. La ventaja de este método es que reduce el tiempo total de cálculo al utilizar una evaluación más rápida y menos selectiva para eliminar a la mayoría de los individuos que se muestran poco o nada prometedores, y sometiendo a una evaluación de aptitud más rigurosa y computacionalmente más costosa sólo a los que sobreviven a esta prueba inicial (DE LA BARRERA 2013).

Selección por rango

El operador de selección por rango le asigna a cada cromosoma un rango numérico basado en su aptitud y la selección se realiza en base a este ranking luego de ello los *M* menos dignos son eliminados y sustituidos por la descendencia de alguno de los *M* mejores con algún otro individuo de la población. Este esquema mantiene un porcentaje de la población, generalmente la mayoría, para la siguiente generación (CONCEPCIÓN *et al.* 2004).

El operador produce una variedad genética más rica que el de selección por rueda de ruleta. El problema de su selección es que la convergencia puede ser más lenta ya que no existe notable diferencia entre el mejor cromosoma y el resto como ocurre en su homólogo (OROZCO 2007).

Operadores de cruzamiento

El principal operador genético es el cruzamiento o crossover (del inglés, se usan indistintamente), a tal grado que si no existe cruzamiento se puede decir que no es un AG, aunque puede serlo perfectamente sin mutación, según descubrió Holland en el Teorema de los esquemas (HOLLAND 1975), para hallar la mejor solución a un problema, combinando soluciones parciales (OROZCO 2007). Este es fundamentalmente un intercambio de genes entre dos cromosomas persiguiendo conseguir que los nuevos individuos generados mejoren la aptitud de sus padres (ARRANZ DE LA PEÑA and PARRA TRUYOL 2014; OROZCO 2007).

Cruzamiento por un punto

El operador de cruzamiento por un punto es una de las formas clásicas de cruzamiento para la misma se genera un número aleatorio que determinará el punto por el cual se dividirán ambos cromosomas padres. Luego se procede a copiar la información genética de uno de los padres desde el inicio hasta el punto de cruce y el resto se copia del otro progenitor (OROZCO 2007).

Cruzamiento por N puntos

Similar al operador anterior, con la diferencia de que en este caso los dos cromosomas se dividen por *n* puntos, y el material genético situado entre ellos se intercambia. Lo más habitual es un cruzamiento de un punto o de dos puntos (CONCEPCIÓN *et al.* 2004).

Cruzamiento aritmético

Los progenitores se recombinan según algún operador aritmético para generar su descendiente. Aleatoriamente se selecciona una constante (0,1) y se obtienen dos hijos sustituyendo la posición por el resultado de la suma entre la multiplicación de la constante por el gen correspondiente a la posición y la multiplicación de la inversa de la constante por el gen correspondiente a la misma posición en el otro padre. Puede que la constante no cambie durante el todo el proceso, en tal caso, se trata de un cruzamiento aritmético uniforme, en caso contrario, es un cruzamiento aritmético no uniforme (PINO, VÍCTOR H. and MONDOL 2010).

Cruzamiento uniforme

Existe dos variantes de este operador, una simple y otra con máscara. Cada gen del descendiente se obtiene de cualquiera de los padres de forma aleatoria. Una opción es generar un número aleatorio y si este número supera cierto umbral, se elige un padre determinado, de lo contrario se elige al otro (ARRANZ DE LA PEÑA and PARRA TRUYOL 2014; CONCEPCIÓN *et al.* 2004).

Cruzamiento uniforme con máscara binaria

Similar al operador anterior, cada gen del descendiente se obtiene de cualquiera de los padres de forma aleatoria. Para este método se genera un patrón aleatorio de valores de 1 y 0, y se intercambian los bits de los dos cromosomas que coincidan donde hay un 1 en el patrón. Otra variante es que en caso de que el bit correspondiente a la máscara posea valor 1, se copia el gen de un progenitor y en caso de que posea valor 0, se copia el gen del otro progenitor (ARRANZ DE LA PEÑA and PARRA TRUYOL 2014; CONCEPCIÓN *et al.* 2004).

Operadores de mutación

La mutación consiste en alterar de forma aleatoria ciertos genes, atendiendo a la probabilidad de mutación establecida anteriormente. La mutación obedece a la codificación y a la reproducción, las cuales suelen en gran parte ser ventajosas pues favorecen a la diversidad genética de la especie y de este modo advierten las salidas de la población para evitar verse condicionadas por un óptimo local (OROZCO 2007).

Producto al abuso la mutación puede conllevar a la utilización del algoritmo genético como una simple búsqueda aleatoria (VALENCIA 1997), por lo que es aconsejable estudiar otros procedimientos que tributen pluralidad a la población antes de incrementar las mutaciones. Algunas soluciones podrían ser el incremento del cuerpo de la población o asegurar la aleatoriedad de la población inicial (OROZCO 2007).

Mutación binaria

Entre los operadores de mutación este es el más elemental, pues se puede emplear en codificaciones de tipo binaria y se basa en cambiar la valía de un gen por su opuesto (VITERI *et al.* 2009).

Para este operador, se delimita un principio conocido como probabilidad de mutación que comúnmente es un valor pequeño. Se genera un número aleatorio y de ser menor que el umbral señalado, se asume que el individuo mutará (HOUCK 1995; MEBANE JR 2011; VITERI *et al.* 2009). Se emplea un método similar al método de la ruleta para asegurar que entre los genes exista la equidad de probabilidades para ser seleccionado al escoger el gen específico a mutar, creando un nuevo número aleatorio y se le cambia al gen escogido el valor por el de su opuesto (VITERI *et al.* 2009).

Mutación al borde

Selecciona un gen del cromosoma aleatoriamente y lo iguala a cualquiera de los genes que se hallan en el borde del cromosoma (HOUCK 1995; MEBANE JR 2011; VITERI *et al.* 2009).

Mutación uniforme

Se asemeja mucho a la mutación al borde, salvo que el cambio del valor del gen a mutar se realiza por un valor intermedio entre el actual y el extremo que sea seleccionado (máximo o mínimo) (OROZCO 2007).

Estrategias de reemplazo.

La forma en que los cromosomas de la población son reemplazados por los nuevos descendientes afecta también la presión selectiva. Pueden ser utilizados métodos de reemplazamiento aleatorios, o determinísticos. Cuando se considera un modelo estacionario, nos encontramos con diferentes propuestas (GOLDBERG 1989):

1. **Reemplazar al peor de la población:** genera alta presión selectiva.
2. **Torneo Restringido:** se reemplaza al más parecido. Mantiene una cierta diversidad.
3. **Peor entre semejantes:** se reemplaza el peor cromosoma del conjunto de los padres más parecidos al descendiente generado (seleccionados de toda la población). Busca equilibrio entre diversidad y presión selectiva.
4. **Algoritmo de Crowding Determinístico:** el hijo reemplaza a su padre más parecido. Mantiene diversidad.

Posición asumida ante las alternativas encontradas

Las ventajas que brindan los AG sobre otras técnicas tradicionales de optimización los hacen sumamente atractivos para resolver el problema en cuestión. El AG seguirá un modelo generacional, sin embargo, los operadores clásicos para problemas tipo solo cubren parcialmente las necesidades de la presente investigación. Es por ello que haciendo uso de las bondades que brindan estos algoritmos en la creación de nuevos operadores, se aplican algunas variaciones a los ya existentes, a fin de obtener nuevos que respondan a las necesidades del problema planteado. Es el caso de los operadores de selección, cruzamiento y mutación.

Para la resolución del problema en cuestión se requiere considerar en el cifrado, que el valor de las medidas base que describen los genes del cromosoma, se representa mediante números reales comprendidos entre 0 y 1. Es por ello que se decidió utilizar la codificación por valor directo, dado que esta define cada gen como una cadena de valores y acepta el uso de números reales.

La selección se realizó ordenando jerárquicamente los cromosomas de la población y luego seleccionando el rango de los *N* individuos más próximos al estado inicial. El valor de N que corresponde al tamaño de la muestra poblacional, se calcula utilizando el método probabilístico de cálculo de tamaño de muestra poblacional conociendo el tamaño de la población.

Al realizar el cruzamiento es importante tener en cuenta que es de único interés cruzar los genes del cromosoma sobre los cuales incida la correlación existente entre las medidas bases que describen los escenarios y las buenas prácticas que puede aplicar la organización. Debido a lo anteriormente planteado se determina que el cruzamiento uniforme con máscara binaria es el más apropiado. Sin embargo, por necesidades de la investigación el operador es modificado en su ejecución pues la generación de la máscara binaria para el cruzamiento no se efectúa de manera aleatoria como originalmente describe el operador, sino que se realiza de forma intencionada por la correlación entre las medidas bases y las buenas prácticas.

Para la mutación se definió un nuevo operador que utiliza la misma máscara binaria que la generada para el cruzamiento. Dicha máscara posibilita que en la mutación solo sean alterados los genes que intervinieron en el cruzamiento. Según Valencia en (VALENCIA 1997), para evitar la utilización del AG como una sencilla búsqueda aleatoria la alteración a los cromosomas debe ser mínima, no alterando en más del 1% de su valor. Finalmente, la estrategia de remplazo se define como el incremento de la muestra poblacional resultante de la selección, con los individuos cruzados y mutados.

Para la implementación de la solución se realizó un análisis de las herramientas que conforman desde el punto de vista tecnológico la plataforma de desarrollo. Dado que la implementación formará parte de un sistema de pronóstico al que se nombrará KAIRÓS se siguen los patrones y herramientas de dicho sistema.

Plataforma de desarrollo

Patrón arquitectónico

Para la implementación de la solución se propone utilizar el patrón Modelo Vista Plantilla (MVP) el cual está formado por tres niveles (DJANGO 2016):

- El modelo define los datos almacenados, se encuentra en forma de clases de Python, cada tipo de dato que debe ser almacenado se encuentra en una variable con ciertos parámetros, posee métodos también. Todo esto permite indicar y controlar el comportamiento de los datos.
- La vista se presenta en forma de funciones en Python, su propósito es determinar qué datos serán visualizados. También se encarga de tareas conocidas como la autenticación con servicios externos y la validación de datos a través de formularios. Lo más importante a entender con respecto a la vista es que no se refiere al estilo

de presentación de los datos, sólo se encarga de los datos, la presentación es tarea de la plantilla.

- La plantilla es básicamente una página HTML con algunas etiquetas extras propias de Django, en sí no solamente crea contenido en HTML (también XML, CSS, JavaScript).

Lenguaje de modelado

El Lenguaje Unificado de Modelado (UML por sus siglas en inglés) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir (BOOCH 2005; FOWLER and SCOTT 1999; STEVENS and POOLEY 2002).

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Este lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML es una notación con la cual se construyen sistemas por medio de conceptos orientados a objetos. Este prescribe un conjunto de notaciones y diagramas estándares, y describe la semántica esencial de lo que estos diagramas y símbolos significan (BOOCH 2005).

Programación del lado del cliente

Mozilla Firefox 44

Es un navegador web libre y de código abierto. Entre sus características incluyen la tradicional navegación por pestañas, corrector ortográfico (que puede ser incluido vía Mozilla Addons - Complementos de Mozilla), búsqueda progresiva, marcadores dinámicos, un administrador de descargas, lector RSS, navegación privada, navegación con georreferenciación, aceleración mediante GPU, e integración del motor de búsqueda que desee el usuario. Además, se puede instalar tanto sin conexión como también en línea desde la página web, este último es utilizado para descargar los componentes de segundo plano, ideal para equipos con conexiones mínimas. Es compatible con varios lenguajes web, incluyendo HTML, XML, XHTML, SVG 1.1 (parcial), CSS 1, 2 y 3, ECMAScript (JavaScript), DOM, MathML, DTD, XSLT, XPath, e imágenes PNG con transparencia alfa. También incorpora las normas propuestas por el WHATWG, y es compatible con el elemento HTML Canvas. Implementa el sistema SSL/TLS para proteger la comunicación con los servidores web, utilizando fuerte criptografía cuando se utiliza el protocolo https.

También soporta tarjetas inteligentes para fines de autenticación (NETWORK and CONTRIBUTORS 2014).

JavaScript

Es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, aunque existe una forma de JavaScript del lado del servidor. Su uso en aplicaciones externas a la web, por ejemplo, en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo (GAUCHAT 2012).

Los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Modelo de Objetos del Documento que es una interfaz de programación de aplicaciones (API por sus siglas en inglés). Gran parte de la programación en este lenguaje está centrada en describir objetos, escribir funciones que respondan a movimientos del ratón, aperturas, utilización de teclas, cargas de páginas entre otros (GAUCHAT 2012).

Programación del lado del servidor

Apache 2.0

El servidor HTTP Apache es un servidor web HTTP de código abierto, para plataformas Unix, Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.12 y la noción de sitio virtual. Es el servidor web por excelencia. Su usabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa (FOUNDATION 2013).

Es un software que está estructurado en módulos, así los programadores asumen que el software puede ser ampliado por otros desarrolladores, los cuales pueden escribir pequeñas partes del código que se integrará en Apache de una manera fácil. Esto se lleva a cabo al haber creado un API modular y una serie de fases bien definidas por las que cada petición al servidor debe atravesar. Estas fases van desde la inicialización del servidor hasta la traducción de una URL en un nombre de fichero del servidor, o registrar los resultados de la transacción (KABIR 1998).

Python 2.7

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Python usa tipado dinámico y conteo de referencias para la administración de memoria (LUTZ 2013).

Una característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos). Otro objetivo del diseño del lenguaje es la facilidad de extensión. Python puede incluirse en aplicaciones que necesitan una interfaz programable (LUTZ 2013).

Django 1.8.5

Django es un framework de desarrollo web de código abierto, escrito en Python, que respeta el patrón de diseño conocido como Modelo-Vista-Plantilla. Django proporciona una aplicación incorporada para administrar los contenidos, que puede incluirse como parte de cualquier página hecha con Django y que puede administrar varias páginas hechas con Django a partir de una misma instalación; la aplicación administrativa permite la creación, actualización y eliminación de objetos de contenido, llevando un registro de todas las acciones realizadas sobre cada uno, y proporciona una interfaz para administrar los usuarios y los grupos de usuarios. La meta fundamental de Django es facilitar la creación de sitios web complejos. Django pone énfasis en el re-uso, la conectividad y extensibilidad de componentes, el desarrollo rápido y el principio “No te repitas”. Python es usado en todas las partes del framework, incluso en configuraciones, archivos, y en los modelos de datos (HOLOVATY and KAPLAN-MOSS 2009).

Herramientas de desarrollo

Visual Paradigm 8.0

Durante la etapa de modelación y para diseñar el sistema se utilizó Visual Paradigm, una herramienta de modelado multiplataforma que no se inclina por ninguna metodología específica. Además, ofrece un entorno de creación de diagramas para UML, con soporte para los diagramas de la última versión (UML 2.0) (PARADIGM 2013).

Usa un lenguaje estándar común para todo el equipo de desarrollo que facilita la comunicación y está capacitado para la ingeniería directa e inversa, además de la capacidad de generación de código (PARADIGM 2013).

Presenta dos tipos de diagramas de modelado de bases de datos: entidad-relación (ERD) y mapeo objeto-relacional (ORM). Los diagramas ERD modelan la base de datos a nivel físico y los ORM muestran la relación entre las clases y las entidades (PARADIGM 2013).

PyCharm 4.0.4

PyCharm 4.04 es un entorno de desarrollo integrado (IDE por sus siglas en inglés) que se utiliza para la programación en Python. Proporciona análisis de código, un depurador gráfico, un medidor de unidad integrada, integración con sistemas de control de versiones (VCSes), y apoya el desarrollo web con Django. Comparte la mayoría de las características con otros entornos de desarrollo de la familia IDEA, que significa que tiene HTML robusta, JavaScript y soporte de CSS. Es multiplataforma funciona en Windows, Mac OS X y Linux. PyCharm Professional Edition es libre para proyectos de código abierto y para algunos usos educativos (ISLAM 2015).

Base de datos

PostgreSQL 9.4

PostgreSQL 9.4 es un sistema de bases de datos: relacional potente y de código abierto. Tiene más de 15 años de desarrollo activo y una arquitectura probada que ha ganado una sólida reputación de fiabilidad, integridad de datos y exactitud. Se ejecuta en los sistemas operativos más importantes, incluyendo Linux, Unix, y Windows. Además, incluye la mayor parte de los tipos de datos de SQL 2008, incluyendo INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL y TIMESTAMP. También soporta el almacenamiento de objetos binario grandes, incluyendo imágenes, sonidos y videos. Tiene interfaces nativas de programación para C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, entre otros y una excepcional documentación (PGADMIN 2016).

PostgreSQL 9.4, como base de datos de clase empresarial, cuenta con características avanzadas como Control de Concurrencia Multiversión, espacios de tablas, replicación asincrónica, transacciones anidadas, salvas en línea, un planificador de consultas sofisticadas y se provee de tolerancia a fallos. Es altamente escalable tanto en la enorme cantidad de datos que puede manejar y en el número de usuarios simultáneos que puede acomodar (PGADMIN 2016).

Conclusiones parciales del capítulo

A partir del análisis desarrollado en el presente capítulo se concluye que:

Las principales investigaciones respecto a las relaciones entre las buenas prácticas y los FCE reflejan que: no se tiene en cuenta la influencia de las buenas prácticas sobre FCE y existe una baja reutilización de experiencias asociadas a la influencia de buenas prácticas sobre FCE, que no guía la toma de decisiones en base a escenarios de mejora para obtener mejores resultados frente a los programas de MPS.

Del análisis de los referentes teóricos se identifica como elemento aplicable el uso de los AG con un modelo estacionario y operadores de selección, cruzamiento y mutación, además de una estrategia de reemplazo. Dada la particularidad de las características del problema, se hizo necesario definir nuevos operadores de selección, cruzamiento y mutación que resultan aportes de la investigación.

El uso del framework Django bajo la arquitectura Modelo-Vista-Plantilla y el lenguaje de programación Python para la implementación provee al algoritmo de una programación simplificada debido a las facilidades brindadas por el lenguaje y el framework para la codificación.

Capítulo II - Algoritmo de optimización de estados en la MPS

En el presente capítulo se realiza la descripción de la propuesta de solución. Se describen los artefactos resultantes del diseño e implementación del algoritmo diseñado y se detalla la ejecución del AG a partir del funcionamiento de los operadores.

Operadores del algoritmo de optimización de estados

Codificación

Los individuos o cromosomas que intervienen en los AG se componen de genes. Para el problema en cuestión los cromosomas están representados por los escenarios organizacionales y los genes por las medidas base que describen dichos escenarios. Es por ello que puede entenderse un escenario como un conjunto de medidas base.

Los valores de las medidas base son números reales tal que dicho valor sea mayor o igual que cero y menor o igual que uno. Para un mejor entendimiento de la codificación véase el ejemplo de la Figura 3.

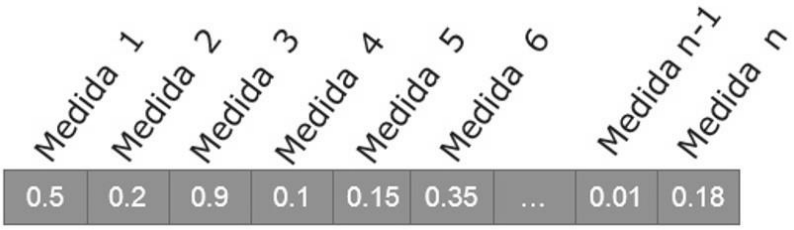

Figura 3: Cromosoma o escenario
Fuente: Elaboración propia

Selección

Para la selección de los escenarios se calcula primero el tamaño de la muestra poblacional mediante el método de cálculo del tamaño de la muestra conociendo el tamaño de la población. Se proceden a evaluar todos los escenarios de la población (incluido el estado inicial que se quiere optimizar) mediante la función objetivo y se ordenan descendentemente los escenarios por su fitness. Se selecciona la muestra poblacional con los *N/2* individuos que estén por encima y los *N/2* que estén por debajo del estado inicial, siendo *N* la cantidad de individuos de la muestra poblacional.

Evaluación

Para la evaluación de los escenarios se lleva a cabo la suma de los productos de la función de aptitud por el grado de mejora de cada una de las medidas base que conforman el

cromosoma en cuestión. El resultado de dicha suma se divide entre la cantidad de medidas base.

La función de aptitud para cada medida se calcula en dependencia de si la misma pertenece o no al conjunto de las medidas que son afectadas por la correlación entre las buenas prácticas que puede aplicar la organización (entrada al AG) para mejorar su estado y las medidas base que describen a dicha organización.

En caso de que pertenezcan a dicho conjunto, la función de aptitud tomará valor 0 si la medida del estado inicial es mayor que la medida del escenario que se está analizando. Si la medida del estado inicial es igual que la medida del escenario que se está analizando entonces la función tomará valor 0,5; mientras que en otro caso tomará valor 1.

Para aquellas medidas que no pertenezcan al conjunto de las que son afectadas por la correlación, se le asignará valor 1 a la función de aptitud de la medida cuando la variación entre la medida del estado inicial y la medida del escenario que se está analizando sea igual a 0. Para cualquier caso diferente la función de aptitud tomará valor 0.

El grado de mejora se calcula restando uno a la normalización de la diferencia entre el valor óptimo alcanzable por la medida en cuestión (resultante del análisis de correlación) y el valor real que tenga dicha media.

Función objetivo

$$f_{obj} = \sum_{i=1}^{n} f\big(Ei(i), Em(i)\big)\, Gm\big(Me(i), Em(i)\big) \Big/ n \quad (1)$$

Donde:

f_{obj} es la función objetivo o de evaluación.

$f\big(Ei(i), Em(i)\big)$ es la función de aptitud para la medida (gen) *i* y se determina mediante la ecuación 2 si $i \in M$ o la ecuación 3 si $i \notin M$.

M es el conjunto de medidas asociadas a las buenas prácticas seleccionadas (se obtiene del análisis de correlación entre las buenas prácticas y las medidas que componen a los FCE).

$Ei(i)$ es el valor de la medida (gen) en la posición *i* del estado (cromosoma) inicial.

$Ei(i) \in \mathbb{R} \,/\, 0 \leq Ei(i) \leq 1$

$Em(i)$ es el valor de la medida (gen) en la posición *i* del escenario (cromosoma) a analizar.

$Em(i) \in \mathbb{R} \,/\, 0 \leq Em(i) \leq 1$

$Gm\big(Me(i), Em(i)\big)$ es el grado de mejora para la medida (gen) *i* y se determina mediante la ecuación 5.

$Me(i)$ es el valor de mejora deseado para $Em(i)$ (se obtiene del análisis de correlación entre las buenas prácticas y las medidas que componen a los FCE)

$Me(i) \in \mathbb{R} \,/\, 0 \leq Me(i) \leq 1$

n es la cantidad de medidas base (genes) del escenario (cromosoma).

Para $\boldsymbol{i} \in \boldsymbol{M}$

$$f\big(Ei(i), Em(i)\big) = \begin{cases} 1 & si \quad Ei(i) < Em(i) \\ 0.5 & si \quad Ei(i) = Em(i) \\ 0 & si \quad Ei(i) > Em(i) \end{cases} \qquad (2)$$

Para $\boldsymbol{i} \notin \boldsymbol{M}$

$$f\big(Ei(i), Em(i)\big) = \begin{cases} 1 & si \quad \delta\big(Ei(i), Em(i)\big) = 0 \\ 0 & si \quad \delta\big(Ei(i), Em(i)\big) = 1 \end{cases} \qquad (3)$$

Donde:

$\delta\big(Ei(i), Em(i)\big)$ es la distancia existente entre el valor de $Ei(i)$ y $Em(i)$. Se calcula mediante la ecuación 4.

$$\delta\big(Ei(i), Em(i)\big) \begin{cases} 1 & si \quad Ei(i) - Em(i) \neq 0 \\ 0 & si \quad Ei(i) - Em(i) = 0 \end{cases} \qquad (4)$$

$\boldsymbol{Gm\big(Me(i), Em(i)\big)$ se determina de la siguiente manera:

$$Gm\big(Me(i), Em(i)\big) = 1 - |Me(i) - Em(i)| \qquad (5)$$

Cálculo del tamaño de la muestra poblacional

Una vez evaluados los escenarios de la población, se procede a realizar un ordenamiento jerárquico de manera descendente atendiendo a los resultados de la función objetivo para cada escenario y se selecciona una muestra de población que corresponde al rango (ranking) de los n_{mp} primeros escenarios (cromosomas) de la lista ordenada, donde n_{mp} constituye el resultado de la aplicación de la técnica de *muestreo probabilístico* que se determina por la ecuación 6 (TORRES, MARIELA *et al.* 2006).

$$n_{mp} = \frac{N \times Z_{\alpha}^{2} \times p \times q}{d^{2} \times (N-1) + Z_{\alpha}^{2} \times p \times q} \quad (6)$$

Donde:

n_{mp} es el tamaño de la muestra probabilística

N es el tamaño de la población total

Z es el nivel de confianza, dado por la probabilidad de que la estimación efectuada se ajuste a la realidad; es decir, que esté comprendida dentro de un intervalo determinado basado en el estimador y que dicha estimación capte el valor verdadero del parámetro a medir. El valor de Z se calcula mediante la campana de Gauss (MODE 1990).

p es la proporción esperada, en caso de desconocerse se toma el valor 0,5.

q es la probabilidad de fracaso dada por la ecuación 7:

$$q = 1 - p \quad (7)$$

d es la precisión que se desea o error que se prevé cometer. Su valor suele oscilar entre el 1% (0,01) y 9% (0,09).

En el contexto de la presente investigación se asume nivel de confianza de 1,962; dado que se desconoce la proporción esperada se asumen para esta un valor de 0,5 y probabilidad de fracaso de 0,5; el nivel de precisión será de 0,01.

Cruzamiento

Utilizando la máscara binaria que identifica cuáles son las medidas base que son afectadas por la correlación con las buenas prácticas se procede a crear un nuevo individuo. El mismo contiene los genes del estado inicial que coinciden con valores igual a 0 en la máscara binaria y los genes del escenario *N* de la base de conocimientos que coinciden con valores igual a 1 en dicha máscara.

Atípicamente al cruzar el estado inicial con uno de los escenarios de la base de conocimientos se obtiene un solo hijo o escenario de mejora. Esto se debe a que uno de los hijos contiene valores diferentes a los que tiene el estado inicial en los genes que no se deben alterar por no estar afectados por la correlación. Por lo tanto, debe ser desechado pues no es de interés en el resultado final. Para un mejor entendimiento del cruzamiento ver el ejemplo de la Figura 4.

Figura 4: Cruzamiento por máscara binaria
Fuente: Elaboración propia

Mutación

Al final del proceso evolutivo, los algoritmos que posean probabilidad de mutación deberán ser mutados. Al igual que en el cruzamiento se hace necesario seleccionar el gen a mutar de forma intencional, dado que se debe evitar alterar algún cromosoma que no esté afectado por la correlación existente entre las buenas prácticas que la organización pueda aplicar para mejorar su estado inicial y las medidas base que componen dicho estado. Para ello se utiliza la misma máscara binaria generada en el cruzamiento y se genera un número aleatorio mayor que 0 y menor que la cantidad de medidas base que componen el estado inicial. En caso de que el número aleatorio coincida con la posición de alguna medida afectada por la correlación, entonces el valor de dicha medida se incrementa en 1%, en caso contrario se ignora el escenario en el proceso de mutación. Para un mejor entendimiento de la mutación ver el ejemplo de la Figura 5.

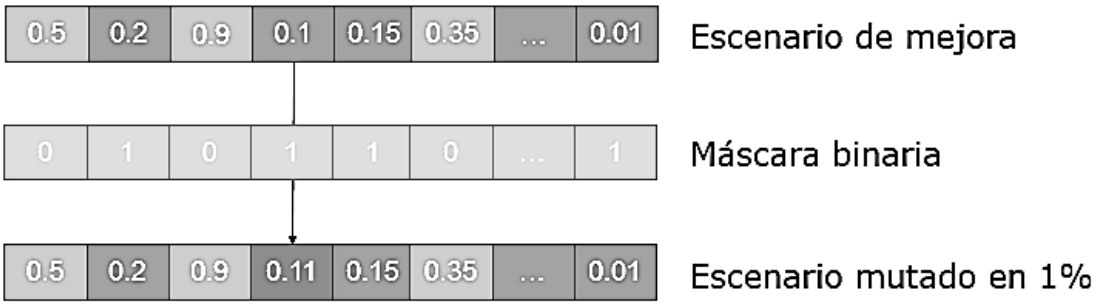

Figura 5: Mutación por máscara binaria
Fuente: Elaboración propia

Reemplazo

El algoritmo sigue un Modelo Generacional aplicando los operadores de Selección, Cruzamiento y Mutación a todos los escenarios (cromosomas) de la muestra poblacional. Para cada nueva iteración del algoritmo es necesario realizar el reemplazo, que se realiza adicionando al conjunto de escenarios seleccionados de la iteración anterior el conjunto de escenarios cruzados y mutados, tomando la población resultante como la población inicial para la próxima selección (ver Figura 6).

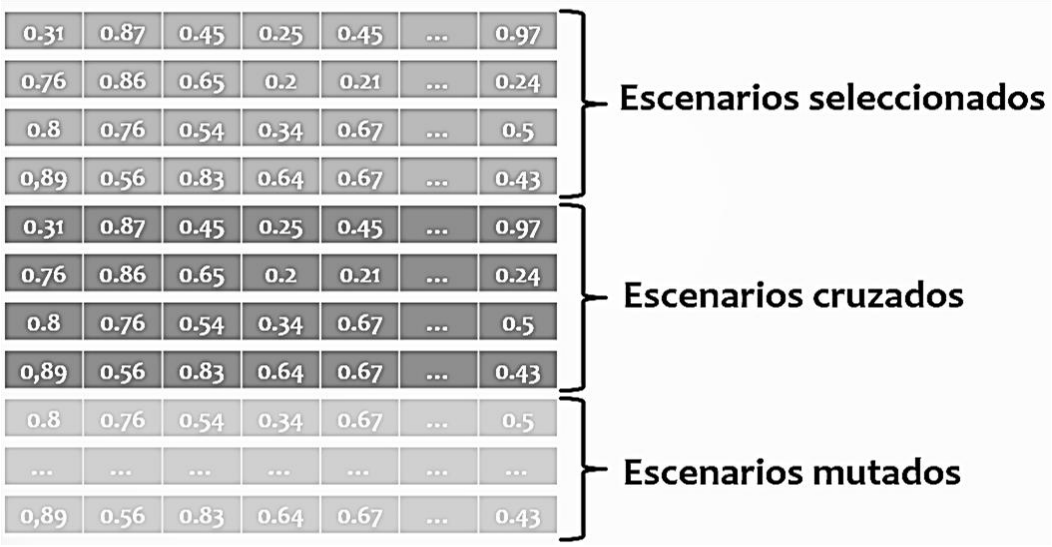

Figura 6: Reemplazo
Fuente: Elaboración propia

Descripción del algoritmo de optimización de estados

1. Generar población inicial.
2. Seleccionar escenarios.
3. Comprobar si entre los escenarios seleccionados se encuentra la solución.
    a. Si se encuentra devuelve el primer y último cromozomas de la población.
    b. Si no se encuntra ejecuta el paso 4.
4. Cruzar escenarios.
5. Mutar escenarios.
6. Incrementar población.
7. Ejecutar el paso 2.

Paso 1: Generar la población inicial.

1. Calcular el tamaño de la muestra poblacional.
2. Seleccionar aleatoriamente de la base de conocimientos la muestra poblacional.

Paso 2: Seleccionar escenarios.

1. Calcular el tamaño de la muestra poblacional.
2. Evaluar todos los escenarios mediante la función objetivo.
3. Ordenar jerárquicamente todos los escenarios atendiendo a su aptitud.
4. Seleccionar la muestra poblacional atendiendo a que la mitad se correspondan a los mayores más cercanos al estadio inicial y la otra mitad a los menores más cercanos.

Paso 3: Comprobar si entre los escenarios seleccionados se encuentra la solución.

1. Se toma el último de los escenarios de la muestra poblacional y se verifica si su fitnes sobrepasa al menos en un 75% al fitnes del estado inicial.

a. En caso de ser positivo el sistema retorna el primer y último escenario de la muestra poblacional.

Paso 4: Cruzar escenarios.

1. Para cada escenario de la muestra poblacional:
2. Verificar que exista probabilidad de cruzamiento entre el estado inicial y el escenario que se analiza:

a. En caso que exista, para cada gen del escenario:
   i. Verificar si la posición del gen que se está analizando corresponde con valor 1 en la máscara binaria.
      1. En caso de ser verdadero agregar al escenario nuevo el gen del escenario que se analiza.
      2. En caso de ser falso agregar al escenario nuevo el gen del estado inicial.
   ii. Adicionar el nuevo escenario al conjunto de escenarios cruzados.

Paso 5: Mutar escenarios.

1. Para cada escenario de los escenarios cruzados:
2. Generar un número aleatorio entero mayor que cero y menor que el tamaño del escenario.
3. Verificar que exista probabilidad de mutación

a. En caso que exista incrementar el valor de la medida ubicada en la posición correspondiente al número aleatorio generado.
b. Adicionar el nuevo escenario al conjunto de escenarios mutados.

Paso 6: Incrementar población.

1. Adicionar a la población inicial el conjunto de escenarios cruzados y escenarios mutados.

Paso 7: Ejecutar el paso 2.

Pseudocódigo del algoritmo genético de optimización de estados en la MPS

```
arreglo baseDeConocimientos [][]
arreglo poblacion [][]
arreglo estadoInicial []
arreglo mascaraBinaria []
arreglo gradoDeMejora []

        arreglo escenariosSeleccionados[][] = SeleccionarEscenarios(poblacion[][])
        SI Encontrado(escenariosSeleccionados[escenariosSeleccionados.longitud-1])
                retornar escenariosSeleccionados[0],
                        escenariosSeleccionados[estadosSeleccionados.longitud - 1]
        SINO
                OptimizacionDeEstados(IncrementarPoblacion(CruzarEscenarios(estadosS
                eleccionados),
                MutarEscenarios(CruzarEscenarios(estadosSeleccionados))))
```

Pseudocódigo para generar la población inicial

```
PoblacionInicial()
        entero tamanoDeMuestra = baseDeConocimientos[][].longitud
        entero i = 0
        while (i < tamanoDeMuestra)
                i = i + 1
                seleccionado = random(0, TamanoDeMuestra)
                poblacion[][].agregar(BaseDeConocimientos[seleccionado])
```

Pseudocódigo del cálculo de tamaño de muestra poblacional

```
CalcularTamanoDeMuestra(entero tamanoDePoblacion)
        real confianza = 1.962
        real presición = 0.01
        real propEsperada = 0.5
        real probFracaso = 1 - p
        return (tamanoDePoblacion * confianza * propEsperada * q) / (errorEsperado *
(tamanoDePoblacion - 1) + confianza * propEsperada * q)
```

Pseudocódigo para la selección de estados

```
SeleccionarEstados(arreglo poblacionInicial[][])
        EvaluarEscenarios(poblacionInicial[][])
        entero tamanoDeMuestra =
                        CalcularTamanoDeMuestra(poblacionInicial[][].longitud-1)
         return SeleccionarEscenariosPorRango(tamanoDeMuestra,
                        OrdenarEscenariosJerarquicamente(poblacionInicial[][]))
```

Pseudocódigo para evaluar los escenarios

```
EvaluarEscenarios(arreglo poblacionInicial[][])
        Evaluar(estadoInicial[][])
        PARA i = 1 MIENTRAS i < PoblacionInicial[][].longitud HACER i = i + 1
                Evaluar(PoblaciónInicial[i][])

Evaluar(arreglo escenario[]):
        entero evaluacion
        PARA entero i = 1, MIENTRAS i < escenario[].longitud HACER: i = i + 1
                SI mascaraBinaria[i] == 1:
                        SI escenario[i] > estadoInicial[i]
                                entero FA = 0
                SINO SI escenario[i] < estadoInicial[i]
                                entero FA = 0.5
                        SINO
                                entero FA = 1
                SINO
                        SI Distancia(escenario[i], estadoInicial[i]) = verdadero
                                entero FA = 0
                        SINO
                                entero FA = 1

                real mo = gradoDeMejora[i]
                real GM = 1 - absoluto(mo - i)
                evaluacion = evaluacion + (FA * GM)
        escenario[escenario.longitud - 1] = evaluacion/(escenario.longitud)
```

Distancia(real gen1, real gen2):

```
        SI (gen1-gen2)!=0:
                retornar verdadero
        retornar falso
```

Pseudocódigo para ordenar escenarios jerárquicamente

```
OrdenarEscenariosJerarquicamente(arreglo escenarios[][], entero primero, entero ultimo)
        entero central = (primero + ultimo)/2
        real pivote = escenarios[central]
        entero i = primero
        entero j = ultimo

        HACER:
                MIENTRAS (escenarios[i] < pivote):
                        i = i + 1
                MIENTRAS (escenarios[j] > pivote):
                        j = j - 1
                SI (i<=j):
                        real temp = arreglo[i]
                        escenarios[i] = escenarios[j]
                        escenarios[j] = temp
                        i = i + 1
```

```
                    j = j - 1
        MIENTRAS (i <= j)

        SI (primero < j)
                OrdenarEscenariosJerarquicamente(escenarios, primero, j)
        SI (i < ultimo)
                OrdenarEscenariosJerarquicamente(escenarios, i, ultimo)
```

Pseudocódigo para seleccionar escenarios por rango

```
SeleccionarEscenariosPorRango(entero tamanoDeMuestra, arreglo poblacionInicial[][])
        arreglo poblacion[]
        entero menor
        PARA entero i = 1 MIENTRAS i < PoblacionInicial[][].longitud HACER: i = i + 1
                SI (poblacionInicial[i][ poblacionInicial[i].longitud -1 ] <=
                                                EstadoInicial[EstadoInicial[].longitud -1])
                        menor = i

        entero puntero = menor - (tamanoDeMuestra/2)

        MIENTRAS (tamanoDeMuestra > 0):
                poblacion.adicionar(poblacionInicial[puntero])
                puntero = puntero + 1
                tamanoDeMuestra = tamanoDeMuestra - 1

        retornar poblacion
```

Encontrado(estado[])

```
        real var1 = estado[estado[].longitud -1]
        real var2 = EstadoInicial[EstadoInicial[].longitud - 1]
        SI var1 >= (var2 + (var2 * 0.75))
                retornar verdadero
        SINO
                retornar falso
```

Pseudocódigo para cruzar escenarios

```
CruzarEscenarios(estadosSeleccionados [][])
        arreglo estadosCruzados [][]
        PARA entero i = 1 MIENTRAS i < = estadosSeleccionados [][].longitud HACER i = i
+ 1
                arreglo estadoNuevo []
                PARA entero j = 1 MIENTRAS i > EstadoInicial [].longitud
                        SI mascaraBinaria[j]==1:
                                estadoNuevo[j]=Escenario[j]
                        SINO
                                estadoNuevo[j]=EstadoInicial[j]
                        estadosCruzados.adicionar(estadosCruzados)
        retornar EstadosCruzados[][]
```

Pseudocódigo para mutar escenarios

```
MutarEscenarios(arreglo estadosCruzados[][])
    arreglo estadosMutados []
    PARA entero i = 1 MIENTRAS i < estadosCruzados[][].longitud HACER i = i + 1
        entero aleatorio=random(0, escenario[].longitud-1)
        SI ProbabilidadDeMutacion(aleatorio) = verdadero
        arreglo EstadoNuevo[] = estadosCruzados[i][]
        estadoNuevo[A] = estadoNuevo[aleatorio] + estadoNuevo[aleatorio]*0.01
        estadosMutados[].adicionar(estadoNuevo[])
    retornar EstadosMutados[]

ProbabilidadDeMutacion(entero posicion):
    SI mascaraBinaria[posicion]==1:
        retornar verdadero
    SINO
        Retornar falso
```

Pseudocódigo para incrementar la población

```
Incrementar_Poblacion(estadosCruzados[][], estadosMutados[][])
    retornar poblacion.adicionar(estadosCruzados[][].adicionar(estadosMutados[][]))
```

Diseño de la solución

Diagramas de clases persistentes

Mediante el diagrama de clases persistentes se definen los conceptos que son empleados en el AG y que describen la estructura de la base de datos. En dicho modelo se representan las entidades, sus atributos y tipos, sus relaciones y las restricciones que deben cumplirse.

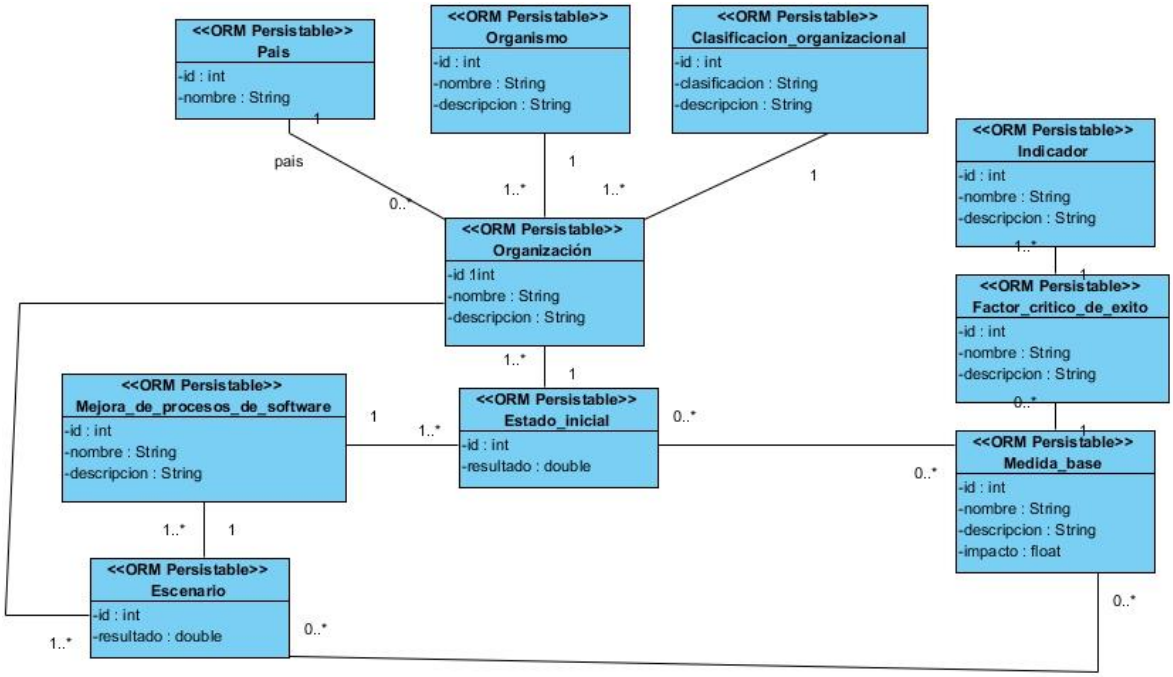

Figura 7: Diagrama de clases persistentes del sistema KAIRÓS
Fuente: Elaboración propia

Descripción de las clases persistentes

Se identificaron 12 clases persistentes. A continuación, se detallan 2 de ellas y el resto, en los anexos.

Nombre: Estado_inicial	
Descripción: Describe los estados iniciales de las organizaciones	
Atributo	**Descripción**
id	Identificador
resultado	Resultado del estado
medidas	Arreglo de Medidas base que describen el estado inicial
Descripción de los métodos	
Método	Descripción
insertarEstado_inicial	Busca el objeto Estado_inicial recibido como parámetro en la base de datos. En caso de no encontrarlo, lo registra.
eliminarEstado_inicial	Busca el objeto Estado_inicial recibido como parámetro en la base de datos. En caso de encontrarlo, lo elimina.

Tabla 2: Descripción de la clase Estado_inicial
Fuente: Elaboración propia

Nombre: Escenario	
Descripción: Describe todos los escenarios de la base de conocimientos	
Atributo	**Descripción**
id	Identificador
resultado	Resultado del escenario
medidas	Arreglo de Medidas base que describen el escenario
Descripción de los métodos	
Método	Descripción
insertarEscenario	Busca el objeto Escenario recibido como parámetro en la base de datos. En caso de no encontrarlo, lo registra.
eliminarEscenario	Busca el objeto Escenario recibido como parámetro en la base de datos. En caso de encontrarlo, lo elimina.

Tabla 3: Descripción de la clase Escenario
Fuente: Elaboración propia

Diseño de la base de datos

En el modelo de datos se definen los conceptos que son utilizados para la solución y que describen la estructura de la base de datos diseñada. Se representan los datos, sus atributos y tipos, sus relaciones y las restricciones que deben cumplirse.

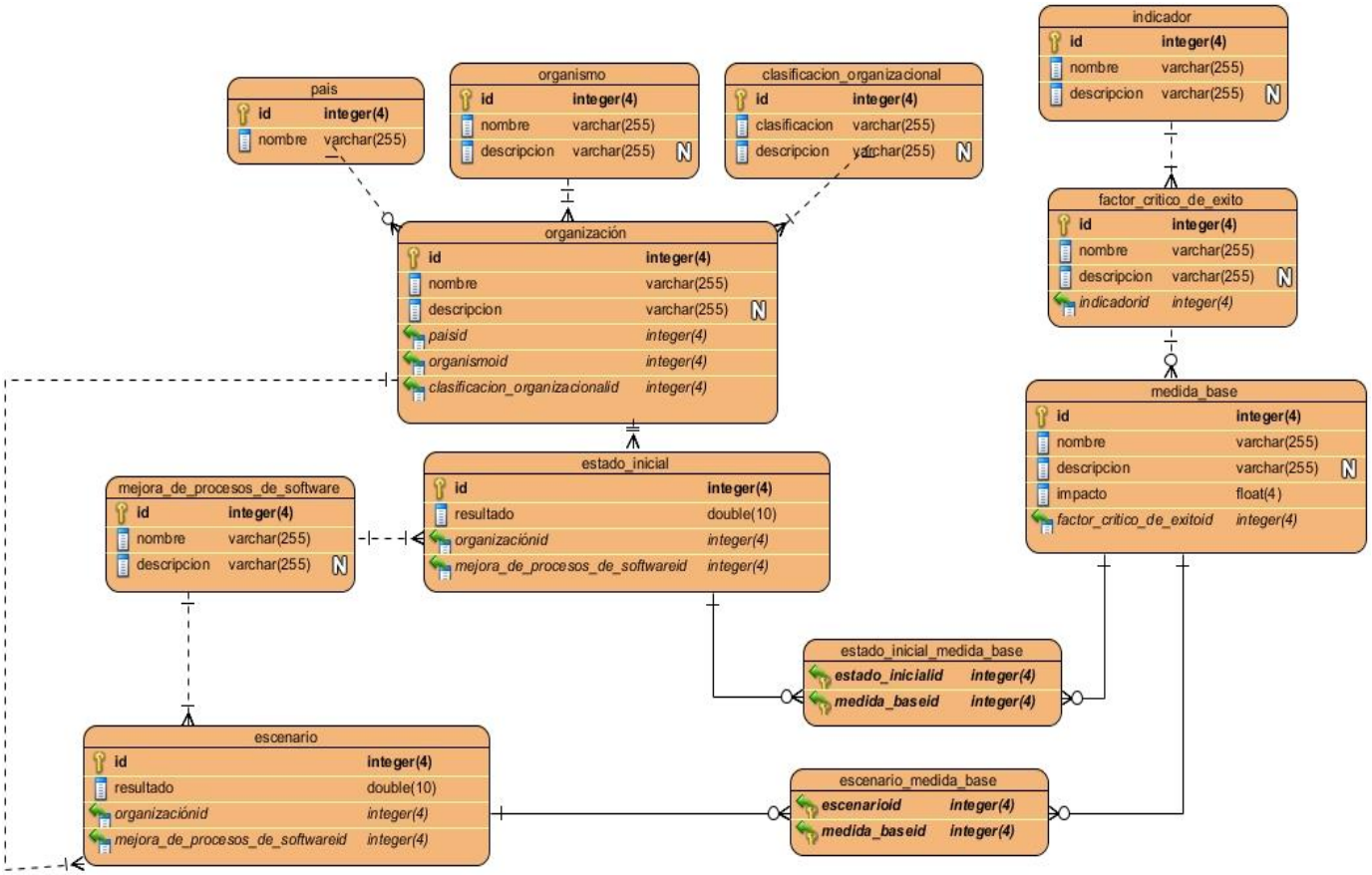

Figura 8: Diagrama Entidad-Relación
Fuente: Elaboración propia

Descripción de las tablas

Se identificaron 12 tablas. A continuación, se detallan 2 de ellas y el resto, en los anexos.

Nombre: estado_inicial	
Descripción: Almacena todos los estados iniciales de las organizaciones	
Atributo	**Descripción**
id	Identificador
resultado	Resultado del estado
mejora_proceso_softwareid	Llave foránea que referencia al identificador de la tabla mejora_proceso_software
organizacion_id	Llave foránea que referencia al identificador de la tabla organizacion.

Tabla 4: Descripción de la tabla estado_inicial
Fuente: Elaboración propia

Nombre: escenarios	
Descripción: Almacena todos los escenarios de la base de conocimientos	
Atributo	**Descripción**
id	Identificador
resultado	Resultado del escenario
mejora_proceso_softwareid	Llave foránea que referencia al identificador de la tabla mejora_proceso_software
organizacion_id	Llave foránea que referencia al identificador de la tabla organizacion.

Tabla 5: Descripción de la tabla escenario
Fuente: Elaboración propia

Implementación

Diagrama de componentes

Los diagramas de componentes permiten describir los elementos físicos que integran la solución y las relaciones que existen entre ellos. Muestran además las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la producción de aplicaciones informáticas. Pueden ser simples archivos, paquetes, y/o bibliotecas cargadas dinámicamente (PRESSMAN 2010). En la Figura 9 se presenta el diagrama de componentes asociado al AG implementado.

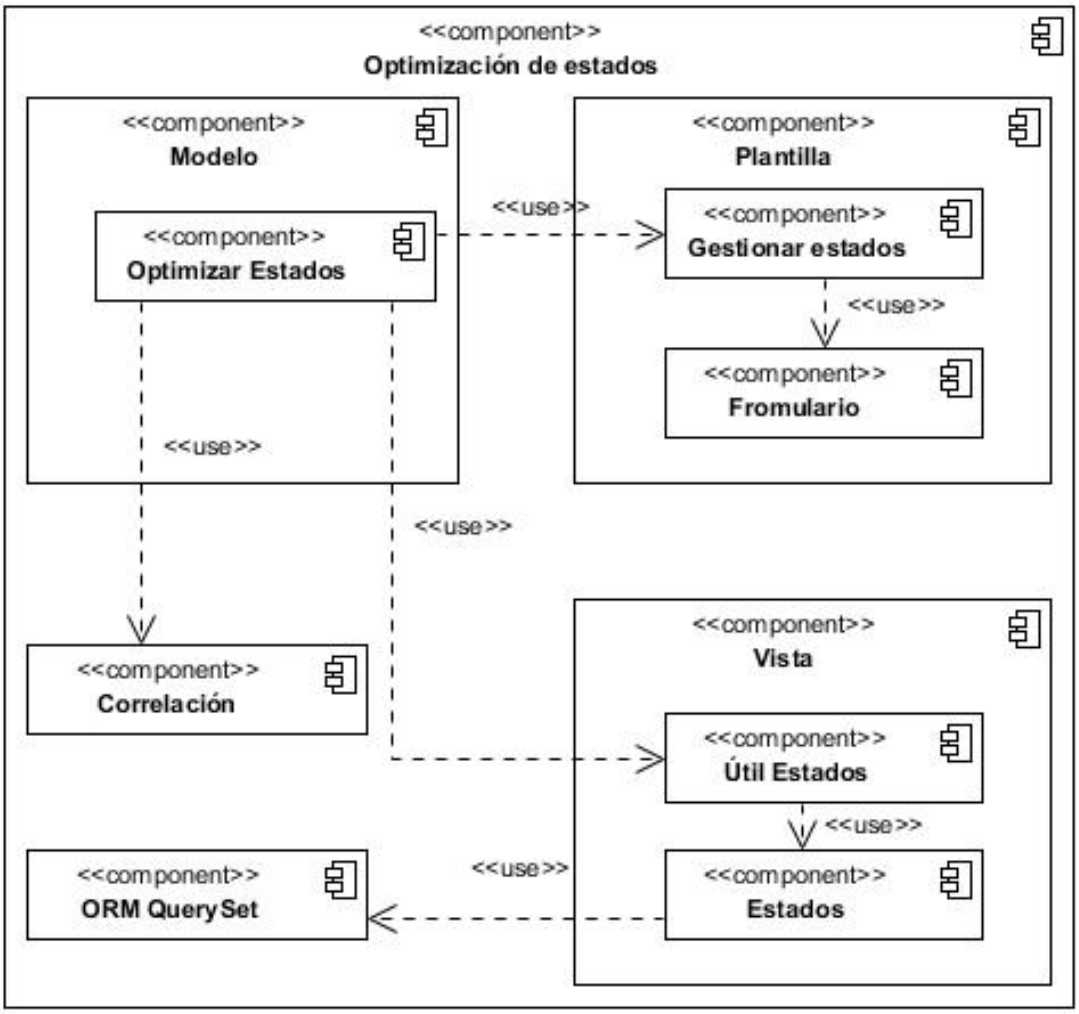

Figura 9: Diagrama de componentes
Fuente: Elaboración propia

Estándares de codificación

En pos de reflejar un estilo armonioso y garantizar la mantenibilidad y limpieza de los datos se define el estándar de codificación que comprende los aspectos de la generación de código. Bajo la premisa de que el estándar debe tender siempre a lo práctico, se utilizó como referencia los estándares definidos para el lenguaje de programación empleado (LUTZ 2013).

Nombres de ficheros

Sufijos de ficheros en Python:

- Código fuente: .py
- Código compilado: .pyc

Organización de ficheros

Cada fichero consta de secciones separadas por líneas en blanco y líneas de comentario, ningún fichero tiene más de 2000 líneas de código.

Indentación

La unidad de indentación de bloques de sentencias es de 4 espacios

Líneas largas

Cuando una línea no cabe en una única línea se fracciona atendiendo a los siguientes principios generales:

- Fraccionar después de una coma.
- Fraccionar después de un operador.

Comentarios

- Los comentarios añaden claridad al código.
- Los comentarios son concisos.

Declaraciones

Se declara cada variable en una línea distinta, posibilitando que cada variable se pueda comentar por separado.

Inicialización

Se inicializa cada variable en su declaración a menos que su valor inicial dependa de algún cálculo.

Sentencias

Sentencias simples

Cada línea contiene una sola sentencia.

Sentencias compuestas

En Python las sentencias compuestas no se encierran entre llaves, sino que las sentencias del tipo *if*, *for*, *while*, etc deben estar sucedidas del símbolo dos puntos. Además, todas las sentecias compuestas estan indentadas a un nivel superior que el precedente.

Líneas en blanco

Se usan dos líneas en blanco entre métodos y una entre:

- Comentarios y métodos.
- Variables locales de un método y la primera sentencia.

Asignación de nombres

Se usan descriptores en español que dejan claro el cometido de la variable, método o clase.

- **Para los métodos:** Se usan verbos. La primera letra de la primera palabra en minúsculas, el resto de las palabras empiezan por mayúsculas.
- **Para las variables:** Comienzan por minúscula.

Tratamiento de errores

El lenguaje de programación Python maneja los errores mediante las sentencias ***try*** y ***except***. Primero, ejecuta el bloque ***try*** y si no ocurre ninguna excepción, el bloque ***except*** se omite y termina la ejecución de la declaración ***try***. Si ocurre una excepción durante la ejecución del bloque ***try***, el resto del bloque se omite. Si su tipo coincide con la excepción nombrada luego de la palabra reservada ***except***, se ejecuta el bloque ***except***, y la ejecución continúa luego de la declaración ***try***. Si ocurre una excepción que no coincide con la excepción nombrada en el ***except***, esta se pasa a declaraciones ***try*** de más afuera; si no se encuentra nada que la maneje, es una excepción no es manejada, y la ejecución se frena con un mensaje (LUTZ 2013).

Conclusiones parciales del capítulo

A partir de lo anteriormente expuesto en el capítulo se concluye:

Se diseñó e implementó un AG para la optimización de estados en la MPS que sigue un modelo estacionario y cuya función objetivo permite la obtención de escenarios de mejora a partir de su estado inicial y el conjunto de buenas prácticas que se puede aplicar para mejorarlo.

El operador de selección empleado, posibilita que los escenarios seleccionados en cada iteración no disten de las posibilidades reales de la organización siendo asequibles para la misma. Además, los operadores de cruzamiento y mutación modificados para la solución, estimulan la generación de nuevos individuos con mejores condiciones de adaptación al objetivo de optimización.

El diseño de las clases persistentes y el modelo de datos, facilitó la modelación, especificación y documentación gráfica del modelo estructural para la implementación del AG; a la vez que sirvió de guía para la codificación de la solución.

Los estándares de codificación empleados para la implementación del AG contribuyeron a obtener una limpieza en el código fuente con vista a su mantenibilidad y comprensión.

Capítulo III - Validación de la solución

El presente capítulo presenta el diseño del proceso de validación de la solución y realiza un análisis de los resultados obtenidos. El proceso de validación de la solución está conformado por: el desarrollo de un pre-experimento con el objetivo de valorar el funcionamiento del AG a partir de la reutilización de experiencias y la aplicación de la técnica de Iadov para medir el grado de satisfacción del cliente respecto a la utilidad y aplicabilidad del algoritmo desarrollado. El capítulo concluye con un análisis del impacto económico de la propuesta de solución.

Proceso de validación de los resultados

Aplicación del pre-experimento y resultados pre-prueba y post-prueba

Para desarrollar el **pre-experimento**, se tuvieron en cuenta como entrada 3 centros de la UCI (FORTES, GEYSED e ISEC). En la etapa pre-prueba se utilizó la herramienta Hefexto a la que se le introdujo los datos resultantes de un diagnóstico realizado en 2012 mediante el modelo Si.MPS.Cu (GARCIA 2013). Como resultado Hefexto le pronosticó a cada centro un 100% de fracaso, lo que concuerda con en el diagnóstico aplicado a 14 centros de desarrollo de la UCI en el año 2012 (GARCIA 2013; TRUJILLO 2014).

Para la etapa post-prueba se seleccionaron además un total de 13 buenas prácticas aplicables a los centros y se le aplicó a cada uno el algoritmo de optimización de escenarios resultando de ello un total de seis escenarios; dos para cada centro (escenario de mejora mínimo y escenario de mejora máximo).

Posteriormente se procedió a relazar el diagnóstico a los 6 escenarios resultantes con le herramienta Hefexto y se obtuvieron los siguientes resultados:

- Centro FORTES:
  - **Escenario de mejora máximo:** 11 casos recuperados como semejantes arrojando un pronóstico 100% éxito.
  - **Escenario de mejora mínimo:** 11 casos recuperados como semejantes arrojando un pronóstico 80% éxito.
- Centro GEYSED:
  - **Escenario de mejora máximo:** 10 casos recuperados como semejantes arrojando un pronóstico 80% éxito.
  - **Escenario de mejora mínimo:** 10 casos recuperados como semejantes arrojando un pronóstico 70% exito.

- Centro ISEC:
  - **Escenario de mejora máximo:** 11 casos recuperados como semejantes arrojando un pronóstico 100% éxito.
  - **Escenario de mejora mínimo:** 11 casos recuperados como semejantes arrojando un pronóstico 100% éxito.

Los resultados de pronóstico arrojados por Hefexto se corresponden con la certificación de estos centros en 2015 en el nivel 2 de CMMI (CMMI 2015). Por lo tanto, existe una concordancia entre los escenarios obtenidos y la situación actual de los centros luego de haber sido diagnosticados de fracaso en 2013 (GARCIA 2013; TRUJILLO 2014) y lograr mejoras que los llevaron el éxito en el programa de mejora en 2015.

A partir del pre-experimento realizado, se corrobora que el algoritmo implementado para la optimización de estados, tiene un efecto positivo en la reutilización de experiencias para guiar las decisiones en función del éxito en la MPS.

Valoración de la satisfacción del cliente con el algoritmo genético desarrollado

Se aplicó la técnica Iadov que posibilita el estudio del grado de satisfacción del personal involucrado en un proceso objeto de análisis para valorar la satisfacción del cliente respecto a la utilidad y aplicabilidad del AG implementado en entornos reales. Fungieron como clientes 10 profesionales que constituyen especialistas y profesores del centro de calidad de software de la UCI.

La técnica Iadov se compone de cinco preguntas claves: tres cerradas y dos abiertas, las cuales se reformulan en la investigación para valorar el grado de satisfacción de los clientes sobre la implementación del algoritmo propuesto, para la optimización de estados en la MPS. Una vez establecidas las preguntas se conforma el “cuadro lógico de Iadov” y el número resultante de la interrelación de las tres preguntas, indica la posición de los sujetos en la escala de satisfacción. La escala de satisfacción está dada por los criterios (KUZMINA 1970):

1. Clara satisfacción.
2. Más satisfecho que insatisfecho.
3. No definida.
4. Más insatisfecho que satisfecho.
5. Clara insatisfacción.
6. Contradictoria.

A continuación, se muestra el cuadro lógico de Iadov generado a partir de la encuesta entregada a los clientes (ver Anexo 4).

¿En qué medida considera que el algoritmo genético para obtener estados de mejora es aplicable a las necesidades de su organización para la toma de decisiones en la MPS?	¿Considera usted que las organizaciones deben guiar sus esfuerzos para la mejora sin tener en cuenta las buenas prácticas que pueden aplicar en función de disminuir la acción de los Factores Críticos de Éxito?								
	No			No sé			Sí		
	¿Considera útil la posibilidad contar con escenarios de mejora para guiar las decisiones de la organización a partir de la reutilización de experiencias teniendo en consideración el impacto de las buenas prácticas sobre los FCE?								
	Sí	No sé	No	Sí	No sé	No	Sí	No sé	No
Aplicabilidad muy adecuada	1	2	6	2	2	6	6	6	6
Aplicabilidad adecuada	2	2	3	2	3	3	6	3	6
Resulta indiferente	3	3	3	3	3	3	3	3	3
Aplicabilidad poco adecuada	6	3	6	3	4	4	3	4	4
No es nada aplicable	6	6	6	6	4	4	6	4	6
No sé qué decir	2	3	6	3	3	3	6	3	4

Tabla 6: Cuadro lógico de Iadov
Fuente: Elaboración propia

Las preguntas abiertas que se formularon fueron:

- ¿Qué elemento(s) considera favorables respecto al algoritmo propuesto?
- ¿Qué recomendación(es) sugiere para mejorar la solución propuesta?

Nivel de satisfacción	Cantidad	%
Máxima satisfacción	9	90,00
Más satisfecho que insatisfecho	1	10,00
No definida	0	0,00

Tabla 7: Resultado de aplicación de la técnica Iadov
Fuente: Elaboración propia

Para calcular el nivel de satisfacción grupal se utiliza la fórmula que propone la técnica (KUZMINA 1970):

$$ISG = \frac{9(+1) + 1(+0{,}5) + 0(0)}{n}$$

Al procesar las respuestas a las encuestas en el cuadro lógico de Iadov, se obtiene un grado de satisfacción grupal de 0,90, lo cual se traduce en una clara satisfacción con el uso del AG implementado para optimizar los estados organizacionales de las empresas interesadas en iniciarse en la MPS. En el criterio de los clientes respecto a la utilidad del algoritmo implementado, hubo una concordancia de un 100,00% con la aceptación de la utilidad del AG. Respecto a la aplicabilidad del algoritmo existió una concordancia de un 90,00% con la calificación de “Máxima satisfacción” y el otro 10,00% lo calificó como “Más satisfecho que insatisfecho”.

Entre las valoraciones positivas obtenidas como respuestas a las preguntas abiertas, se recopilaron criterios como los siguientes:

- Los clientes destacan la reutilización de experiencias de otros centros u organizaciones para la optimización de estados como un aporte importante, pues logra mejorar sus estados a partir de situaciones reales.
- Se señala positivamente la facilidad que brinda la implementación del algoritmo para la visualización de escenarios de mejora que contribuyan a orientar la toma de decisiones antes de iniciarse en la MPS.
- Se considera muy útil la existencia de un algritmo que procese la información referente a la interación entre las buenas prácticas y los FCE, para dirigir los esfuerzos de las organizaciones con intenciones de iniciarse en programas de mejora, teniendo como base los resultados obtenidos por otras organizaciones con comportamientos similares.
- Se subraya como elemento positivo la generalidad del algoritmo propuesto y su flexibilidad para incorporar nuevos indicadores, FCE y medidas.
- El sistema permite llegar a todos los centros u organizaciones involucrados en la MPS para mejorar su estado frente a los programas de MPS.

Las recomendaciones emitidas por los encuestados estuvieron dirigidas a:

- Se considera favorable visualizar además la factibilidad de los escenarios de mejora propuestos para enfrentar exitosamente la MPS.

La aplicación de la técnica de Iadov aportó datos significativos respecto al grado de satisfacción de los clientes. Los resultados obtenidos y los criterios emitidos validan la fortaleza del algoritmo, reflejándose una opinión muy positiva respecto a la satisfacción del cliente con implementación del mismo, así como la consideración de que la reutilización de experiencias que realiza el mismo proporcionar estados de mejora, resulta útil y aplicable en entornos reales.

Impacto económico-social

La implantación del algoritmo propuesto en la presente investigación, tiene un impacto positivo muy acorde con el cumplimiento de los lineamientos de la política económica y social de nuestro país. Se destaca y evidencia el impacto de la propuesta de solución en los lineamientos (GRANMA 2011):

Lineamiento 83, “Trabajar para garantizar, por las empresas y entidades vinculadas a la exportación, que todos los bienes y servicios destinados a los mercados internacionales respondan a los más altos estándares de calidad”

Lineamiento 131, “Sostener y desarrollar los resultados alcanzados en el campo de la biotecnología, la producción médico-farmacéutica, la industria del software y el proceso de informatización de la sociedad, las ciencias básicas, las ciencias naturales, los estudios y el empleo de las fuentes de energía renovables, las tecnologías sociales y educativas, la transferencia tecnológica industrial, la producción de equipos de tecnología avanzada, la nanotecnología y los servicios científicos y tecnológicos de alto valor agregado”

El algoritmo que se propone permite que las organizaciones enfrenten la MPS en mejores condiciones y conociendo posibles escenarios de mejora alcanzables para obtener mejores resultados, lo cual es indispensable para incrementar sus estándares de calidad, generando como consecuencia un alto valor agregado en la industria del software. El algoritmo puede ser aplicado en los centros de desarrollo de software de la UCI, así como extenderse a las organizaciones de la industria cubana del software que presenten interés en iniciar la MPS.

El aporte fundamental para las organizaciones que inicien la MPS, está dado por los altos costos de los programas de mejora y el alto índice de fracaso que existe actualmente. Las salidas del algoritmo se convierten en un criterio para emplear correctamente los presupuestos asociados a la MPS en las organizaciones. De esta manera las organizaciones cuentan con escenarios de mejora alcanzables para guiar la toma de

decisiones en los programas de MPS, evitando invertir en el programa hasta tener un escenario favorable para el éxito en la MPS.

Para una mejor visualización de lo antes expresado, se muestran los reportes de costos de algunas organizaciones:

- Costo de la consultoría del programa de MPS ejecutado en la UCI, USD 70000,00.
- Precio de la consultoría que ofrece la empresa MS SPI Solutions, la cual incluye el SCAMPI, USD 25000,00.
- Precio de las consultorías que ofrece el ESI Center del Tecnológico de Monterey el cual incluye el SCAMPI, USD 23000,00; cursos relativos al modelo referencia USD 16000,00; Acompañamiento USD 27000,00; Pre SCAMPI USD 8000,00.
- Costo del SCAMPI que ofrece la empresa Procesix, USD 52000,00.
- Costo de la consultoría y el SCAMPI que ofrece la empresa América XXI, USD 83000,00.

Con el algoritmo y la implementación que se proponen, las organizaciones pueden mejorar sus condiciones para iniciar la MPS con sus propios recursos, teniendo el valor agregado de la información que se brinda a partir de la reutilización de las experiencias en torno a los FCE.

Conclusiones parciales del capítulo

A partir de los elementos abordados en el capítulo se concluye:

La validación del AG para la optimización de estados mediante la aplicación del pre-experimento, demostró la efectividad del mismo para obtener escenarios de mejora que guíen los esfuerzos hacia el éxito en la MPS.

Los resultados de la aplicación de la técnica de Iadov reflejaron un alto nivel de satisfacción por parte del cliente, dejando en evidencia una concordancia del 90,00% con la calificación de “máxima satisfacción” respecto a la utilidad y aplicabilidad del algoritmo para guiar los esfuerzos en la MPS y el otro 10,00% lo calificó como “más satisfecho que insatisfecho”.

Se señaló positivamente por los clientes encuestados la obtención de escenarios de mejora para guiar los esfuerzos de las organizaciones y apoyar la toma de decisiones antes de iniciarse en la MPS.

Los elevados costos de los programas de mejora y el alto índice de fracasos, fundamentan la necesidad de aplicación del AG para proveer escenarios de mejora alcanzables y así emplear correctamente los presupuestos asociados a la MPS en las organizaciones.

Conclusiones

El análisis de los referentes teóricos entorno al objeto y campo de acción evidenció la necesidad de guiar los esfuerzos de mejora de las organizaciones en la MPS. Para ello se determinó esencial la optimización de estados en la MPS mediante la aplicación de AG.

Para la concepción del AG de optimización de estados en la MPS se diseñó un operador de selección y se modificaron operadores de cruzamiento y mutación, estimulando la generación de nuevos individuos con mejores condiciones de adaptación respecto al objetivo de optimización, para dar solución al problema planteado.

La aplicación del pre-experimento validó la efectividad del AG para la optimización de estados y corroboró que tiene un efecto positivo en la reutilización de experiencias para guiar las decisiones en función del éxito en la MPS.

Los resultados de la aplicación de la técnica de Iadov evidenciaron una concordancia del 90,00% con la calificación de “máxima satisfacción” respecto a la utilidad y aplicabilidad del algoritmo para apoyar la toma de decisiones frente a programas de MPS, mientras que el otro 10,00% lo calificó como “más satisfecho que insatisfecho”, lo cual reflejar un alto nivel de satisfacción por parte del cliente con la solución propuesta.

Recomendaciones

Incorporar a la solución visualizar además la factibilidad de los escenarios de mejora propuestos para enfrentar exitosamente la MPS.

Referencias bibliográficas

ALLISON, I. *Organizational Factors Shaping Software Process Improvement in Small-Medium Sized Software Teams: a Multi-Case Analysis*. *Quality of Information and Communications Technology (QUATIC), 2010 Seventh International Conference on the*. Porto, IEEE Xplore Digital Library, 2010. 418 - 423

ARRANZ DE LA PEÑA, J. and A. PARRA TRUYOL Algoritmos Genéticos *Universidad Carlos III*, 2014.

ASHRAFI, N. The impact of software process improvement on quality: in theory and practice *Inf. Manage*, 2003, 40(7): 677-690.

BABAR, M. A. and M. NIAZI. *Implementing Software Process Improvement Initiatives: An Analysis of Vietnamese Practitioners' Views*. *Proceedings of the 2008 IEEE International Conference on Global Software Engineering*, IEEE Computer Society, 2008. 67-76.

BASILI, V. R.; F. E. MCGARRY*, et al. Lessons learned from 25 years of process improvement: the rise and fall of the NASA software engineering laboratory*. *Proceedings of the 24th International Conference on Software Engineering*. Orlando, Florida, ACM, 2002. 69-79.

BAUER, F. L. *Software Engineering, Information Processing*. *Software Engineering*. Amsterdam, North Holland Publishing Co, 1972. 530.

BOAS, G. V.; A. R. C. DA ROCHA*, et al. An Approach to Implement Software Process Improvement in Small and Mid Sized Organizations*. *2010 Seventh International Conference. Quality of Information and Communications Technology (QUATIC)*. Porto, IEEE Xplore Digital Library, 2010. 447 - 452

BOEHM, B. W. Software Engineering *Computers, IEEE Transactions on*, 1976, C-25(12): 1226 - 1241.

BOOCH, G. *The Unified Modeling Language User Guide*. Pearson Education, 2005. p.

CALISOFT, C. N. D. C. D. S. *Libro del Diagnóstico UCI -2012*. La Habana, CALISOFT. Centro Nacional de Calidad de Software, 2012. 94.

CARNEGIE-MELLON. *Software Engineering Institute*, University Carnegie Mellon. U.S. Department of Defense., 2013. [2013]. Disponible en: http://www.sei.cmu.edu/

CASE, D. O. *Looking for information: A survey of research on information seeking, needs and behavior*. Emerald Group Publishing, 2012. p. 1780526547

CASTRO, F. La Ciencia para el desarrollo en el Siglo XXI *Anales de la Academia de Ciencias de Cuba*, 2013, 2(2): 10.

CATTANEO, F.; A. FUGGETTA*, et al.* Pursuing coherence in software process assessment and improvement *Software Process: Improvement and Practice*, 2001, 6(1): 3-22.

CAVALCANTI, A. R. and K. CHAVES. *MPS.BR Lecciones Aprendidas*, Sistemas de Bibliotecas de la UNICAMP, 2008. [2013]. Disponible en: http://www.softex.br/mpsbr/_livros/licoes/mpsbr_es.pdf

CMMI, I. *CMMI Institute. Clearmodel*, Carnegie Mellon, 2016. [2016]. Disponible en: http://cmmiinstitute.com/

---. *Published Appraisal Results*, 2015. [Disponible en: https://sas.cmmiinstitute.com/pars/pars_detail.aspx?a=25323

COELLO, C. A. C.; D. A. VAN VELDHUIZEN*, et al. Evolutionary algorithms for solving multi-objective problems*. Springer, 2002. p.

COELLO, C. A. C. and C. S. P. ZACATENCO Introducción a la computación evolutiva (Notas de curso) *CINVESTAV-IPN, Departamento de Ingeniería Eléctrica, Sección de Computación. México, DF*, 2004.

COMPETISOFT, P. *COMPETISOFT. Mejora de Procesos para Fomentar la Competitividad de la Pequeña y Mediana Industria del Software de Iberoamérica* Ciencia y Tecnología para el Desarrollo - CYTED (3789) 2008.

CONCEPCIÓN, L. E. P.; Z. E. M. RODRÍGUEZ*, et al.* ALGORITMOS GENÉTICOS, SUS PROPUESTAS DE APLICACIÓN: APRENDIZAJE CORPORATIVO *Revista de investigación de Sistemas e Informática*, 2004, 1(1): 59-67.

CHOO, C. W. and D. R. DÍAZ. *La organización inteligente: el empleo de la información para dar significado, crear conocimiento y tomar decisiones*. Oxford university press México DF, 1999. p. 9706134476

DEB, K.; A. PRATAP*, et al.* A fast and elitist multiobjective genetic algorithm: NSGA-II *evolutionary computation, IEEE transactions on*, 2002, 6(2): 182-197.

DÍAZ DUARTE, D. Toma de decisiones: el imperativo diario de la vida en la organización moderna *Acimed*, 2005, 13(3): 1-1.

DJANGO. *Django*, 2016. [2016]. Disponible en: http://django.es/

DOUNOS, P. and G. BOHORIS. *Factors for the Design of CMMI-based Software Process Improvement Initiatives*. *Informatics (PCI), 2010 14th Panhellenic Conference on* Tripoli IEEE Xplorer Digital Library, 2010. 43 - 47.

FOGEL, L. J. *Artificial Intelligence Through Simulated Evolution.[By] Lawrence J. Fogel... Alvin J. Owens... Michael J. Walsh*. John Wiley & Sons, 1966. p.

FONSECA, C. M. and P. J. FLEMING. *Genetic Algorithms for Multiobjective Optimization: FormulationDiscussion and Generalization*. ICGA, Citeseer, 1993. 416-423 p.

FORRADELLAS, P.; G. PANTALEO*, et al. El modelo CMM/CMMI - Cómo garantizar el éxito del proceso de mejoras en las organizaciones, superando los conflictos y tensiones generados por su implementación*. Universidad CAECE, Av. de Mayo 866, Capítulo Argentino de la IEEE COMPUTER SOCIETY e it-Mentor, 2005. 21.

FOUNDATION, T. A. S. *Documentación del Servidor HTTP Apache 2.0*, 2013. [2016]. Disponible en: https://httpd.apache.org/docs/2.0/

FOWLER, M. and K. SCOTT. *UML gota a gota*. Pearson Education, 1999. p.

GARCIA, A. M. *Proceso para pronosticar el éxito en la Mejora de Procesos de Software*. Calidad de Software, Universidad de las Ciencias Informáticas, 2013. p.

GARCIA, A. M.; Y. TRUJILLO*, et al.* Pronóstico de éxito en la Mejora de Procesos de Software *RCCI*, 2016: 15-30.

GAUCHAT, J. D. *El gran libro de HTML5, CSS3 y Javascript*. Marcombo, 2012. p. 8426717829

GLOVER, F. W. and G. A. KOCHENBERGER. *Handbook of metaheuristics*. Springer Science & Business Media, 2006. p. 0306480565

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989. p.

GRANMA. *Lineamientos de la política económica y social del Partido y la Revolución*. *Granma*. La Habana, VI Congreso del Partido Comunista de Cuba, 2011.

HERRERA, F. Introducción a los algoritmos metaheurísticos *Ciencias de la Computación e IA*, 2006.

HERTZ, A. and D. KOBLER A framework for the description of evolutionary algorithms. *European Journal of Operational Research*, 2000, 126(1): 1-12.

HOLOVATY, A. and J. KAPLAN-MOSS. *The definitive guide to Django: Web development done right*. Apress, 2009. p. 1430219378

HOLLAND, J. H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. U Michigan Press, 1975. p. 0472084607

--- Concerning efficient adaptive systems *Self-Organizing Systems*, 1962, 230.

HORN, J.; N. NAFPLIOTIS*, et al. A niched Pareto genetic algorithm for multiobjective optimization*. Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on, Ieee, 1994. 82-87 p. 0780318994

HOUCK, C. R. J., JEFF

KAY, MICHAEL G. A genetic algorithm for function optimization: a Matlab implementation *NCSU-IE TR*, 1995, 95(09).

HUMPHREY, W. S. *A Discipline for Software Engineering*. 1st. Inc. Boston, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA ©1995 1995. 816 p. 0201546108

IEEE. *610-1990 - IEEE Standard Computer Dictionary: Compilation of IEEE Standard Computer Glossaries*. IEEE Xplore Digital Library, IEEE Computer Society, 1990.

ISLAM, Q. N. *Mastering PyCharm*. Packt Publishing, 2015. p. 1783551321

ISO. *ISO 9000: 2005. Quality management systems*. ISO, 2005a.

---. *ISO. International Organization for Standardization*, ISO, 1997. 2013.

---. *ISO/IEC 15504-1:2004. Information technology*. ISO, 2004.

---. *ISO/IEC 25000:2005. Software Engineering -- Software product Quality Requirements and Evaluation (SQuaRE)*, ISO, 2005b.

KABIR, M. J. *Apache Server Bible*. IDG Books Worldwide, Inc., 1998. p. 0764532189

KAUTZ, K.; H. W. HANSEN*, et al. Applying and adjusting a software process improvement model in practice: the use of the IDEAL model in a small software enterprise*. *Proceedings of the 22nd international conference on Software engineering*. Limerick, Ireland, ACM, 2000. 626-633.

KOSA, J. Genetic programming: On the programing of computers by natural selection. *MIT Press*, 1992.

KUZMINA, N. V. *Metódicas investigativas de la actividad pedagógica*, Leningrado, 1970.

LAPORTE, C. Y. and S. TRUDEL Addressing the people issues of process improvement activities at Oerlikon Aerospace *Software Process: Improvement and Practice*, 1998, 4(4): 187-198.

LI, E. Y.; H.-G. CHEN*, et al. Software Process Management in Taiwan´s top 1000 companies: a longitudinal analysis*. *11th Annual Conference of Asia Pacific Decision Sciences Institute*. Hong Kong, 2006.

LUTZ, M. *Learning python*. " O'Reilly Media, Inc.", 2013. p. 1449355714

MATEOS ANDALUZ, A. *Inteligencia en Redes de Comunicaciones. Algoritmos Evolutivos y Algoritmos Genéticos*. Ingeniería de Telecomunicación, Universidad Calos III de Madrid, 2004. p.

MATHIASSEN, L. and P. POUYA Managing knowledge in a software organization *Journal of Knowledge Management*, 2003, 7(2).

MATURRO, G. *Modelo para la gestión del conocimiento y la experiencia integrada a las prácticas y procesos de desarrollo software*. Facultad de Informática. Madrid, Universidad Politécnica de Madrid, 2010. 393. p.

MCFEELEY, B. *IDEAL: A User's Guide for Software Process Improvement*, DTIC Document, 1996.

MEBANE JR, W. R. Genetic optimization using derivatives: the rgenoud package for R *Journal of Statistical Software*, 2011, 42(11): 1-26.

MODE, E. B. *Elementos de probabilidad y estadística*. Reverté, 1990. p. 8429150927

MOITRA, D. Managing change for software process improvement initiatives: a practical experience-based approach *Software Process: Improvement and Practice*, 1998, 4(4): 199-207.

MONTONI, M. A. and A. R. ROCHA. *Applying Grounded Theory to Understand Software Process Improvement Implementation*. *Proceedings of the 2010 Seventh International Conference on the Quality of Information and Communications Technology*, IEEE Computer Society, 2010. 25-34.

MÜLLER, S. D.; L. MATHIASSEN*, et al.* Software Process Improvement as organizational change: A metaphorical analysis of the literature *Journal of Systems and Software*, 2010, 83(11): 2128-2146.

MUÑOZ, M. A.; J. A. LÓPEZ*, et al.* Inteligencia de enjambres: sociedades para la solución de problemas (una revisión) *Revista Ingeniería e Investigación*, 2008, 28(2): 119-130.

NETWORK, M. D. and I. CONTRIBUTORS. *Firefox OS architecture*, línea][Citado el: 2013 de Junio de 14.] https://developer. mozilla. org/en-US/docs/Mozilla/Firefox_OS/Platform/Architecture, 2014.

NGWENYAMA, O. K. Competing values in software process improvement: an assumption analysis of CMM from an organizational culture perspective *Engineering Management, IEEE Transactions on*, 2003, 50(1): 100 - 112

NIAZI, M.; M. A. BABAR*, et al.* Software Process Improvement barriers: A cross-cultural comparison *Information and Software Technology*, 2010, 52(11): 1204-1216.

NIAZI, M.; D. WILSON*, et al.* Critical success factors for software process improvement implementation: an empirical study *Software Process: Improvement and Practice*, 2006, 11(2): 193-211.

NYCE, S. C. *NMX-I-059-NYCE (MoProSoft)* Normalización y Certificación Electrónica S.C. (NYCE), 2011. [2016]. Disponible en: http://www.moprosoft.com.mx/

OROZCO, S. Algoritmos Genéticos *Guatemala, Febrero de*, 2007.

PADRÓN, J. M. and I. A. N. MORA Posicionamiento óptimo de enrutadores en WSAN mediante algoritmos evolutivos, 2014.

PARADIGM, V. Visual paradigm for uml *Visual Paradigm for UML-UML tool for software application development*, 2013.

PETTIS, S. L. Una aplicación de programación genética al área de control, 2005.

PGADMIN. *pgAdmin PostgreSQL Tools*, pgAdmin, 2016. [2016]. Disponible en: https://www.pgadmin.org/

PINO, F. J.; F. GARCIA*, et al. Key processes to start software process improvement in small companies*. *Proceedings of the 2009 ACM symposium on Applied Computing*. Honolulu, Hawaii, ACM, 2009. 509-516.

PINO, F. J.; F. GARCÍA*, et al.* Software process improvement in small and medium software enterprises: a systematic review *Software Quality Journal*, 2008, 16(2): 237-261.

PINO, V. H. and A. O. MONDOL Algoritmos genéticos con codificación real en la optimización de funciones de una variable real. *Innovación Tecnológica*, 2010, 15(4).

PMI. *A Guide to the Project Management Body of Knowledge*, 5th. Project Management Institute, 2013a. [2013]. Disponible en: http://www.pmi.org/PMBOK-Guide-and-Standards.aspx

---. *PMI. Project Management Institute.*, 6.2. Project Management Institute, Inc., 2013b. [2013]. Disponible en: http://www.pmi.org/default.aspx

PRESSMAN, R. S. *Ingenieria del Software - Un Enfoque Practico 5b: Edicion*. McGraw-Hill Companies (February 2002), 2002. p. ISBN-10: 8448132149

ISBN-13: 978-8448132149

---. *Software Engineering: A Practitioner's Approach,* . 7th. New York, Raghothaman Srinivasan, 2010. 870 p. *McGraw-Hill Companies, Inc.* 978-0-07-337 597-7

PURIS, A. and R. BELLO. *Optimización basada en Mallas Dinámicas. Su aplicación en la solución de problemas de optimización continuos*. *VI Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB'09)*, 2009.

RAE. *Real Academia Española*, Real Academia Española, 2016. [Disponible en: http://www.rae.es/

RAMÍREZ, D. and R. SANTANA PROPUESTA DE PROCESO PARA LA DEFINICIÓN DE ACTIVOS DE PROCESOS DE LA ORGANIZACIÓN PARA LAS PYMES DE DESARROLLO DE SOFTWARE *Fordes. 13 Semana Tecnológica. III Taller Nacional de Atención a la Población y Calidad de los Servicios*, 2013: 12.

RECHENBERG, I. Cybernetic solution path of an experimental problem, 1965.

ROSAS, M.; H. A. LEIVA*, et al. Técnicas de niching: estrategias evolutivas vs. algoritmos genéticos*. VI Congreso Argentino de Ciencias de la Computación, 2000. p.

SANTOS, G.; M. KALINOWSKI*, et al. MPS.BR: A Tale of Software Process Improvement and Performance Results in the Brazilian Software Industry*. *Proceedings of the 2010 Seventh International Conference on the Quality of Information and Communications Technology*, IEEE Computer Society, 2010. 412-417.

SCHAFFER, J. D. *Multiple objective optimization with vector evaluated genetic algorithms*. Proceedings of the 1st international Conference on Genetic Algorithms, L. Erlbaum Associates Inc., 1985. 93-100 p. 0805804269

SCHEIN, E. *Process consultation.* Cambridge, Addison-Wesley Publishing Company, 1988. 81.

SINERTIC. *ESI Center*, ESICENTER SINERTIC ANDINO, 2011. [2013]. Disponible en: http://www.esicenter-sinertic.org/index.php?option=com_content&task=view&id=33&Itemid=72

SOFTEX. *MPS.BR-Mejora de Proceso del Software Brasileño*, Ana Regina C. Rocha

Cristina Ângela Filipak Machad. Sistemas de Bibliotecas de la UNICAMP, 2012. [2016]. Disponible en: http://www.softex.br/wp-content/uploads/2013/10/MPS.BR_Gu%C3%ADa_General_Software_2012.pdf

SOMMERVILLE, I. *Software Engineering*. 8. Addison-Wesley, 2007. 840 p. 9780321313799

SRINIVAS, N. and K. DEB Muiltiobjective optimization using nondominated sorting in genetic algorithms *Evolutionary computation*, 1994, 2(3): 221-248.

STANDISH-GROUP. *Chaos Manifiesto 2015*. *The Standish Group*, The Standish Group International, Inc, 2015.

STELZER, D. and W. MELLIS. *Success Factors of Organizational Change in Software Process Improvement*. SOFTWARE PROCESS–IMPROVEMENT AND PRACTICE, Citeseer, 1999. p.

STEVENS, P. and R. POOLEY. *Utilización de UML en Ingeniería del Software con Objetos y Componentes*. 2. Addison-Wesley Publishing Company, 2002. p.

TALBI, E.-G. *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009. p. 0470496908

TOMASSINI, M. *A survey of genetic algorithms*. *Annual Reviews of Computational Physics*, 1995.

TORRES, B. C. *Metodología para la optimización de múltiples objetivos basada en ag y uso de preferencias*, Master's thesis, Universidad Nacional de Colombia, 2009. p.

TORRES, M.; K. PAZ*, et al.* Tamaño de una muestra para una investigación de mercado *Boletín electrónico*, 2006, 2.

TRUJILLO, Y. *Modelo para valorar las organizaciones desarrolladoras de software al iniciar la mejora de procesos*, Universidad de las Ciencias Informáticas, 2014. p.

TRUJILLO , Y.; A. FEBLES *, et al.* Diagnóstico al iniciar la mejora de proceso de software *Ingeniería Industrial*, 2014, 35(2): 172-183.

TRUJILLO, Y.; A. FEBLES*, et al. Modelo para valorar las organzaciones previo a la mejora de proceso de software*. VI Taller de Calidad en las Tecnologías de la Información y las Comunicaciones en la XV Convención y Feria Internacional Informática 2013, Habana, Cuba, Informática 2013, 2013. 10 p. 978-959-7213-02-4

UCI. *Programa de Mejora. Procesos y Guías*, Universidad de las Ciencias Informáticas, 2011.

VALENCIA, E. *Optimización mediante algoritmos genéticos*. Anales del Instituto de Ingenieros de Chile, 1997. 83-92 p.

VÉLEZ, M. C. and J. A. MONTOYA Metaheurísticos: una alternativa para la solución de problemas combinatorios en administración de operaciones *Revista EIA*, 2007, (8): 99-115.

VIRTANEN, P.; S. PEKKOLA*, et al. Why SPI Initiative Failed: Contextual Factors and Changing Software Development Environment*. *Proceedings of the 2013 46th Hawaii International Conference on System Sciences*, IEEE Computer Society, 2013. 4606-4615.

VITERI, K.; C. SALAZAR*, et al.* Algoritmos genéticos, 2009.

ZAHRAN, S. *Software Process Improvement: Practical Guidelines for Business Success*. ADDISON WESLEY Publishing Company Incorporated. 1998. p. 978-0201177824

ZITZLER, E.; M. LAUMANNS*, et al. SPEA2: Improving the strength Pareto evolutionary algorithm*, Eidgenössische Technische Hochschule Zürich (ETH), Institut für Technische Informatik und Kommunikationsnetze (TIK) Zürich, Switzerland, 2001.

ZITZLER, E. and L. THIELE Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach *evolutionary computation, IEEE transactions on*, 1999, 3(4): 257-271.

ANEXO # 1: Indicadores – FCE – Medidas Base

Indicadores	Factores Críticos de Éxito	Medidas Base
Influencia del personal	Relaciones Interpersonales	Colaboración – Competencia
		Relaciones individuo – individuo
		Relaciones intergrupales
	Formación del personal	Formación para la mejora de procesos
		Capacidad de Aprendizaje
		Capacidad de Adaptación y Autorrenovación
	Experiencia del personal	Experiencias en la producción
		Experiencias en roles
	Efectividad del programa de reconocimiento y remuneración	Reconocimientos y Castigos
		Satisfacción con la Política de Retribuciones
		Satisfacción con la Política de Estimulaciones
	Motivación y compromiso del personal	Motivación por el Trabajo
		Satisfacción con el Trabajo
		Identificación con la Organización
Influencia de la alta gerencia	Orientación estratégica	Orientación a la mejora continua
		Orientación a la satisfacción del cliente
		Orientación a procesos
		Gestión del cambio
	Administración estratégica	Planeación estratégica
		Establecimiento y dominio de los objetivos organizacionales
		Establecimiento y delimitación de roles organizacionales
	Apoyo de la alta gerencia	Confianza en la dirección
		Competencia de los directivos
		Supervisión
		Estilo de Dirección

		Relaciones Jefe – Subordinados
	Atención al capital humano	Selección de personal e inducción a la organización
		Programas de Desarrollo y planes de superación
		Evaluación del desempeño
		Protección e higiene del trabajo
Características de la organización	Disponibilidad de recursos	Disponibilidad de las personas
		Disponibilidad de tiempo
		Disponibilidad de infraestructura
	Comunicación	Participación
		Información
		Comunicación
	Funcionamiento	Perspectivas de la organización
		Eficiencia
		Eficacia
		Estabilidad interna de la organización
		Trabajo en Equipo

Tabla 8: Indicadores-FCE-Medidas
Fuente: (TRUJILLO 2014)

ANEXO # 2: Descripción de las clases persistentes

Nombre: Pais	
Descripción: Detalla los países que se involucran en programas de mejora.	
Atributo	**Descripción**
id	Identificador
nombre	Nombre del país
Descripción de los métodos	
Método	Descripción
insertarPais	Busca el objeto Pais recibido como parámetro en la base de datos. En caso de no encontrarlo, lo registra.
eliminarPais	Busca el objeto Pais recibido como parámetro en la base de datos. En caso de encontrarlo, lo elimina.

Tabla 9: Descripción de la clase Pais
Fuente: Elaboración propia

Nombre: Organismo	
Descripción: Detalla los organismos que se involucran en programas de mejora.	
Atributo	**Descripción**
id	Identificador
nombre	Nombre del organismo
descripcion	Descripción del organismo
Descripción de los métodos	
Método	Descripción
insertarOrganismo	Busca el objeto Organismo recibido como parámetro en la base de datos. En caso de no encontrarlo, lo registra.
eliminarOrganismo	Busca el objeto Organismo recibido como parámetro en la base de datos. En caso de encontrarlo, lo elimina.

Tabla 10: Descripción de la clase Organismo
Fuente: Elaboración propia

Nombre: Organizacion	
Descripción: Agrupa las organizaciones que se involucran en programas de mejora.	
Atributo	**Descripción**
id	Identificador
nombre	Nombre de la organización
descripcion	Descripción de la organización
Descripción de los métodos	
Método	Descripción
insertarOrganizacion	Busca el objeto Pais u Organismo recibido como parámetro en la base de datos. En caso de no encontrarlo, lo registra.
eliminarOrganizacion	Busca el objeto Pais u Organismo recibido como parámetro en la base de datos. En caso de encontrarlo, lo elimina.
listarPaises	Muestra todos los países que se encuentran en la base de datos
listarOrganismos	Muestra todos los organismos que se encuentran en la base de datos

Tabla 11: Descripción de la clase Organizacion
Fuente: Elaboración propia

Nombre: Mejora_de:procesos_de_software	
Descripción: Detalla los programas de Mejora de Procesos de Software	
Atributo	**Descripción**
id	Identificador
nombre	Nombre del programa de MPS
descripcion	Describe el programa de MPS
Descripción de los métodos	
Método	Descripción
insertarMPS	Busca el objeto MPS recibido como parámetro en la base de datos. En caso de no encontrarlo, lo registra.

eliminarMPS	Busca el objeto MPS recibido como parámetro en la base de datos. En caso de encontrarlo, lo elimina.

Tabla 12: Descripción de la clase Mejora_de_procesos_de_software
Fuente: Elaboración propia

Nombre: Medida_base	
Descripción: Detalla las medidas base que describen a una organización	
Atributo	**Descripción**
id	Identificador
nombre	Nombre de la medida base
descripcion	Descripción de la medida base
impacto	Impacto de la medida base
Descripción de los métodos	
Método	Descripción
insertarMedidaBase	Busca el objeto Medida_base recibido como parámetro en la base de datos. En caso de no encontrarlo, lo registra.
eliminarMedidaBase	Busca el objeto Medida_base recibido como parámetro en la base de datos. En caso de encontrarlo, lo elimina.

Tabla 13: Descripción de la clase Medida_base
Fuente: Elaboración propia

Nombre: Factor_critico_de_exito	
Descripción: Detalla los Factores Críticos de Éxito que inciden en los programas de mejora.	
Atributo	**Descripción**
id	Identificador
nombre	Nombre del FCE
descripcion	Descripción del FCE
Descripción de los métodos	
Método	Descripción

insertarPais	Busca el objeto Factor_critico_de_exito recibido como parámetro en la base de datos. En caso de no encontrarlo, lo registra.
eliminarPais	Busca el objeto Factor_critico_de_exito recibido como parámetro en la base de datos. En caso de encontrarlo, lo elimina.
listarMedidas	Retorna las medidas que componen el FCE

Tabla 14: Descripción de la clase Factor_critico_de_exito
Fuente: Elaboración propia

Nombre: Indicado	
Descripción: Detalla los Factores Críticos de Éxito que inciden en los programas de mejora.	
Atributo	**Descripción**
id	Identificador
nombre	Nombre del indicador
descripcion	Descripción del indicador
Descripción de los métodos	
Método	Descripción
insertarIndicador	Busca el objeto Indicador recibido como parámetro en la base de datos. En caso de no encontrarlo, lo registra.
eliminarIndicador	Busca el objeto Indicador recibido como parámetro en la base de datos. En caso de encontrarlo, lo elimina.
listarFCE	Retorna las FCE por indicador.

Tabla 15: Descripción de la clase Indicador
Fuente: Elaboración propia

ANEXO # 3: Descripción de las tablas de la base de datos

Nombre: pais	
Descripción: Almacena los países involucrados en programas de mejora.	
Atributo	**Descripción**
id	Identificador
nombre	Nombre del país

Tabla 16: Descripción de la tabla pais
Fuente: Elaboración propia

Nombre: organismo	
Descripción: Almacena los organismos involucrados en programas de mejora.	
Atributo	**Descripción**
id	Identificador
nombre	Nombre del organismo
descripcion	Descripción del organismo

Tabla 17: Descripción de la tabla organismo
Fuente: Elaboración propia

Nombre: clasificacion_organizacional	
Descripción: Almacena las clasificaciones de las organizaciones involucrados en programas de mejora.	
Atributo	**Descripción**
id	Identificador
clasificacion	Clasificación que reciben las organizaciones
descripcion	Descripción de la clasificación

Tabla 18: Descripción de la tabla clasificacion_organizacional
Fuente: Elaboración propia

Nombre: organizacion	
Descripción: Almacena las organizaciones involucradas en programas de mejora.	
Atributo	**Descripción**
id	Identificador
nombre	Nombre de la organización
descripcion	Descripción de la organización

paisid	Identificador del país
organismoid	Identificador del organismo
clasificación_organizacionalid	Identificador de la clasificación organizacional

Tabla 19: Descripción de la tabla organizacion
Fuente: Elaboración propia

Nombre: mejora_de_procesos_de_software	
Descripción: Almacena programas de MPS.	
Atributo	**Descripción**
id	Identificador
nombre	Nombre del programa de MPS
descripcion	Descripción del programa de MPS

Tabla 20: Descripción de la tabla mejora_de_procesos_de_software
Fuente: Elaboración propia

Nombre: indicador	
Descripción: Almacena los indicadores o conjunto de métricas que inciden en la MPS.	
Atributo	**Descripción**
id	Identificador
nombre	Nombre del indicador
descripcion	Descripción del indicador

Tabla 21: Descripción de la tabla indicador
Fuente: Elaboración propia

Nombre: factor_critico_de_exito	
Descripción: Almacena los FCE u objetivos de medición en los programas de MPS.	
Atributo	**Descripción**
id	Identificador
nombre	Nombre del FCE
descripcion	Descripción del FCE
indicadorid	Identificador del indicador

Tabla 22: Descripción de la tabla factor_critico_de_exito
Fuente: Elaboración propia

Nombre: medida_base	
Descripción: Almacena las medidas base de los FCE	
Atributo	**Descripción**
id	Identificador
nombre	Nombre de la medida base
descripcion	Descripción de la medida base
impacto	Impacto de la medida base
factor_critico_de_exitoid	Identificador del FCE

Tabla 23: Descripción de la tabla medida_base
Fuente: Elaboración propia

Nombre: estado_inicial_medida_base	
Descripción: Relaciona las tuplas estado inicial y medida base	
Atributo	**Descripción**
estado_inicialid	Identificador de los estado inicial
medida_baseid	Identificador de las medidas bases

Tabla 24: Descripción de la tabla estado_inicial_medida_base
Fuente: Elaboración propia

Nombre: escenario_medida_base	
Descripción: Relaciona las tuplas escenario y medida base	
Atributo	**Descripción**
escenarioid	Identificador de los escenarios
medida_baseid	Identificador de las medidas bases

Tabla 25: Descripción de la tabla escenario_medida_base
Fuente: Elaboración propia

ANEXO # 4. Encuesta para valorar la satisfacción de los clientes con la implementación del algoritmo de optimización

El objetivo de la presente encuesta constituye valorar la satisfacción respecto a la utilidad y aplicabilidad del algoritmo de optimización propuesto para guiar los esfuerzos en la toma de decisiones en la MPS, a partir del juicio que usted emita. Según su criterio, responda las siguientes preguntas:

1. ¿Considera usted que las organizaciones deben guiar sus esfuerzos para la mejora sin tener en cuenta las buenas prácticas que pueden aplicar en función de disminuir la acción de los Factores Críticos de Éxito?
   ____Sí ____No sé ____No

2. ¿Considera útil la posibilidad contar con escenarios de mejora para guiar las decisiones de la organización a partir de la reutilización de experiencias teniendo en consideración el impacto de las buenas prácticas sobre los FCE?
3. ____Sí ____No sé ____No

4. ¿En qué medida considera que el algoritmo genético para obtener estados de mejora es aplicable a las necesidades de su organización para la toma de decisiones en la MPS?
   ____Aplicabilidad muy adecuada
   ____Aplicabilidad adecuada
   ____Resulta indiferente
   ____Aplicabilidad poco adecuada
   ____No es nada aplicable
   ____No sé qué decir

5. ¿Qué elemento(s) considera favorables respecto al algoritmo propuesto?

6. ¿Qué recomendación(es) sugiere para mejorar la solución propuesta?