2. MARCO CONCEPTUAL

En este capítulo se describen los conceptos teóricos en los que se enmarca este trabajo, tales como sistemas legados y su metodología de evaluación, reingeniería y refactorización.

2.1 LOS SISTEMAS LEGADOS

Varios autores han hecho diversas definiciones de los sistemas legados. A. Massari y M. Mecella [4] recopilan definiciones tales como "algo de valor recibido de un antepasado, un predecesor o del pasado", "un gran sistema software con el que no se quiere operar pero que es vital para la organización"; según estos autores, las características de los sistemas legados son las siguientes:

- Son sistemas fundamentales para la operatividad de la organización.
- Se ha invertido mucho en ellos en los últimos años y por ende no se puede dejar de lado.
- Generalmente son enormes, con un núcleo diseñado hace más de una década con las consiguientes "viejas ideas".
- Escritos en un lenguaje de vieja generación.
- Son soportados por un sistema de gestión de base de datos obsoleto.
- No tienen interfaz gráfica de usuario.
- Son monolíticos.
- No tienen documentación.
- Son difíciles de comprender.
- En ellos reposan años de experiencia.
- En sus líneas de código se encuentra de manera implícita el modo de operación de la organización [4].

Se han enunciado otras definiciones que complementan la definición de los sistemas legados y las características presentadas anteriormente, desde una perspectiva técnica. A. Dedeke define un sistema legado como un paquete de componentes software y hardware cuyos lenguajes, estándares, códigos y tecnologías pertenecieron a una generación anterior o época de innovación [5]. Otros autores afirman que no necesariamente estas características deben estar presentes a la vez en un sistema legado, por ejemplo, un sistema se pueden considerar legado si su base tecnológica en la que se ha construido es débil, así se haya construido recientemente [6]. De este modo, también se define a un sistema legado como un "sistema software de misión crítica desarrollado alguna vez en el pasado que ha existido y ha cambiado a lo largo del tiempo sin someterse a acciones sistémicas correctivas" [6].

Con base en estas definiciones, se puede concluir que un sistema software legado es aquel que cumple con una misión crítica en la organización, fue desarrollado un tiempo atrás basado en tecnología obsoleta, con pobre documentación y con malas prácticas de software en su proceso evolutivo para la gestión de nuevos requisitos, comprometiendo la calidad del sistema y haciendo difícil su mantenimiento y evolución.

2.2 LA EVALUACIÓN DE UN SISTEMA LEGADO – TRABAJOS RELACIONADOS

Puesto que un sistema legado no es simplemente un sistema software antiguo, sino que es un sistema de cómputo socio - técnico y por ende abarca otro tipo de componentes además de los técnicos como los datos y los procesos de negocio [7], se requiere evaluarlo de manera integral para tomar una decisión adecuada con respecto al mismo y evitar efectos no deseados en la organización.

J. Ransom, I. Sommerville e I. Warren proponen evaluar un sistema legado desde las perspectivas del valor de negocio, una evaluación del ambiente externo (involucrando el hardware, el soporte y la infraestructura organizacional) y una evaluación tecnológica de la aplicación [8]. Una vez realizada la evaluación desde dichas perspectivas, proponen unir la evaluación de las características de hardware, soporte de software y de la aplicación del software en una perspectiva técnica, y comparar dichos valores con la perspectiva de negocio, para proveer una aproximación inicial del tipo de evolución requerido para el sistema, utilizando el método de Nola Norton & Co, que consiste en ubicar al sistema evaluado en un mapa de cuadrantes; en cada cuadrante se determina el estado del sistema legado, y para tomar la decisión adecuada con respecto al mismo. Véase la Figura 2.1.

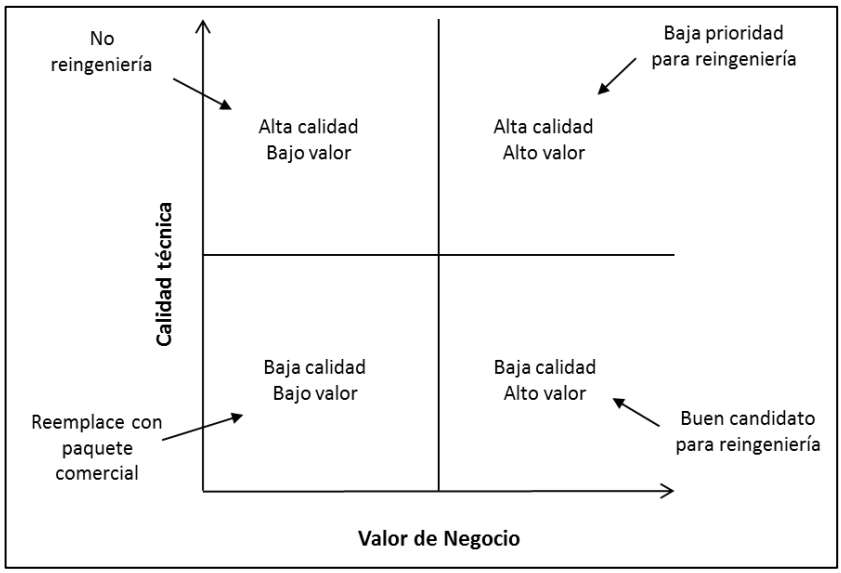

Figura 2.1. Valor Técnico y Valor De Negocio. Fuente [8].

Otra propuesta similar a la anterior es la dada por A. De Lucia, A. R. Fasolino y E. Pomella [6], en la que especifican un marco de trabajo de decisión para la gestión de sistemas legados. El proceso de dicho marco de trabajo consiste en definir unas metas, un análisis de gaps, un análisis de cartera (Portfolio Analysis), la definición de alternativas y la definición de estrategias de conversión. En el análisis de cartera utilizan el mapa de cuadrantes con las mismas especificaciones dadas por S. Ducasse y D. Pollet [1], y hacen una especificación de las variables y las métricas para evaluar las perspectivas técnicas y de negocio. Véase la Tabla 2.1.

ATRIBUTO	VARIABLE	MÉTRICAS
Valor de Negocio	Valor económico	Valor de mercado, índice de rentabilidad, tasa interna de retorno
	Valor de datos	Porcentaje de archivos de misión crítica, porcentaje de archivos de aplicaciones dependientes
	Utilidad	Tasa de cubrimiento de funciones de negocio, frecuencia de uso actual, métricas de satisfacción de usuario
	Especialización	Porcentaje de funciones altamente especializadas, porcentaje de funciones genéricas
Valor Técnico	Mantenibilidad	LOC, Puntos de función, nudos de flujo de control, complejidad ciclomática, tasa de código muerto
	Descomponibilidad	Modularidad de la arquitectura, porcentaje de módulos con separación de intereses
	Deterioro	Incremento de trabajo acumulado, incremento de la tasa de defectos, incremento del tiempo de respuesta, tiempo de mantenimiento por incremento de solicitud.
	Obsolescencia	Edad del sistema, versión del sistema operativo, versión del hardware, disponibilidad del soporte técnico

Tabla 2.1. Modelo de evaluación de un sistema legado. Fuente: [6]

A. Dedeke en [5] hace una propuesta menos rígida que la anteriormente mencionada. Incluye también el análisis de cartera basado en el mapa de cuadrantes; él plantea la definición del valor de negocio por medio de 6 factores para cuantificar este valor: La ventaja competitiva, el impacto en la rentabilidad, el potencial de crecimiento, la estandarización y la conformidad, la interdependencia del sistema y la seguridad del sistema. Para el valor técnico, propone la definición de A. Sage, en el que hace la relación entre calidad, servicio, costo y degradación del sistema.

Este estudio experimental por tanto se basa principalmente en el trabajo realizado por A. De Lucia, A. R. Fasolino y E. Pomella, tomando en cuenta el contexto particular de la organización para proponer las variables y métricas necesarias de acuerdo con las circunstancias particulares en estudio.

2.3 LA REINGENIERÍA Y LA REFACTORIZACIÓN PARA LA EVOLUCIÓN DE UN SISTEMA LEGADO

Generalmente no es fácil distinguir los términos reingeniería y refactorización. Chifofsky y Cross en 1990 ya habían definido la reingeniería de software como el "examen y la alteración de un sistema sujeto para reconstruirlo en una nueva forma y en las implementaciones subsecuentes a dicha implementación" [9]. De acuerdo con esta definición, los términos examen y alteración se refieren formalmente a ingeniería inversa e ingeniería directa, respectivamente. Significa entonces que la reingeniería, al combinar los dos tipos de ingeniería, busca transformar implementaciones concretas a otras implementaciones concretas, enfocándose en las partes más relevantes de un sistema legado [10].

Para cada cambio que se requiera hacer que la reingeniería haya encontrado, se debe evaluar qué partes del sistema requieren ser reestructuradas o se requiere que se implementen desde cero. Cabe resaltar que Chifofsky y Cross también habían definido el término Reestructuración, como "la transformación de una forma de representación a otra en el mismo nivel de abstracción, mientras se preserva el comportamiento externo del sistema". A partir de estas definiciones, Fowler direccionó el término de refactorización como una forma de reestructuración, pero enfocándose al código fuente como tal, al ser una "forma disciplinada para limpiar código que minimiza la probabilidad de introducir errores (bugs). En esencia, cuando usted refactoriza usted está mejorando el diseño del código después de que él ha sido escrito" [10].

Sin embargo, otros expertos definen la refactorización más allá de la mejora de código fuente; Buschmann indica que "la Refactorización no está limitada al detalle del código, sino que puede alcanzar hasta la escala más grande de la arquitectura de un sistema software" [11].

Es importante recalcar que un proceso de refactorización, si bien es cierto conlleva riesgos, también se puede llevar a cabo de manera transversal mientras se continúa con el desarrollo y mantenimiento de los sistemas. Hoy en día la refactorización es una de las prácticas que se encuentra naturalmente integrada dentro de equipos de desarrollo ágiles, como una estrategia de ir mejorando la calidad interna del código a medida que avanza el desarrollo por iteraciones las cuales están orientadas a entregas parciales del producto con funcionalidad significativa para el cliente. Tal como afirma Buschmann, el proceso de refactorización es "muy ligero. No hay necesidad para una estrategia de prueba extendida; podemos usar las pruebas existentes. No hay necesidad para hacer tareas de refactorización altamente visibles, las refactorizaciones son pequeños cambios que pueden ser ejecutados con bajos rituales en cualquier parte del tiempo en los procesos de desarrollo de un sistema." Aduce también que la "refactorización es un poderoso y ágil método para mantener la alta calidad del sistema que se está desarrollando. Si se practica con regularidad, la refactorización tiene un efecto positivo en las habilidades del desarrollador y en los costos del ciclo de vida del sistema" [11].

2.4 REFACTORIZACIÓN DE ARQUITECTURA

El foco de este trabajo es orientar el cambio del sistema desde la perspectiva de refactorización de arquitectura. Es por eso que en esta sección se entra a la investigación en el tema de la refactorización de la arquitectura. Dentro de esta línea se han realizado varios trabajos.

S. Dersten, J. Fröberg, J. Axelsson y R. Land analizaron el impacto que tuvo la refactorización de la arquitectura del software implementado en los automóviles producidos por la compañía Volvo [12]. Otro de los trabajos llevado a cabo en la Technische Universität Ilmenau (Alemania) se enfoca en la evaluación de alternativas para llevar un proceso de refactorización a nivel de arquitectura, por medio de la Teoría de la Decisión [13]. También se han desarrollado trabajos de refactorización de arquitectura en las aplicaciones software para el sector médico, en el cual la gran cantidad de código software legado debe reutilizarse de diversas maneras, y las arquitecturas deben adaptarse a las nuevas formas de trabajar, con los riesgos extremos que esto conlleva [14].

Se han planteado también trabajos para atacar el problema de la erosión de la arquitectura de software a través de la refactorización, por medio de un diseño preliminar de una guía de refactorización para desarrolladores y personal técnico de soporte durante la tarea de revertir un proceso de erosión arquitectónica [15]; otro trabajo realizado por I. Griffith, S. Wahl y C. Izurieta estudiaron la evolución de la comprensibilidad de un sistema legado por medio de un proceso de refactorización automatizada, en donde se utilizan algoritmos evolutivos (es decir, técnicas de Inteligencia Artificial) para realizar el proceso de refactorización sin necesidad de un entendimiento subyacente del software a mejorar [16].

Para la mejora de un sistema se tienen dos alternativas: La reingeniería y la refactorización de la arquitectura. No obstante, como se expuso en el numeral anterior, la reingeniería implica un cambio en el comportamiento externo en el sistema, lo que pudiera implicar la reescritura de código, el desarrollo de nuevas interfaces, transformación de datos, cambios de plataformas tecnológicas, etc., mientras que en la refactorización se busca la mejora de la calidad interna del sistema, mejorar la comprensibilidad y la mantenibilidad del sistema, conservando la funcionalidad actual; además, la refactorización tampoco busca cambios en los procesos de negocio (contrario a lo que busca la reingeniería [17]). Por tales motivos, este trabajo busca estas mejoras internas del sistema para darle un mayor valor al sistema legado con que cuenta la Institución, preservando su comportamiento externo.

3. PLANEACIÓN DEL ESTUDIO

3.1 DEFINICIÓN DEL CONTEXTO

3.1.1 La Institución Universitaria y su Sistema de Información.

La institución universitaria en la cual fue realizado el estudio experimental es la Corporación Universitaria Adventista (Medellín, Colombia); dicha institución fue fundada en el año de 1937, originalmente con el nombre de "Colegio Industrial Coloveno", con el fin de atender las necesidades educativas de la Iglesia Adventista del Séptimo Día en Colombia, Venezuela y las Antillas menores, comprendiendo todos los niveles de educación; en 1981 se crea a partir de esta institución la Corporación Universitaria Adventista, con el objetivo de impartir la educación post – secundaria en la modalidad universitaria [18].

La Institución cuenta con su propio sistema de información, denominado SION (Sistema de Información Organizacional Universitario), por medio del cual busca la integración y cohesión de la información en cada una de sus áreas, propiciando la formalización de procesos y eliminando las islas de información formadas a lo largo de los años de operación; dicho sistema comenzó a desarrollarse a partir del año 2000, y sigue en continua evolución de acuerdo a las necesidades internas y externas de la Institución.

El Sistema SION está dividido por módulos, desarrollados para cada área institucional específica (i.e. contabilidad, presupuestos, tesorería). En la Figura 3.1 se visualizan los componentes del Sistema y la interacción que tienen otros productos software desarrollados en la Institución y con las aplicaciones externas hacia las cuales debe exportar información para el reporte periódico que demandan los organismos de control, tales como el Ministerio de Educación Nacional.

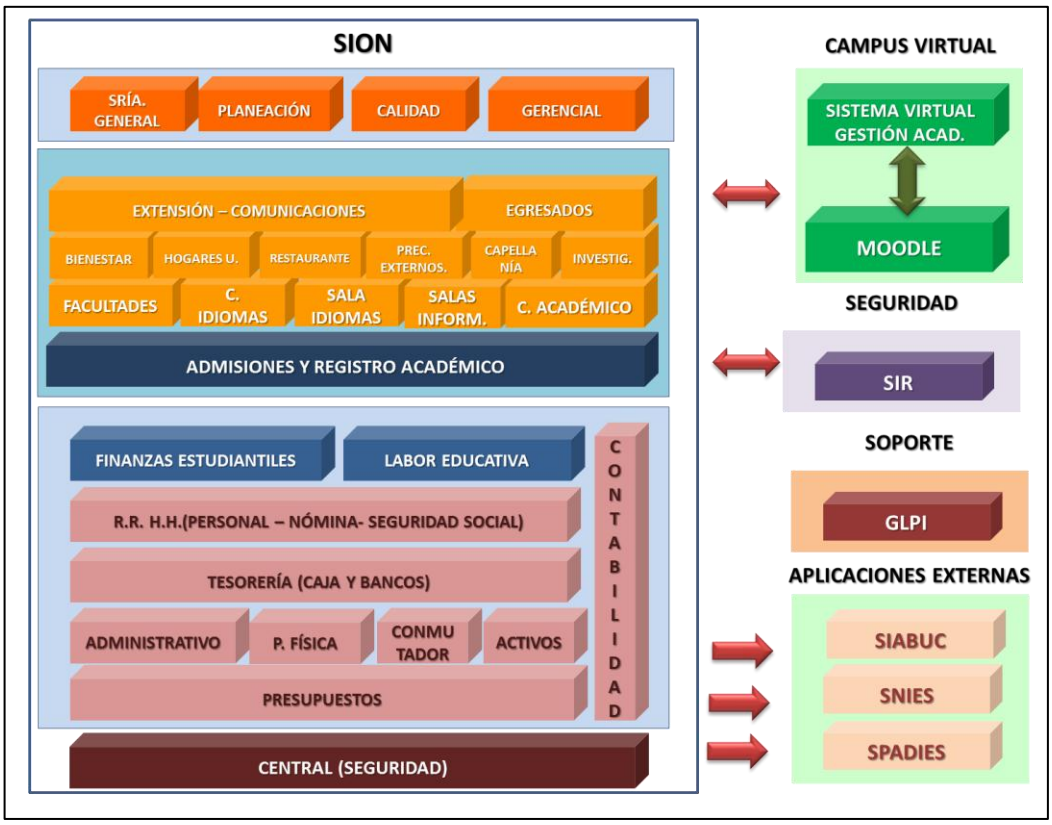

Figura 3.1. Módulos del Sistema SION y los sistemas relacionados en la Corporación Universitaria Adventista. Fuente: Departamento de Sistemas e Informática (DSI) – Corporación Universitaria Adventista.

Este sistema se compone de 29 módulos, los cuales se encuentran clasificados en 4 categorías que el DSI ha establecido: Altamente desarrollado, Medianamente desarrollado, en inicio y planeado a desarrollar. 14 módulos se encuentran en la categoría "Altamente desarrollado", mientras que en la categoría "Medianamente desarrollado" hay 5 módulos; esto significa que 19 módulos se encuentran en producción, pero que hay un alto porcentaje de módulos que están en fase de inicio o planeados a desarrollar (31%). Véase la Figura 3.2.

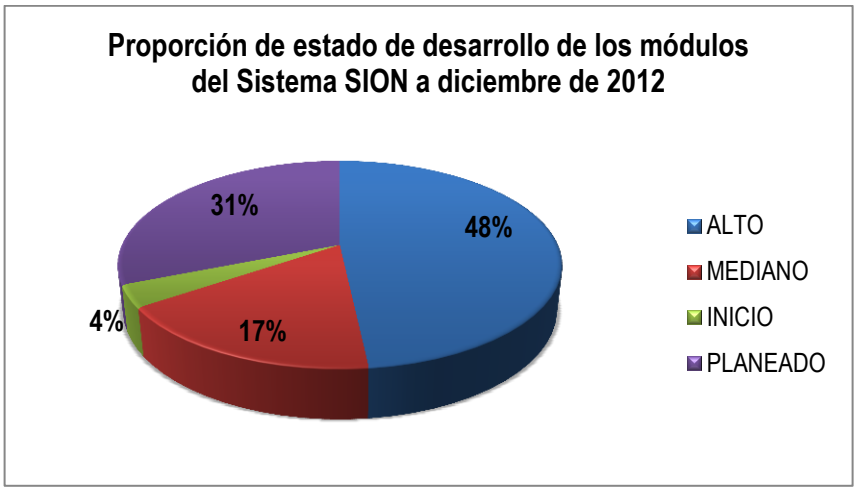

Figura 3.2. Proporción de estado de desarrollo de los módulos del Sistema SION a diciembre de 2012. Fuente: Departamento de Sistemas e Informática (DSI) – Corporación Universitaria Adventista.

De los 19 módulos en producción del Sistema SION, 17 son desarrollados en ambiente de escritorio y 2 en ambiente Web. Para el desarrollo y soporte, el Sistema cuenta con 7 personas, discriminadas de la siguiente manera:

- Dos desarrolladores profesionales.
- Una profesional para el área de soporte (Mesa de Ayuda)
- Dos estudiantes practicantes para el área de desarrollo
- Dos estudiantes practicantes para labores de soporte.

Diversos factores han afectado el desarrollo del sistema, entre ellos, la denominada "Ley de la Entropía del software" (aun cuando un software haya sido bien diseñado desde un principio, la evolución de los requisitos y las nuevas demandas de los clientes hacen que el diseño original se desajuste) [10].

3.1.1 La herramienta de desarrollo base del Sistema de Información referido.

Desde sus inicios se determinó que el IDE de desarrollo fuera MS – Visual Basic 6.0 y como gestor de base de datos MS – SQL Server (en aquella época la versión 7.0). Al paso del tiempo se han hecho algunas funcionalidades de los módulos en tecnología .NET, y otros módulos han sido desarrollados enteramente en esta última tecnología. En la Tabla 3.1 se clasifican los módulos según su herramienta de desarrollo.

MÓDULOS	HERRAMIENTA DE DESARROLLO	ESTADO DE DESARROLLO
Egresados	Visual Basic 6.0	ALTO
Hogares	Visual Basic 6.0	MEDIANO
Restaurante	.NET	MEDIANO
Externos	Visual Basic 6.0	MEDIANO
Facultades	Visual Basic 6.0	ALTO
Idiomas	Visual Basic 6.0	ALTO
Sala idiomas	.NET	ALTO
Salas informáticas	.NET	ALTO
Centro académico	Visual Basic 6.0	ALTO
Admisiones	Visual Basic 6.0	ALTO
Finanzas estudiantiles	Visual Basic 6.0	ALTO
Labor educativa	.NET	ALTO
Recursos Humanos	Visual Basic 6.0	MEDIANO
Tesorería	Visual Basic 6.0	ALTO
Administrativo	Visual Basic 6.0 / .NET	ALTO
Conmutador	Visual Basic 6.0	ALTO
Presupuesto	Visual Basic 6.0	ALTO
Central seguridad	Visual Basic 6.0	MEDIANO
Contabilidad	Visual Basic 6.0	ALTO

Tabla 3.1. Módulos del Sistema SION de acuerdo con su herramienta de desarrollo. Fuente: Departamento de Sistemas e Informática (DSI) – Corporación Universitaria Adventista

14 módulos del Sistema SION han sido desarrollados exclusivamente en la herramienta de desarrollo Visual Basic 6.0, representando un 74%, cifra considerablemente alta; un módulo fue desarrollado en las dos tecnologías (el Administrativo) y 4 en tecnología .NET. Véase la Figura 3.3.

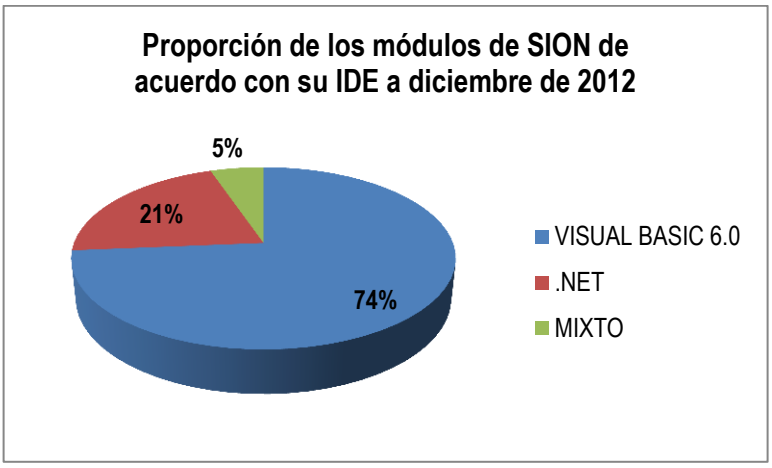

Figura 3.3. Proporción de estado de los módulos del Sistema SION de acuerdo con su herramienta de desarrollo, a diciembre de 2012. Fuente: Departamento de Sistemas e Informática (DSI) – Corporación Universitaria Adventista.

En resumen, la herramienta principal de desarrollo a lo largo de la historia del Sistema SION ha sido Visual Basic 6.0. Esto conlleva a realizar una descripción de dicho entorno de desarrollo para analizar las ventajas y desventajas que esta tecnología conlleva.

- **Descripción general del IDE Visual Basic 6.0.** Visual Basic es un lenguaje de programación dirigido por eventos, con la intención de hacer más simple la programación en un ambiente de desarrollo completamente gráfico basado en Windows. Su primera versión oficial fue lanzada en 1991, prometiendo ser, en palabras de varios expertos de la época, "asombroso", "milagro nuevo increíble", que cambiaría de manera radical la forma en que la gente piensa y utiliza Microsoft Windows, y "un entorno perfecto para la programación de los años noventa" [19]. En aquella época la programación para Windows no era fácil, se requería ser experto en lenguaje C con "10 kilos de documentación y al menos 20 (normalmente más de 30) megabytes de espacio de disco duro para el compilador de C y el resto de los añadidos" [19]. Sin embargo, este lenguaje fue altamente criticado, específicamente al ser lanzada su versión 3.0, porque no era considerado un "verdadero lenguaje de programación", debido a que no era capaz de crear archivos ejecutables reales, debía ser interpretado en tiempo de ejecución y por ende corría más lento que un programa normalmente compilado, a pesar de la argumentación dada por Microsoft de que Visual Basic era una herramienta apropiada para el desarrollo rápido de aplicaciones y que realmente era un lenguaje de programación legítimo [20]. Dicho lenguaje evolucionó de la arquitectura de los 16 bits (versión 3.0) a la de 32 (versión 4.0) para lograr la compatibilidad de la versión de sistema operativo de entonces (Windows 95), pero a partir de la versión 5.0 solo permitía el desarrollo de aplicaciones de 32 bits, aunque esto no originó dificultades en la migración de las aplicaciones escritas en la arquitectura anterior [20]. La versión 5.0

además incluyó la posibilidad de crear *controles ActiveX*¸ para poder realizar la distribución y la reutilización de la lógica y de las funcionalidades gráficas, algo que hasta entonces solo se podía hacer en los lenguajes de bajo nivel; dicha reutilización, que además tenía la facilidad de ser gráfica, permitió un gran impulso del lenguaje y una estandarización en el mercado [21]. La versión 6 del lenguaje incluyó algunas nuevas palabras, así como un estándar de conexión llamado *Microsoft Activex Data Object* (ADO), por medio del cual se puede realizar una conexión a un origen de datos de manera sencilla, y por medio de ella conectarse y desconectarse al origen de datos especificado, llamar a procedimientos almacenados, etc. [21].

La tendencia de agregar nuevas funcionalidades a las versiones posteriores y mantener la compatibilidad con las versiones anteriores (incluyendo el conocimiento adquirido de los desarrolladores) originó inconsistencias, lo que obligaba al programador a realizar trucos para poder lidiar con las mismas. La avalancha de la tecnología Web de finales del siglo XX y principios del XXI hizo que Microsoft planteara nuevas técnicas y lenguajes (tales como Visual InterDev), con la desventaja de obligar al desarrollador a aprender un nuevo entorno de desarrollo y emplear solamente un conjunto de elementos del lenguaje. Las soluciones dadas por Microsoft eran más un parche que soluciones reales, y esto obligó a un replanteamiento del modelo de arquitectura distribuida, que pudiera integrarse a las nuevas tecnologías y además que resolviera muchos de los problemas presentados por las restricciones de la arquitectura existente. El cambio presentado originó riesgos, como el cambio de sintaxis y en sí del paradigma de programación en el lenguaje [21].

Microsoft estableció en el sistema operativo Windows un estándar de comunicación entre objetos llamado *Component Object Model* (COM) para que una aplicación o biblioteca pudiera utilizar otra sin preocuparse de su ruta física ni del lenguaje en el cual fue implementada; de este modo, el componente debía ofrecer un conjunto de características y propiedades, definiéndolos en el archivo de registro, en un proceso denominado *Registro de Biblioteca COM*. COM es un contrato entre una aplicación cliente y otra servidora de intercomunicación de objetos, de tal manera que la mayoría de aplicaciones pueden compartir bibliotecas de otras aplicaciones o del sistema operativo de manera natural [21].

- **Problemas del IDE Visual Basic 6.0.** El marco de trabajo COM que planteó Microsoft en aquella época tenía ventajas, aunque realmente era muy frágil, ya que obligaba al programador a preocuparse de varios asuntos, tales como la gestión de memoria, pudiendo tomar decisiones equivocadas sobre cuándo debe o no liberar una referencia de memoria de un objeto creado; además podía el programador de caer en el error de realizar llamadas circulares (X apunta a Y e Y a X), generando un caos en la aplicación desarrollada; a esto

se le suma que pueden existir cientos de aplicaciones y bibliotecas en el sistema operativo que emplean el modelo COM para comunicarse entre sí, lo cual puede generar un panorama caótico [21].

Quizá el peor problema que este modelo tenía es el denominado "Infierno de las DLL", refiriéndose a un conjunto de problemas comunes causados por la utilización de bibliotecas compartidas; al existir una separación entre la definición de la interfaz de los componentes instalados (en el archivo de registro) y los archivos físicos que contienen la implementación; si bien conceptualmente es correcta esta separación, en la práctica había dificultades, ya que la actualización y desinstalación se volvían difíciles al coordinar ambos recursos. Además de esto, la distribución de una aplicación se requiere incluir todas las librerías que la misma requiere para su funcionamiento correcto; si alguna librería cambia de versión (y peor aún, si esta librería pertenece a Windows) puede originar inconsistencias o fallos inesperados al ejecutar la aplicación, ya que no se puede garantizar que la nueva versión realice las mismas cosas de igual forma [21].

A estos puntos se le puede agregar que la flexibilidad que el lenguaje ofrece puede originar malas prácticas de programación: "*On error resume next*" permite que el flujo del código siga a pesar de encontrar un error; *"Goto*", sentencia heredada del BASIC clásico, que permite saltos del flujo de código sin importar el punto en el que se encuentre; la declaración implícita de variables en un archivo (y para evitar esto requiere de escribir la instrucción "*Option Explicit*"), la declaración de variables sin ser fuertemente tipeadas (a diferencia de lenguajes como C# y Java), entre otras "libertades" que dicho lenguaje permite.

Otro problema crítico con el IDE en cuestión es el soporte por parte de Microsoft; dicha empresa retiró el soporte estándar para Visual Basic en sus ediciones Enterprise y Standard el 31 de marzo de 2005, y el soporte extendido terminó el 8 de abril de 2008 [22].

- **La tendencia a futuro del IDE Visual Basic 6.0 – Su supervivencia.** A pesar de la expiración del soporte para VB6, Microsoft ha garantizado el funcionamiento de las aplicaciones desarrolladas en esta herramienta durante el ciclo de vida de Windows Vista, Windows Server 2008 (incluyendo R2), Windows 7 y el recientemente lanzado Windows 8, tanto para arquitecturas de 32 como de 64 bits [23]; aunque varias librerías que durante años se usaron en Visual Basic clásico ya no son soportadas en los nuevos sistemas operativos, la gran mayoría de ellas tienen garantizado su funcionamiento por parte de Microsoft (según listado expuesto por dicha empresa en [23]), y para arquitecturas de 64 bits las aplicaciones VB son soportadas en el ambiente de emulación WOW (Windows On Windows), puesto que el ambiente nativo del IDE de VB6 es de 32 bits y requiere de dicho ambiente de emulación [23]. Por tal motivo, se puede afirmar

que una aplicación VB6 podría tener hasta 25 años de vida, si fue realizada en 1998 (porque el soporte extendido de Windows 8 termina en enero de 2023); es paradójico e interesante además que una aplicación desarrollada bajo el Framework .NET en su versión 1.0 (que originalmente era el llamado a reemplazar al IDE VB6 y se liberó en 2002, es decir, 4 años después) ya sea incompatible con el sistema operativo Windows 7 [24].

Por otro lado, según K.E. Peterson, autoridad en este tipo de lenguajes, afirma que VB6 sigue teniendo vigencia porque aún hay empresas de negocio y gubernamentales que tienen aplicaciones realizadas en este entorno de desarrollo y para ellas no es fácil realizar migraciones o cambios totales, además de la existencia de una comunidad grande que fue renuente a cambiarse a la tecnología .NET; para él, "tiene que haber una buena razón para migrar el código VB Clásico. No hay nada peor que tomar nuevos lenguajes y comenzar nuevos proyectos con ellos. Pero rescribir código funcional, de manera no recreativa, no es rentable" [25]; por tal motivo, Peterson llama a VB6 como el "COBOL de los 2020s" [26]. M. Desmond en su artículo "Old Soldiers Never Die" afirma, con base en las declaraciones de Peterson, que los desarrolladores optan por mantener el código existente y extender su funcionalidad usando herramientas contemporáneas y lenguajes, y aunque es una solución imperfecta, balancea el costo contra la estabilidad [25]; D. Platt (experto en tecnologías .NET y nombrado por Microsoft como "Leyenda del Software" en 2002) afirma que la supervivencia de VB6 se debe a que una gran comunidad de desarrolladores nunca aceptaron el cambio de paradigma implantado por Microsoft al pasar de VB6 a .NET, y que dicha comunidad continúa trabajando en ese IDE por la facilidad de aprendizaje y el consecuente desarrollo rápido de aplicaciones en programadores de menor experiencia, y que a pesar de la carencia de características orientadas a objetos como el polimorfismo o la sobrecarga de operadores, estos programadores no lo lamentan [24]; este autor ve el potencial que hay en los programadores con gran experiencia, porque todavía hay demanda en el desarrollo de componentes COM [25], además de la larga vida que aún tiene VB6 y apuesta a que Microsoft seguirá soportando VB6 en los futuros sistemas operativos Windows 9 y 10 [24].

Por otra parte, las últimas mediciones de popularidad realizadas por la empresa TIOBE [27] indican que aunque hay lenguajes que cada vez toman más fuerza debido a las últimas tendencias tecnológicas como los móviles (por ejemplo, Objective – C), VB6 no pierde vigencia. Cabe resaltar que el IDE en su popularidad está por encima de Python, Perl, JavaScript, Visual Basic .NET, entre otros. En la Tabla 3.2 se observa la clasificación de los 12 primeros lenguajes a enero de 2013, con su variación respectiva con respecto a enero de 2012.

POSICIÓN ENERO 2013	LENGUAJE DE PROGRAMACIÓN	DELTA ENERO 2012
1	C	+0,89%
2	Java	-0,05%
3	Objective – C	+3,37%
4	C++	+1,09%
5	C#	-2,57%
6	PHP	-0,16%
7	(Visual) Basic	+0,23%
8	Python	+0,96%
9	Perl	-0,50%
10	JavaScript	-0,34%
11	Ruby	+0,34%
12	Visual Basic.NET	+0,56%

Tabla 3.2. Clasificación de Entornos de Desarrollo de acuerdo con su popularidad. Fuente [27]

- **La convivencia de Visual Basic 6.0 con otras tecnologías.** De acuerdo con lo expuesto anteriormente, en una organización pueden coexistir aplicaciones con las dos tecnologías de Microsoft (COM y .NET), y con el respaldo de soporte a las diferentes versiones de sistemas operativos Windows, se espera que esta convivencia perdure por varios años.

Según S. Swigart [28], existen básicamente tres estrategias que se pueden tomar con respecto a una aplicación VB6:

- **Mantenerla como está:** No usar .NET en absoluto; esta estrategia tiene la ventaja de su mantenimiento, el cual es sencillo, porque los desarrolladores conocen bien el lenguaje y pueden realizar cambios o corrección de errores con rapidez. La desventaja es que VB6 per se no contiene nuevas funcionalidades, y desarrollar dichas funcionalidades deseadas requiere de un gran esfuerzo y resultaría muy costoso; se considera además el "punto de fuga", es decir, la migración paulatina de desarrolladores de VB6 a .NET u otras tecnologías.

- **Migración a Visual Basic .NET:** Supone tomar toda la aplicación VB6 y migrarla (ya sea con herramientas o asistida por desarrolladores) a la tecnología .NET. Como ventajas se cuentan la garantía de incorporar las últimas innovaciones y tecnologías (servicios Web, actualización automática de aplicaciones, funcionalidades en Internet), además de la productividad que estas versiones tienen comparadas con VB6.

Sin embargo, se deben considerar algunos riesgos; el primero, la migración a .NET no es fácil, más aún si la aplicación es crítica y los desarrolladores no tienen experiencia en .NET, haciendo que la migración no se realice correctamente; el segundo riesgo es el tiempo, ya que se requiere que la aplicación .NET haga lo mismo que hacía VB6, conllevando costos. A estos riesgos se puede adicionar que la migración de código no garantiza una migración a una tecnología de punta, por ejemplo, un método migrado a .NET que accedía a datos en VB6 por medio del Activex Data Object (ADO) seguiría utilizando dicha tecnología obsoleta, cuando hay propuestas más robustas y con menos código que vienen en .NET como LINQ y Entity Framework.

- **Visual Basic Fusion:** Es el acto de fusionar las funcionalidades de .NET desde una aplicación VB6. Como ventajas tiene el acceso a la biblioteca de clases completa de .NET Framework, llamar a servicios web o usar bibliotecas .NET de terceros o de código abierto desde las aplicaciones VB6; esto implica un bajo riesgo porque el código VB6 continúa funcionando como está y no se requiere la reescritura de código, y el rendimiento a corto plazo es alto. Sin embargo, este enfoque tiene desventajas, tales como la continuación del uso del IDE VB6, y como tal no hay tanta productividad; se requiere además mantener ambos entornos abiertos para el proceso de depuración, y requiere desarrollo de contenedores de las clases .NET para que puedan exponerse como objetos COM [28].

- **La estrategia a tomar en el Sistema SION.** Se optó por utilizar la primera estrategia comentada en el numeral anterior (dejar la aplicación en VB6). A pesar de los inconvenientes que un IDE como VB6 tiene para el desarrollo de software, se tomaron en cuenta las particularidades existentes en el Sistema SION. El primero de ellos es la magnitud en LOC que todos los módulos pueden tener; un ejemplo es el módulo en estudio, en solamente sus componentes de la capa de Datos y de la Lógica de Negocios en su estado inicial mide 16.750 LOC, y como se mencionó en el apartado anterior, la migración no siempre es exitosa para proyectos de gran magnitud y esto requiere tiempo, esfuerzo y costos adicionales; además la experiencia en VB6 de los desarrolladores del Sistema es amplia, comparada con la utilización de la tecnología .NET. Por otra parte, el soporte que Microsoft da a los sistemas operativos Windows en sus últimas versiones garantizan la convivencia de las aplicaciones COM hasta el 2023, lo que permitiría que los componentes y aplicaciones desarrolladas en esta tecnología puedan continuar con vida en los próximos años. Se adiciona a este panorama que la comunidad de desarrollo en VB6 aún es importante (como lo demostrado en los índices de popularidad medidos por la empresa TIOBE) y se encuentran en la Red varios recursos para este entorno. Todo esto permitirá la continuidad en las operaciones de la Institución y la disminución del riesgo de cambios drásticos en el comportamiento del sistema, ajustándose a la capacidad con que cuenta la Institución para llevar a cabo estas tareas.

Una vez concluido este proceso, la gerencia del Sistema puede optar por estrategias de integración de sistemas con nuevas tecnologías, especialmente por medio de la Arquitectura Orientada a Servicios.

3.2 SELECCIÓN DEL MÓDULO PARA EL EXPERIMENTO

El Módulo seleccionado para el estudio fue el de Admisiones y Registro Académico, debido a su importancia evidente dentro de una institución educativa de tipo universitario. Por medio de éste se gestionan los procesos más importantes de la organización, además de la gestión de la información de los clientes más importantes, a saber, los estudiantes. Para inicios del primer semestre de 2013, el Módulo tenía acceso al registro histórico de las siguientes variables expuestas en la Tabla 3.3:

VARIABLE	CANTIDAD DE REGISTROS
Estudiantes	11.369
Docentes	377
Programas académicos	24
Cursos	1.811
Instancias de cursos	5.337
Registros de matrículas	29.152
Registros históricos de notas de estudiantes	221.465

Tabla 3.3. Cantidad de registros por variable del Módulo SION – Admisiones, al primer semestre de 2013. Fuente: Departamento de Sistemas e Informática (DSI) – Corporación Universitaria Adventista

Esta información abarca tanto la población actual como la que ha sido registrada en años anteriores. Aunque la información es considerable en su magnitud, se aprovecha la capacidad que tiene el motor de base de datos que respalda al Sistema (MS – SQL Server, versión 2008 R2), el cual está diseñado para proporcionar acceso controlado y procesamiento de transacciones rápido para cumplir con los requisitos de las aplicaciones consumidoras de datos más exigentes de una empresa [29], además de aprovechar la amplia capacidad de almacenamiento que posee (hasta 524 PB en la versión utilizada en el Sistema SION, es decir, la versión Standard) [30].

El módulo SION - Admisiones está dividido en 7 secciones, y en total cuenta con 217 funciones definidas; véase la Tabla 3.4 en la que se visualizan el total de funciones por sección y algunos ejemplos de funciones por cada sección.

N°	SECCIÓN	EJEMPLOS DE FUNCIONES	CANTIDAD
1	Gestión de Información de Estudiantes	Crear información de estudiante, gestionar información de registro histórico de notas de estudiante, consultar listado general de estudiantes	29
2	Gestión de Información de Docentes	Consultar información de docentes, consultar carga académica de docente por período académico	11
3	Gestión de Información básica	Crear Programa Académico, crear currículo, listar cursos, crear instancias de curso	23
4	Gestión de proceso de matrículas	Consultar matrícula de estudiante, imprimir matrícula de estudiante, matricular estudiante, listar matrículas bloqueadas	28
5	Gestión de proceso de prematrículas	Consultar prematrícula de estudiante, prematricular curso a estudiante, generar prematrícula a todos los estudiantes	12
6	Gestión de otros procesos de Admisiones	Crear período académico, asignar niveles de programas académicos por período académico.	23
7	Gestión de reportes	Imprimir estadísticas académicas, imprimir estadísticas de bienestar universitario	91
TOTAL			**217**

Tabla 3.4. Funciones del Módulo SION - Admisiones. Fuente: Departamento de Sistemas e Informática (DSI) - Corporación Universitaria Adventista

3.3 DEFINICIÓN DEL MODELO DE EVALUACIÓN

Como se mencionó anteriormente, la propuesta en este trabajo se basó en el modelo de evaluación de un sistema legado propuesto por A. De Lucía, A.R. Fasolino, E. Pompella [6], quienes para cada dimensión (El valor de negocio y el valor técnico) determinaron una serie de variables y unas métricas respectivas. En este trabajo se agregaron dos variables adicionales a la propuesta original en la dimensión técnica, el desempeño y la seguridad, las cuales la norma ISO 25010 contempla para evaluar la calidad de un sistema [31] y de esta manera abarcar una medición de calidad más amplia y ajustada a dicha norma. Véase la Tabla 3.5 en el que se despliega la propuesta de evaluación para el Sistema.

DIMENSIÓN	SUBCARACTERÍSTICA	ATRIBUTOS
Valor de Negocio	Valor económico	Valor de mercado
	Valor de información	Valor de información
	Utilidad	Cobertura de funcionalidad de negocio, Frecuencia actual de uso, satisfacción de usuario
	Especialización	Cobertura de funcionalidad altamente especializada, cobertura de funcionalidad genérica desarrollada
Valor Técnico	Mantenibilidad	Longitud de código, proporción de código muerto, complejidad ciclomática, instrucciones SQL embebidas en la aplicación.
	Descomponibilidad	Modularidad – acoplamiento, modularidad – cohesión.
	Deterioro	Proporción de trabajo acumulado, proporción de la tasa de defectos, incremento del tiempo de respuesta, esfuerzo de soporte en tiempo crítico
	Obsolescencia	Adaptabilidad en ambientes software, adaptabilidad en ambientes hardware, soporte de IDE, soporte motor de base de datos.
	Desempeño	Comportamiento en el tiempo, consumo de recursos – CPU, consumo de recursos – Input / Output
	Seguridad	Resistencia al acceso – cifrado en credenciales de usuario, resistencia al acceso – ocultamiento de conexión al servidor, resistencia al acceso – utilización de usuario privilegiado

Tabla 3.5. Modelo de evaluación propuesto para el Módulo SION – Admisiones. Fuente: Elaboración propia, adaptado de [6].

4. DEFINICIÓN DEL ESTUDIO EXPERIMENTAL

4.1 PREGUNTAS DE INVESTIGACIÓN Y MÉTRICAS ASOCIADAS

A partir del modelo de evaluación presentado en el capítulo anterior, se plantearon 4 preguntas de investigación, a partir de las cuales se midieron los valores de negocio y técnico del módulo seleccionado, tanto antes como después del proceso de refactorización. Las preguntas son las siguientes:

- ¿Cuál es el valor de negocio del Módulo de Admisiones del Sistema SION?
- ¿Cuál es el valor técnico del Módulo de Admisiones del Sistema SION, antes de la refactorización?
- ¿De qué manera se va a llevar a cabo el proceso de refactorización?
- ¿Qué impacto tiene la refactorización en el valor técnico del producto?

En la Tabla 4.1 se resume las métricas asociadas para cada dimensión y subcaracterística.

DIMENSIÓN	SUBCARACTERÍSTICA	ATRIBUTOS	MÉTRICAS
Valor de Negocio	Valor económico	Valor de mercado	**PDM**: Porcentaje de Diferencia de Mercado.
	Valor de información	Valor de información	**VI**: Valor de Información
	Utilidad	Cobertura de funcionalidad de negocio	**PC**: Porcentaje de funciones implementadas correcta y completamente en relación con las funciones especificadas para el módulo
		Frecuencia actual de uso	**PTA**: Porcentaje de tiempo en el que el sistema está realmente apto para ser utilizado, en relación con el tiempo total en el que debe estar listo para ser utilizado **PUFAU**: Percepción del Usuario de la Frecuencia actual de uso.
		satisfacción de usuario	**SU**: Satisfacción de Usuario
	Especialización	Cobertura de funcionalidad altamente especializada	**PFE**: Porcentaje de funciones altamente especializadas terminadas completamente, en relación con el total de funciones especificadas para el módulo. **UPFE**: Percepción de usuario de la cobertura de funcionalidad altamente especializada
		Cobertura de funcionalidad genérica desarrollada	**PFG**: Porcentaje de funciones genéricas terminadas completamente, en relación con el total de funciones especificadas para el módulo. **UPFG**: Percepción de usuario de la cobertura de funcionalidad genérica.
Valor Técnico	Mantenibilidad	Longitud de código	**TLOC**: Total de Líneas de Código
		proporción de código muerto	**PCM**: Proporción de Código Muerto
		complejidad ciclomática	**PCC**: proporción de procedimientos

DIMENSIÓN	SUBCARACTERÍSTICA	ATRIBUTOS	MÉTRICAS
			Visual Basic con complejidad ciclomática categorizados como simples y bien estructurados, en los componentes de capa de datos, lógica de negocio y transversal **PRCC**: Promedio de promedios de complejidad ciclomática en los procedimientos Visual Basic de los componentes de capa de datos, lógica de negocio y transversal
		instrucciones SQL embebidas en la aplicación	**TSQL**: Total de instrucciones SQL embebidas en los procedimientos Visual Basic en los componentes de capa de datos, lógica de negocio, transversal e interfaz de usuario.
	Descomponibilidad	Modularidad – acoplamiento	**PA:** Promedio de acoplamiento entre componentes.
		Modularidad – cohesión	**PC**: Porcentaje de métodos asignados correctamente
	Deterioro	Proporción de trabajo acumulado	**PTA**: Porcentaje de trabajo acumulado en un lapso de tiempo de tiempo determinado
		Proporción de la tasa de defectos	**PD**: Porcentaje de defectos en un lapso de tiempo de tiempo determinado para el módulo en estudio.
		Incremento del tiempo de respuesta	**ETC**: Proporción Horas – Hombre requeridos en procesos críticos de operación para el módulo en estudio.
		Esfuerzo de soporte en tiempo crítico	**PITR**: Proporción de incremento de tiempo de respuesta entre dos solicitudes contiguas.
	Obsolescencia	Adaptabilidad en ambientes software	**PAS**: Porcentaje de funciones del módulo en estudio que operan correctamente en el sistema operativo Windows de una versión determinada.
		adaptabilidad en ambientes hardware	**PAH**: Porcentaje de funciones del módulo en estudio que operan correctamente en un computador con hardware determinado.
		soporte de IDE	**SIDE**: Valoración de acuerdo con la vigencia de soporte al IDE por parte de su casa matriz.
		soporte motor de base de datos	**SMBD**: Valoración de acuerdo con la vigencia de soporte al motor de base de datos por parte de su casa matriz.
	Desempeño	Comportamiento en el tiempo	**PTR**: Promedio de tiempo de respuesta medido en milisegundos.
		Consumo de recursos – CPU	**PCPU**: Promedio de consumo de CPU del servidor de base de datos, medido en milisegundos.
		Consumo de recursos– Input / Output	**PIO**: Promedio de entradas y salidas en el servidor de base de datos, medido en milisegundos.
	Seguridad	Resistencia al acceso – cifrado en credenciales de usuario,	**PUC**: Porcentaje de usuarios que utilizan el módulo con cifrado de credenciales de usuario.
		Resistencia al accceso – ocultamiento	**PUOC**: Porcentaje de usuarios que

DIMENSIÓN	SUBCARACTERÍSTICA	ATRIBUTOS	MÉTRICAS
		de conexión al servidor	utilizan el módulo ocultamiento de la conexión al servidor.
		Resistencia al acceso – utilización de usuario privilegiado	**PUSA**: Porcentaje de usuarios que utilizan el módulo ocultamiento de la conexión al servidor.

Tabla 4.1. Métricas asocidas para atributos, subcaracterísticas y dimensiones. Fuente: Elaboración propia.

A continuación se presentan las métricas asociadas a cada pregunta planteada.

4.1.1 Métricas para determinar el valor de negocio.

Las métricas asociadas a esta pregunta son las siguientes:

- **Valor del mercado:** El valor de mercado de la aplicación en estudio hace referencia a su valor económico comparado con las demás aplicaciones que abarcan el mismo ámbito y se encuentran en el mercado. Se hizo su medición a través de la métrica Porcentaje de diferencia del mercado (PDM), la cual es una métrica propuesta en este trabajo; para dicha métrica se definió la siguiente ecuación:

$$PDM = (VAC - VEM) *100/ VEM$$

Dónde:

PDM: Porcentaje de diferencia del mercado
VAC: Valor de la aplicación de la competencia
VEM: Valor Estimado del Módulo.

El valor ideal a obtener en esta medición es 100.

- **Valor de información (VI):** Por medio de esta métrica se trata de determinar cuán valiosa es para los usuarios la información que se gestiona en el módulo en estudio. Este valor se midió por medio de una encuesta a los usuarios del módulo, con 4 preguntas (específicamente de la 2 a la 5) y valorada por medio de la escala de Likert (valores de 1 a 5). El promedio de estos valores es el valor definitivo para toda la aplicación en esta métrica. El valor ideal de esta métrica es 5 (en la escala de 1 a 5).

- **Cobertura de funcionalidad de negocio**: Porcentaje de funciones implementadas correcta y completamente en relación con las funciones especificadas para el módulo. Esta métrica está basada en la

métrica de porcentaje de requerimientos cumplidos para medir la cobertura y en general para la apropiabilidad del producto [32]. Esta métrica se definió a través de la siguiente ecuación:

$$PC = TFIC * 100 / TF$$

Dónde:

PC: Porcentaje de cobertura de funcionalidad de negocio.
TFIC: Total de funciones implementadas correcta y completamente.
TF: Total de funciones especificadas.

El valor ideal a obtener en esta medición es 100.

- **Frecuencia actual de uso:** Esta métrica se refiere al porcentaje de tiempo en el que el sistema está realmente apto para ser utilizado, en relación con el tiempo total en el que debe estar listo para ser utilizado [32]. La fórmula asignada a esta métrica es la siguiente:

$$PTA = (TP - TF) * 100 / TP$$

Dónde:

PTA: Porcentaje de tiempo apto del módulo
TP: Tiempo pactado en un lapso específico en el que el sistema debe poder usarse.
TF: Tiempo en el que el sistema estuvo fuera de servicio en un lapso específico.

El valor ideal es 100.

Además de ello se determinó la percepción de los usuarios sobre la disponibilidad del sistema para sus labores, por medio de la encuesta, con la pregunta 1; dicha métrica se definió como "Percepción de usuario de la frecuencia actual de uso" (PUFAU), cuyo valor ideal es 5, en la escala de 1 a 5.

- **Satisfacción de usuario (SU):** En un sistema de información, la satisfacción de usuario se puede definir como la valoración de las salidas que el sistema le pueda otorgar, midiendo en estos la cantidad de los mismos y la efectividad de los resultados que obtiene de los servicios, condicionando dicha satisfacción tanto

la disponibilidad como la accesibilidad del sistema a los usuarios [33]. Para medir dicha satisfacción, se realizó a través de la encuesta a los usuarios del módulo, con las preguntas 8 a 21; la valoración de esta métrica se hizo por medio de la escala de Likert (valores de 1 a 5), promediando los resultados para cada usuario encuestado y finalmente promediando dichos promedios, para definir el valor respectivo. El valor ideal de esta métrica es 5.

- **Cobertura de funcionalidad altamente especializada:** Porcentaje de funciones altamente especializadas terminadas completamente, en relación con el total de funciones especificadas para el módulo. Esta métrica es propia de este trabajo, basada en métrica de porcentaje de requerimientos cumplidos para medir la cobertura y en general para la apropiabilidad del producto [32]. La fórmula utilizada para esta métrica es la siguiente:

$$PFE = FEC * 100 / TF$$

Dónde:

PFE: Porcentaje de funciones altamente especializadas.
FEC: Funciones especializadas totalmente completas
TF: Total de funciones especificadas para el módulo.

El valor ideal para esta métrica es de 100.

También se determinó la percepción de los usuarios sobre la cobertura de funcionalidad altamente especializada por medio de la encuesta, con la pregunta 7; dicha métrica se definió como "Percepción de usuario de la cobertura de funcionalidad altamente especializada" (UPFE), cuyo valor ideal es 5, en la escala de 1 a 5.

- **Cobertura de funcionalidad genérica:** Porcentaje de funciones genéricas terminadas completamente, en relación con el total de funciones especificadas para el módulo. Al igual que la anterior, la métrica es propia de este trabajo y tiene la misma base [32] . La fórmula definida en esta métrica fue la siguiente:

$$PFG = FGC * 100 / TF$$

Dónde:

PFG: Porcentaje de funciones genéricas.
FGC: Funciones genéricas totalmente completas
TF: Total de funciones especificadas para el módulo.

El valor ideal para esta métrica es de 100.

Adicionalmente se determinó la percepción de los usuarios sobre la cobertura de funcionalidad genérica por medio de la encuesta, con la pregunta 6; dicha métrica se definió como "Percepción de usuario de la cobertura de funcionalidad genérica" (UPFG), cuyo valor ideal es 5, en la escala de 1 a 5.

4.1.2 Métricas para determinar el valor técnico del Módulo de Admisiones

Las métricas asociadas a esta pregunta de investigación fueron las siguientes:

- **Longitud de Código (LOC):** La medición por líneas de código se define como una medida directa del producto software [34], relacionada con el tamaño del producto [7]. Para este trabajo, se definió una métrica propia, en donde se hace el conteo de líneas de código lógicas desarrolladas en los componentes de datos, de lógica de negocio y transversal para el módulo en estudio, lo cual se expresa a través de la siguiente fórmula:

$$TLOC = TLOCCD + TLOCLN + TLOCTR$$

Dónde:

TLOC: Total líneas de código Visual Basic en componentes de capa de datos, lógica de negocio y transversal.
TLOCCD: Total líneas de código Visual Basic en componentes de capa de datos.
TLOCLN: Total líneas de código Visual Basic en componentes de capa de lógica de negocio.
TLOCTR: Total líneas de código Visual Basic en componentes capa transversal.

El valor ideal de esta métrica es 50% o más de mejora en LOC.

- **Proporción de código muerto:** El código muerto se refiere a líneas de código cuyos resultados nunca se usan, el cual puede ser eliminado sin afectar el comportamiento del programa [35]. En este trabajo se propone una métrica propia, donde se toma el porcentaje de código en VB6 que es redundante, inaccesible o

que no se usa, en los componentes de capa de datos, lógica de negocio y transversal. La fórmula para esta métrica se define así:

$$PCM = TLOCM * 100 / TLOC$$

Dónde:

PCM: Porcentaje de código muerto.
TLOCM: Total Líneas de código muerto
TLOC: Total de líneas de código.

El valor ideal de esta métrica es 0.

- **Complejidad Ciclomática:** La complejidad ciclomática es una métrica que proporciona una medición cuantitativa de la complejidad lógica de un programa [34], fue definida en 1976 por Thomas McCabe como el número de caminos linealmente independientes y por lo tanto, el número mínimo de caminos que debería ser probado [36]. Una alta complejidad ciclomática denota un procedimiento complejo que es difícil de entender, probar y mantener; se debe tener en cuenta además que hay un riesgo entre dicha complejidad en un procedimiento [37]; la relación entre complejidad ciclomática, el tipo de procedimiento y riesgo se visualizan en la Tabla 4.2.

CC	TIPO DE PROCEDIMIENTO	RIESGO
1 – 4	Un procedimiento simple	Bajo
5 – 10	Un procedimiento bien estructurado y estable	Bajo
11 – 20	Un procedimiento más complejo	Moderado
21 – 50	Un procedimiento complejo, alarmante	Moderado
> 50	Un procedimiento propenso a errores, extremadamente dificultoso, sin poder probarse	Muy alto

Tabla 4.2. Relación entre complejidad ciclomática, el tipo de procedimiento y el riesgo. Fuente [37]

Originalmente el valor apropiado para la complejidad ciclomática no debe exceder más allá de 10 [38], sin embargo, otros valores se han sugerido, tales como 15 o 20, pero más allá de este último valor, debe considerarse alarmante y tomar medidas al respecto, tales como dividir el procedimiento en varios pequeños procedimientos [37].

Para esta medición, se crearon dos métricas: La proporción de procedimientos Visual Basic con complejidad ciclomática categorizados como simples y bien estructurados, en los componentes de capa de datos, lógica de negocio y transversal; la segunda métrica es el promedio de promedios de complejidad ciclomática en dichos procedimientos en los mismos componentes. Para la primera métrica se define la siguiente fórmula:

$$PCC = (TPS + TPBE)*100/ TP$$

Dónde:

PCC: Porcentaje de procedimientos Visual Basic con complejidad ciclomática categorizados como simples y bien estructurados.
TPS: Total procedimientos con complejidad ciclomática simple.
TPBE: Total procedimientos con complejidad ciclomática bien estructurada.
TP: Total de procedimientos Visual Basic en los componentes de capa de datos, lógica de negocio y transversal.

El valor ideal de esta métrica es 100%.

Para la segunda métrica, se define la siguiente fórmula:

$$PRCC = (\Sigma PCCD + \Sigma PCCLN + \Sigma PCCT) / N$$

Dónde:

PRCC: Promedio de promedios de complejidad ciclomática en los procedimientos Visual Basic de los componentes de capa de datos, lógica de negocio y transversal
ΣPCCD: Sumatoria de promedios de complejidad ciclomática en los procedimientos Visual Basic de los componentes de capa de datos
ΣPCCLN: Sumatoria de promedios de complejidad ciclomática en los procedimientos Visual Basic de los componentes de lógica de negocio y transversal.
ΣPCCT: Sumatoria de promedios de complejidad ciclomática en los procedimientos Visual Basic de los componentes de capa transversal.
N: Total de promedios de complejidad ciclomática en los componentes de capa de datos, lógica de negocio y transversal.

El valor ideal de esta métrica es 10, teniendo en cuenta el concepto de McCabe explicado anteriormente.

- **Instrucciones SQL embebidas en los procedimientos Visual Basic:** Una buena práctica aconsejada desde hace varios años es que el front – end de la ejecución nunca ejecute instrucciones SQL directamente, sino que se haga a través de procedimientos almacenados que puedan ser accedidos desde la aplicación; esta buena práctica conserva limpio el acceso de datos, además de darle consistencia a través de todos los módulos y permite desacoplar la lógica de negocios [39]. Para determinar la calidad de la aplicación en este aspecto, se creó la métrica "Cantidad de instrucciones SQL embebidas en los procedimientos Visual Basic en los componentes de datos, lógica de negocio, transversal e interfaz de usuario del Módulo", cuya fórmula es:

$$TSQL = \Sigma SQLCD + \Sigma SQLLN + \Sigma SQLT + \Sigma SQLGUI$$

Dónde:

TSQL: Total de instrucciones SQL embebidas en los procedimientos Visual Basic en los componentes de capa de datos, lógica de negocio, transversal e interfaz de usuario.
ΣSQLCD: Total instrucciones SQL embebidas en los procedimientos Visual Basic en los componentes de capa de datos.
ΣSQLLN: Total instrucciones SQL embebidas en los procedimientos Visual Basic en los componentes de lógica de negocios.
ΣSQLT: Total instrucciones SQL embebidas en los procedimientos Visual Basic en los componentes de capa de datos.
ΣSQLGUI: Total instrucciones SQL embebidas en los procedimientos Visual Basic en la interfaz gráfica de usuario.

El valor ideal de esta métrica es 0.

- **Acoplamiento:** El acoplamiento mide la interconexión entre módulos de una aplicación software; depende de la complejidad de interconexión entre módulos, el punto en donde se realiza la entrada o la referencia a un módulo, y los datos que pasan a la interfaz [34].

En este trabajo se realizó su medición a través de la métrica "Promedio de acoplamiento entre componentes", para ello, primero se determinaron las llamadas que cada componente hace a otros (Fan – In) y qué componentes llaman a uno en particular (Fan – Out), y se adaptó la métrica de Dhama para el acoplamiento para determinarlo en cada componente [34]; dicha adaptación se expresa de la siguiente manera:

RELACIÓN = 1/ (TFIN + TFOUT)

Y finalmente determinar el promedio de estas relaciones:

PA = PROMEDIO (RELACIÓN)

Dónde:

PA: Promedio de acoplamiento entre componentes.
TFIN: Total de componentes que llaman al componente en estudio.
TFOUT: Total de componentes que llama el componente en estudio.

Un valor cercano a 1 indica un menor acoplamiento.

- **Cohesión:** Este atributo es una extensión del concepto de ocultamiento de información; un módulo que es cohesivo realiza una sola tarea dentro de un procedimiento software, es decir, requiere poca interacción con los procedimientos que se realizan en otras partes de la aplicación [34]. Para realizar dicha medición en este trabajo, se creó la métrica "Porcentaje de métodos asignados correctamente a las clases correspondientes, en las capas de componente de datos, lógica de negocio y transversal"; esta métrica se expresa por medio de la fórmula:

PC = TAC*100/TP

Dónde:

PC: Porcentaje de métodos asignados correctamente
TAC: Total de procedimientos asignados correctamente.
TP: Total de procedimientos del módulo en las capas de componente de datos, lógica de negocio y transversal.

El valor ideal de esta métrica es 100%.

- **Proporción de trabajo acumulado:** Para determinar las tareas acumuladas del módulo en estudio, se creó la métrica "Porcentaje de tareas asignadas no terminadas del módulo en estudio, durante un lapso de tiempo determinado", expresada por medio de la siguiente fórmula:

$$PTA = TNT*100/TT$$

Dónde:

PTA: Porcentaje de trabajo acumulado en un lapso de tiempo de tiempo determinado.
TNT: Total de tareas no terminadas en el lapso de tiempo determinado, para el módulo en estudio.
TT: Total de tareas reportadas en el lapso de tiempo determinado, para el módulo en estudio.
El valor ideal de esta métrica es 0.

- **Proporción de defectos:** Hay diversas definiciones para los errores, defectos y fallos en un sistema software. Sin embargo, en este contexto se adopta la definición de defecto (bug) a un problema que causa un programa que produce una salida inválida, un bloqueo o una interrupción abrupta del mismo [40], también se conoce la manifestación o demostración de un error. Para medirlo, se determinó la métrica "Porcentaje de tareas determinadas como defectos del módulo con respecto al total de tareas registradas del módulo en estudio, durante un lapso de tiempo determinado", definido por la siguiente fórmula:

$$PD = TD*100/TT$$

Dónde:

PD: Porcentaje de defectos en un lapso de tiempo de tiempo determinado para el módulo en estudio.
TD: Total de defectos reportados en el lapso de tiempo determinado del módulo en estudio.
TT: Total de tareas reportadas en el lapso de tiempo determinado, para el módulo en estudio.

Esta métrica está basada en la denominada "Densidad de Fallas" (DF), para evaluar la madurez de un producto de acuerdo con la norma ISO/IEC 9126 e ISO/IEC 25010 [32]. Su valor ideal es 0.

- **Esfuerzo de soporte en tiempo crítico:** Proporción Horas – Hombre requeridos en procesos críticos de operación del módulo con respecto al total de Horas – Hombre de soporte en el intervalo de tiempo estudio, para el módulo en cuestión. Esta métrica, propuesta específicamente para este trabajo, se definió con la siguiente fórmula:

$$ETC = HH*100/TH$$

Dónde:

ETC: Proporción Horas – Hombre requeridos en procesos críticos de operación para el módulo en estudio.
HH: Total de Horas – Hombre utilizadas en tiempo crítico de operación del módulo en estudio.
TH: Total de horas de soporte del módulo en el lapso de tiempo determinado.

- **Proporción de incremento de tiempo de respuesta:** Porcentaje de diferencias de tiempo de respuesta entre dos solicitudes con valor mayor que cero, para el módulo en cuestión en un lapso de tiempo determinado. La fórmula para esta métrica es la siguiente:

$$PITR = 100 – (TTI*100/TTT)$$

Dónde:

PITR: Proporción de incremento de tiempo de respuesta entre dos solicitudes contiguas.
TTI: Total de diferencia de tiempos de respuesta entre dos tareas ejecutadas que son mayores a cero.
TTT: Total de tareas terminadas y reportadas en el lapso de tiempo determinado, para el módulo en estudio.

Esta métrica es especificada particularmente para este trabajo. Su valor ideal es 0.

- **Adaptabilidad en ambientes software:** Para este atributo se determinó la métrica "Proporción de funciones del módulo en estudio que operan correctamente en el sistema operativo Windows de una versión determinada". Su fórmula se definió de la siguiente manera:

$$PAS = (TFO*100/TF)$$

Dónde:

PAS: Porcentaje de funciones del módulo en estudio que operan correctamente en el sistema operativo Windows de una versión determinada.
TFO: Total de funciones que operan correctamente.
TF: Total de funciones del módulo en estudio.

Esta métrica es una propuesta propia para este trabajo; el atributo está basado en la norma ISO/IEC 9126 e ISO/IEC 25010 que mide la adaptabilidad del producto [32]. Su valor ideal es 100.

- **Adaptabilidad en ambientes hardware:** Para este atributo se determinó la métrica "Proporción de funciones del módulo en estudio que operan correctamente en un computador con hardware determinado". La métrica se definió de la siguiente manera:

$$PAH = TFO*100/TF$$

Dónde:

PAH: Porcentaje de funciones del módulo en estudio que operan correctamente en un computador con hardware determinado.
TFO: Total de funciones que operan correctamente.
TF: Total de funciones del módulo.

Al igual que la métrica anterior, esta métrica es una propuesta propia para este trabajo; el atributo está basado en la norma ISO/IEC 9126 e ISO/IEC 25010 que mide la adaptabilidad del producto [32]. Su valor ideal es 100.

- **Soporte para Entorno Integrado de Desarrollo (IDE):** Valor asignado de acuerdo con la vigencia de soporte al IDE por parte de su casa matriz. Se asigna la nota máxima si lo posee, o la mínima si el soporte ya no es vigente. Esta es una métrica propuesta para este trabajo. La métrica se definió como SIDE. Su valor ideal es 5, en la escala de 1 a 5.

- **Soporte para motor de base de datos:** Valor asignado de acuerdo con la vigencia de soporte al motor de base de datos por parte de su casa matriz. Se asigna la nota máxima si lo posee, o la mínima si el soporte ya no es vigente. Al igual que la métrica anterior, es una propuesta particular de este trabajo. La métrica se definió como SMBD. Su valor ideal es 5, en la escala de 1 a 5.

- **Métricas de Rendimiento:** La medida de rendimiento de un sistema software se define como la cuantificación de la eficiencia y de la efectividad de una acción; un proceso efectivo produce resultados que satisfagan las necesidades de un usuario con una variabilidad de proceso bajo [41]. En los módulos del

Sistema SION el rendimiento depende básicamente de las respuestas que el motor de base de datos pueda dar a una solicitud como una consulta. Por este motivo se desarrollaron algunas métricas que pudieran determinar cuán costosas son estas peticiones y medir las mejoras al realizar el proceso de refactorización.

Solucionar los problemas de rendimiento en una aplicación a veces puede ser una tarea muy dura, porque se requiere determinar qué es lo que se debe buscar, y determinar por qué el sistema reacciona ante una determinada solicitud. En SQL Server (motor de base de datos del Sistema SION) el rendimiento depende de la compilación y recompilación de instrucciones SQL, la ausencia de índices, operaciones con varios subprocesos, el mantenimiento rutinario, y la actividad de extracción, transformación y carga (ETL), la eficiencia del código en los procedimientos almacenados y funciones (En SQL Server el lenguaje de codificación se denomina Transact – SQL), y otros factores que afecten el rendimiento del sistema. Además de esto, pueden existir problemas de hardware. Se recomienda utilizar la vía tanto de la mejora de dispositivos hardware como la base de datos en sí (índices, consultas, planes de mantenimientos, etc.) [42].

- **Promedio de tiempo de respuesta de consultas SQL más costosas:** Este atributo es el que el usuario del sistema se relaciona más, llegando a catalogar al sistema como "bueno o malo" por el tiempo de respuesta a una solicitud dada. La métrica asignada para este atributo es el "Promedio de tiempo de respuesta en el servidor de base de datos, medido en milisegundos, de las 20 consultas SQL más costosas, del módulo en estudio" (PTR) y está basada en la métrica "Tiempo de Respuesta" (TRO) para medir la capacidad para responder a las necesidades en un tiempo de retorno aceptable [32].

La métrica propuesta en este trabajo se define como:

$$PTR = \Sigma TR/20$$

Dónde:

PTR: Promedio de tiempo de respuesta medido en milisegundos.
ΣTR: Tiempo de respuesta de consulta SQL, medido en milisegundos.

El valor ideal es el 100% de mejora, con respecto a la medición inicial.

- **Promedio de consumo de recursos en CPU:** El uso de la CPU del servidor de base de datos en sí no es bueno ni malo, pero es un buen indicador de cuánto consume el sistema al ejecutar una determinada

instrucción [42]. Para este atributo se propuso como métrica el "Promedio de consumo de CPU del servidor de base de datos, medido en milisegundos, al ejecutar las 20 consultas SQL más costosas, del módulo en estudio". (PCPU), definiéndose de la siguiente manera:

$$PCPU = \Sigma CPU/20 \quad (20)$$

Dónde:

PCPU: Promedio de consumo de CPU del servidor de base de datos, medido en milisegundos.
CPU: consumo de CPU del servidor de base de datos, medido en milisegundos.

La métrica es una propuesta propia para este trabajo, pero se basa en las métricas utilizadas para medir el consumo de recursos de un producto software, según las normas ISO/IEC 9126 e ISO/IEC 25010 [32]. El valor ideal es el 100% de mejora, con respecto a la medición inicial.

- **Promedio de entradas y salidas (Input/Output**): Un elevado uso de entradas y salidas puede indicar que haya mecanismos de acceso poco eficaces. SQL Server provee información para determinar cuáles son las bases de datos que leen y escriben la mayoría de páginas lógicas [43]. Por tal motivo, se propuso la métrica "Promedio de entradas y salidas en el servidor de base de datos, medido en milisegundos, al ejecutar las 20 consultas SQL más costosas, del módulo en estudio" (PIO), definiéndose de la siguiente manera:

$$PIO = \Sigma IO/20 \quad (21)$$

Dónde:

PIO: Promedio de entradas y salidas en el servidor de base de datos, medido en milisegundos.
IO: Entradas y salidas en el servidor de base de datos, medidos en milisegundos.

Al igual que la métrica anterior, es una propuesta propia para este trabajo, pero se basa en las métricas utilizadas para medir la capacidad de optimizar el uso de dispositivos I/O, según la norma ISO/IEC 9126 e ISO/IEC 25010 [32]. El valor ideal es el 100% de mejora, con respecto a la medición inicial.

- **Métricas de Seguridad:** De acuerdo con las normas ISO/IEC 9126, se define la seguridad como una subcaracterística de la funcionalidad de un producto software, que indica el grado en que un acceso no autorizado (accidental o deliberado) se prevenga y se permita un acceso autorizado [44]; hace referencia a "la

capacidad del software para proteger los datos y la información, con el fin de que personas no autorizadas nunca puedan ingresar al sistema, leer o modificar los datos, y que a las personas que sí estén autorizadas no se les niegue el acceso al sistema ni a la información que necesiten" [32]. A diferencia de la norma ISO/IEC 9126, el estándar SQuaRE (Software product Quality Requirements and Evaluation) define la seguridad como una característica y no como una subcaracterísitica [31].

Para esta experiencia, se proponen métricas que puedan medir el riesgo de acceso no autorizado tanto a la información propia del módulo como de todas las bases de datos que componen el Sistema; por ello se determinó que el cifrado de credenciales de acceso al módulo por parte del usuario, el ocultamiento de la información de conexión al sistema desde las máquinas cliente y la utilización del inicio de sesión de super administrador (*sa*) fueran la base para las métricas propuestas.

- **Proporción de usuarios con módulo con cifrado de credenciales de usuario:** Porcentaje de usuarios que utilizan el módulo en estudio con algoritmo cifrado. Se define de la siguiente manera:

$$PUC = TUC/TUM \quad (22)$$

Dónde:

PUC: Porcentaje de usuarios que utilizan el módulo con cifrado de credenciales de usuario.
TUC: Total de usuarios que utilizan el módulo con cifrado de credenciales de usuario.
TUM: Total de usuarios que utilizan el módulo en estudio.

- **Proporción de usuarios del módulo con ocultamiento de la conexión al servidor:** Se define esta métrica de la siguiente manera:

$$PUOC = TUOC/TUM \quad (23)$$

Dónde:

PUOC: Porcentaje de usuarios que utilizan el módulo ocultamiento de la conexión al servidor.
TUOC: Total de usuarios que utilizan el módulo con ocultamiento de la conexión al servidor.
TUM: Total de usuarios que utilizan el módulo en estudio.

El valor ideal de esta métrica es 100%.

- **Proporción de usuarios del módulo con utilización del inicio de sesión super – administrador (*sa*):**

Esta métrica se define de la siguiente forma:

$$PUSA = TUSA/TUM \quad (24)$$

Dónde:

PUSA: Porcentaje de usuarios que utilizan el módulo ocultamiento de la conexión al servidor.
TUSA: Total de usuarios que utilizan el inicio de sesión "*sa*".
TUM: Total de usuarios que utilizan el módulo en estudio.

El valor ideal de esta métrica es 0%.

4.1.3 ¿Qué pasos se deben seguir para realizar el proceso de refactorización de la arquitectura?

Los pasos generales fueron registrados y las acciones realizadas para cada atributo fueron tabulados a partir de la experiencia realizada.

4.1.4 ¿Qué impacto tiene la refactorización en el valor técnico del producto?

Se aplicaron las mismas métricas explicadas en el numeral 4.1.2, se realizó la medición en los atributos del valor técnico del Módulo SION - Admisiones y se contrastaron los resultados obtenidos para determinar la mejora de la calidad interna del Sistema.

4.2 ASIGNACIÓN DE PESOS

Para realizar la valoración se tuvo en cuenta los dos ejes principales a evaluar en el módulo, es decir, el valor de negocio y el valor técnico. Se determinó que la evaluación en general tendría una puntuación entre 0,0 y 5,0, donde el valor aprobatorio de la métrica sería una nota mayor a 2,5.

Para determinar el peso de cada atributo por evaluar, se seleccionó primero el personal adecuado para la evaluación, de acuerdo con cada dimensión por evaluar. De manera consensuada se determinaron los pesos, según el criterio de los participantes en esta evaluación. La asignación de los pesos se hizo desde las subcaracterísticas, para luego asignarle los pesos a los atributos y finalmente a las métricas de cada atributo.

Al multiplicar el valor numérico obtenido en la medición de cada métrica por el peso asignado, se encontraron los valores que al sumarse en total dieron la nota definitiva en cada eje; cabe resaltar que cada dimensión es independiente uno del otro, y como tal, cada valor ubica al módulo en estudio en una coordenada específica, para así determinar la ubicación del módulo en el mapa de cuadrantes explicado en la sección 2.2.

A continuación se explica de manera más detallada las evaluaciones tanto para el Valor de Negocio como para el Valor Técnico.

4.2.1 Modo de evaluación del Valor de Negocio.

El personal seleccionado que evaluó la aplicación tenía los siguientes perfiles:

- Evaluador Funcional (F): En este caso, la persona encargada de dirigir la Oficina de Admisiones y Registro Académico de la Institución.

- Evaluador Académico (A): El Decano de la Facultad de Ingeniería de la Institución, aprovechando que su perfil es técnico (Ingeniero de Sistemas) y con amplios conocimientos en la evaluación de la calidad de productos software.

- Evaluador Técnico (T): El representante del equipo de Desarrollo del Sistema de Información de la Institución.

En la Tabla 4.3 se despliegan los pesos asignados por cada evaluador para el Valor de Negocio del módulo. Cabe aclarar que la subcaracterística "Valor económico" solo posee un atributo (Valor del mercado) y por ende el valor del peso para atributo es 100%, de igual modo sucede en el atributo "Valor de Información". Por otra parte, las subcaracterísticas "Utilidad" y "Especialización" poseen cada una varios atributos por evaluar, y por ello sus pesos tienen diversos valores cuya sumatoria es 100%, de manera similar ocurre con el cálculo del peso para las métricas, puesto que la mayoría de atributos tiene una sola métrica, el valor del peso correspondiente es 100%, salvo los atributos de "Frecuencia actual de uso", "satisfacción de usuario", "cobertura de funcionalidad altamente especializada desarrollada" y "cobertura de funcionalidad genérica desarrollada", los cuales tienen dos métricas cada uno y tienen sus respectivos pesos asignados. Para calcular el peso final, se multiplica el peso de la métrica por el peso del atributo y por el peso de la subcaracterística; en aquellas métricas cuyo peso es 100% se multiplica solo el peso de la subcaracterística por el peso del atributo.

SUBCARACTERÍSTICA	PESO	ATRIBUTO	PESO ATRIBUTO	MÉTRICA	PESO MÉTRICA	PESO FINAL
VALOR ECONÓMICO	10%	Valor del Mercado	100%	PDM	100%	**10,0%**
VALOR DE LA INFORMACIÓN	35%	Valor de la información	100%	VI	100%	**35,0%**
UTILIDAD	35%	Cobertura de funcionalidad de negocio	40%	PC	100%	**14,0%**
		Frecuencia actual de uso	40%	PTA	90%	**12,6%**
				PUFAU	10%	**1,4%**
		Satisfacción de usuario	20%	SU	100%	**7,0%**
ESPECIALIZACIÓN	20%	Cobertura de funcionalidad altamente especializada desarrollada	50%	PFE	90%	**9,0%**
				UPFE	10%	**1,0%**
		Cobertura de funcionalidad genérica desarrollada	50%	PFG	90%	**9,0%**
				UPFG	10%	**1,0%**
TOTALES	**100%**					**100,0%**

Tabla 4.3. Pesos asignados para las subcaracterísticas y los atributos correspondientes al Valor de Negocio. Fuente: Elaboración propia.

4.2.2 Modo de evaluación del Valor Técnico

Para la evaluación de esta dimensión se determinó contar con el evaluador Académico y el Técnico, y reemplazar al evaluador de Negocio por el de Soporte, ya que éste último conoce más los criterios de calidad interna que debe tener un producto software y su formación también es técnica.

En la Tabla 4.4 se despliegan los pesos para cada variable y métrica respectiva. El cálculo del peso final es similar al explicado para el cálculo del peso final en la dimensión del Valor de Negocio.

SUBCARACTERÍSTICA	PESO	ATRIBUTO	PESO ATRIBUTO	MÉTRICA	PESO MÉTRICA	PESO FINAL
MANTENIBILIDAD	25%	Longitud de código	10%	TLOC	100%	2,5%
		Proporción de código muerto	20%	PCM	100%	5,0%
		Complejidad ciclomática	35%	PCC	50%	4,4%
				PRCC	50%	4,4%
		SQL embebido	35%	TSQL	100%	8,8%
DESCOMPONIBILIDAD	25%	Modularidad – Acoplamiento	50%	PA	100%	12,5%
	25%	Modularidad – Cohesión	50%	PC	100%	12,5%
DETERIORO	10%	Trabajo acumulado no completado	30%	PTA	100%	3,0%
		Defectos del módulo	20%	PD	100%	2,0%
		Esfuerzo de soporte en tiempo crítico	20%	ETC	100%	2,0%
		Tiempo de respuesta	30%	PITR	100%	3,0%
OBSOLESCENCIA	15%	Adaptabilidad en ambientes software	10%	PAS	100%	1,5%
		Adaptabilidad en ambientes hardware	10%	PAH	100%	1,5%
		Soporte herramienta IDE	60%	SIDE	100%	9,0%
		Soporte motor base de datos	20%	SMBD	100%	9,0%
DESEMPEÑO	10%	Comportamiento en el tiempo	40%	PTR	100%	4,0%
		Consumo de recursos – CPU	30%	PCPU	100%	3,0%
		Consumo de recursos - IO	30%	PIO	100%	3,0%
SEGURIDAD	15%	Cifrado de credenciales	30%	PUC	100%	4,5%
		Ocultamiento de conexión al servidor	30%	PUOC	100%	4,5%
		Utilización de inicio de sesión privilegiado	40%	PUSA	100%	6,0%
TOTALES	100%					100%

Tabla 4.4. Pesos asignados para las subcaracterísticas y los atributos correspondientes al Valor Técnico. Fuente: Elaboración propia.

4.3 ENCUESTA PARA LA RECOPILACIÓN DE INFORMACIÓN CUALITATIVA

Para determinar los valores cualitativos del módulo en estudio, se diseñó una encuesta de 21 preguntas. Dicha encuesta fue dividida (de manera implícita) en 5 secciones, para determinar algunas de las métricas propuestas en el trabajo. En la Tabla 4.5 se visualizan las subcaracterísticas, métricas y las preguntas que fueron asignadas para poder realizar las respectivas mediciones.

SUBCARACTERÍSTICA	ATRIBUTO	MÉTRICA	PREGUNTAS
UTILIDAD	Frecuencia actual de uso	PUFAU	1
VALOR DE LA INFORMACIÓN	Valor de la información	VI	2 – 5
ESPECIALIZACIÓN	Cobertura de funcionalidad genérica desarrollada	UPFG	6
	Cobertura de funcionalidad altamente especializada desarrollada	UPFE	7
UTILIDAD	Satisfacción de usuario	SU	8 – 21

Tabla 4.5. Secciones de la encuesta de acuerdo con las subcaracterísticas, atributos y métricas respectivas. Fuente: Elaboración propia.

Para realizar la valoración de las métricas, se hizo a través de la escala de Likert. En la Tabla 4.6 se expone la escala con sus respectivos valores.

ESCALA	VALOR
Muy en desacuerdo	1
En desacuerdo	2
No estoy seguro	3
De acuerdo	4
Muy de acuerdo.	5

Tabla 4.6. Escala de Likert y sus respectivos valores. Fuente: [45]

Las preguntas realizadas para la encuesta se visualizan en la Tabla 4.7.

N°	PREGUNTA	MÉTRICA
1	El Módulo se encuentra siempre disponible para realizar mis labores cotidianas.	PUFAU
2	La información que me brinda el Módulo es confiable.	VI
3	Las consultas y reportes que me brinda el módulo son exactas y no se presentan inconsistencias.	VI
4	La información que se gestiona en el Módulo es de suma importancia para mi trabajo diario; sin el apoyo de este sistema no podría realizar mis labores.	VI
5	Muchas veces el cumplimiento de mis responsabilidades se ven afectados porque el módulo no me proporciona información confiable y oportuna.	VI
6	La funcionalidad ofrecida por el módulo apoya de manera completa los procesos propios de la Oficina de Admisiones y Registro Académico	UPFG
7	El Módulo contiene funciones que abarcan las necesidades particulares de la Corporación Universitaria Adventista que la Oficina de Admisiones y Registro Académico debe llevar a cabo	UPFE
8	La navegación en el Módulo es fácil.	SU
9	La búsqueda de información y selección de un elemento en el Módulo (un estudiante, docente, curso, instancia de curso, etc.) es sencilla.	SU
10	El Registro o modificación de la información de un ente (un estudiante, un programa académico, curso, instancia de curso, evento) se realiza de manera sencilla.	SU
11	La apariencia del Módulo es estética y agradable, facilitando el trabajo cotidiano.	SU
12	Para operar el Módulo se requiere hacer una capacitación extensa y un continuo acompañamiento de los técnicos.	SU
13	La manera como se comunica el Módulo conmigo en la medida que trabajo con él (mensajes, advertencias, etc.) es entendible.	SU
14	La documentación de ayuda que tiene el Módulo es la apropiada.	SU
15	El Módulo presenta errores continuamente mientras se opera con él.	SU
16	Cuando se solicita información al Módulo, éste despliega dicha información en el tiempo esperado.	SU
17	Considero que el Módulo es un activo para la Corporación Universitaria Adventista.	SU
18	Desde el inicio de mis labores con el Módulo, ha tenido una evolución continua y de mejora progresiva	SU
19	En general me encuentro satisfecho con el Módulo.	SU
20	Considero que el Módulo ya cumplió su ciclo en la institución y que se debe pensar en comprar o en desarrollar un nuevo módulo.	SU
21	El área de informática da una respuesta oportuna y acertada a las sugerencias de cambio al Módulo.	SU

Tabla 4.7. Preguntas y su métrica respectiva, de la encuesta realizada a los usuarios del Módulo SION – Admisiones. Fuente: Elaboración propia.

La ficha técnica de la encuesta es la siguiente:

- **Marco de referencia:** Local

- **Característica de la encuesta:** Cuestionario directo y estructurado.
- **Universo:** Usuarios directos del Módulo de la Oficina de Admisiones y Registro Académico.
- **Muestra**: Todo el universo. 4 personas.
- **Tipo de observación:** Directa.
- **Fecha de realización de la encuesta:** 29 – 30 de noviembre de 2012

4.4 HERRAMIENTAS UTILIZADAS

En esta experiencia se utilizaron diversos métodos de recolección de datos junto con algunas herramientas debido a la naturaleza de las métricas seleccionadas. A continuación se explican las diversas formas de recolección de datos y en qué atributos fueron utilizadas.

4.4.1 MZ – Tools 3.0

Es un programa gratuito *add – in* para Visual Basic 6.0, Visual Basic 5.0 y Visual Basic para Aplicaciones (para MS - Office de 32 bits), el cual añade muchas características de productividad para el IDE. Esta herramienta es de fácil uso, y ofrece características para encontrar código más rápido, ayuda de diseño, soporte para la calidad del trabajo, codificación, y apoyo en la documentación de los proyectos; esta última característica fue utilizada para especificar los procedimientos y funciones de cada clase, además de determinar la longitud de código de cada procedimiento [46]. Véase la Figura 4.1.

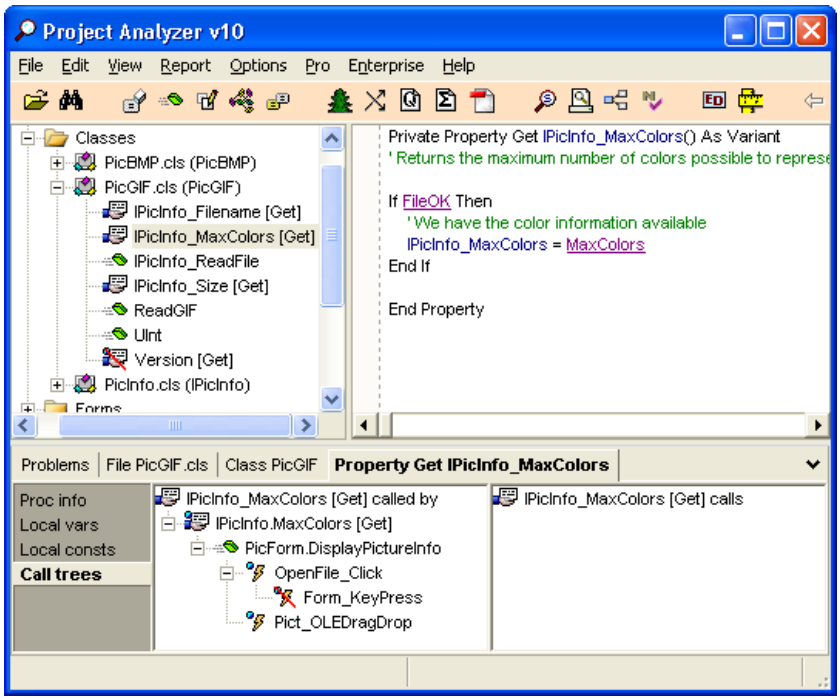

Figura 4.1. Interfaz de la herramienta Project Analyzer. Fuente [46]

4.4.2 Project Analyzer

Es una herramienta que revisa el código Visual Basic en diferentes versiones (.NET y VB6) que da apoyo a la calidad. Ayuda a entender código legado pobremente documentado, y apoya la ingeniería inversa. Detecta código muerto y genera documentación del proyecto, además de otorgar 184 métricas de código, tales como líneas de código, complejidad ciclomática, complejidad relativa, la profundización de anidación, entre otras características [47]. En este trabajo esta herramienta fue utilizada para determinar métricas de complejidad ciclomática y líneas de código, además de apoyo para la documentación de las clases y sus respectivos métodos. La herramienta utilizada fue la versión 10.0.02.

4.4.3 Data Dictionary Creator

Es una aplicación simple que ayuda a documentar las bases de datos de SQL Server, almacenando toda la información de las propiedades extendidas, de tal modo que se puede guardar de manera sencilla y sincronizar con la base de datos [48]. Con esta herramienta se realizó la recuperación de la documentación de la base de datos principal del Módulo para obtener un conocimiento del sistema. Véase la Figura 4.2.

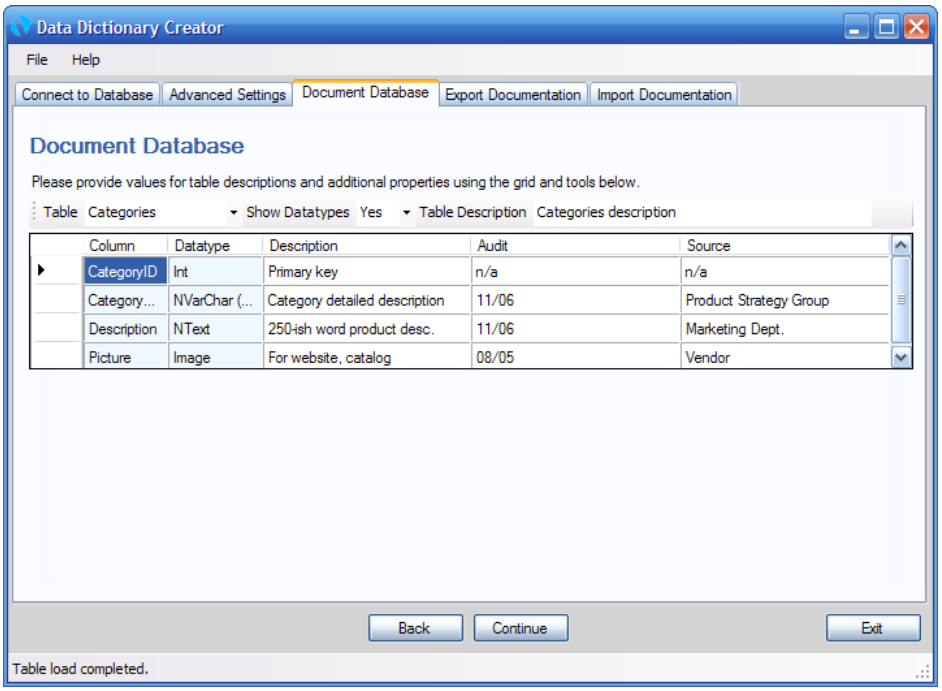

Figura 4.2. Ventana de la aplicación Data Dictionary Creator. Fuente [48]

4.4.4 DB Designer 4

Es una herramienta de diseño de base de datos que integra el diseño, modelado, creación y mantenimiento de la base de datos [49]. Esta herramienta es útil para realizar la ingeniería inversa de la base de datos, visualizando las relaciones existentes entre las tablas. Véase la Figura 4.3.

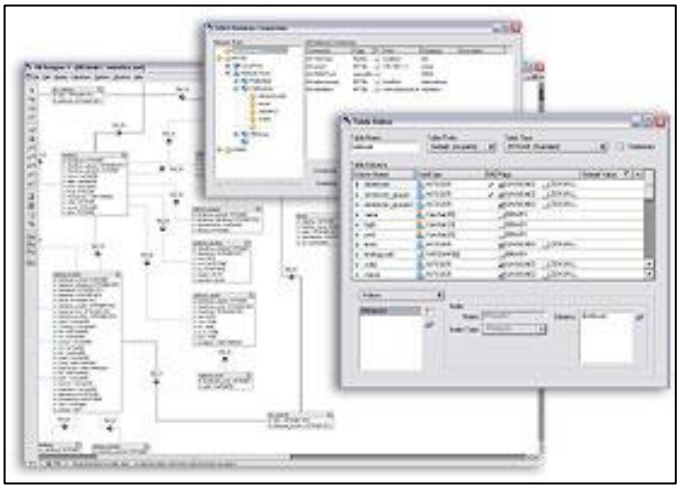

Figura 4.3. Herramienta DB Designer 4. Fuente [49].

4.4.5 MS – Visio 2010

Es una herramienta de diagramación avanzada con el que se puede hacer diferentes tipos de diagramas, como cuadros organizacionales, diagramas de redes, procesos de negocio, diseño de software, entre otros; ofrece una serie de plantillas que facilitan al usuario la creación de diagramas [50]. Con esta herramienta se realizó la ingeniería inversa de los componentes del módulo, analizando desde el componente .EXE pasando por los componentes de lógica de negocio y de datos. La herramienta genera un diagrama estático de paquetes.

4.4.6 Managment Studio de MS – SQL Server

Es un entorno integrado para acceder a todos los componentes de SQL Server, configurarlos, administrarlos y desarrollarlos, combinando un grupo de herramientas gráficas con sus respectivos editores de script para que tanto los desarrolladores como los administradores puedan acceder en todos los niveles de especialización [51]. En esta experiencia esta herramienta fue fundamental para el análisis del costo de las consultas, la ejecución de procedimientos almacenados, creación, actualización de índices, desfragmentación de índices, la creación y actualización de las relaciones entre tablas.

4.4.7 ORACLE VM Virtual Box

Producto para virtualización de sistemas operativos (Windows NT, 2000, XP, Server, Vista, 7, 8, Linux, Solaris, OS/2 y OpenBSD), el cual funciona en cualquier sistema operativo [52]. Con esta herramienta se realizó la medición del rendimiento de las consultas SQL del módulo, para tener condiciones homogéneas de medición antes y después del proceso de mejora de ellas.

4.4.8 Herramienta CONEXION

Esta herramienta fue un desarrollo propio de este trabajo, para poder determinar las llamadas entre componentes, además de determinar las llamadas a instrucciones SQL embebidas y las llamadas que el Sistema hace a los diferentes métodos de cada clase. Se puede determinar la conexión para un componente en particular o todas las conexiones de un componente, para ello se requiere la configuración previa de los componentes del módulo. La herramienta es una versión previa no definitiva y se desarrolló para un propósito específico en este trabajo, sin embargo, con ella se pudo realizar las mediciones necesarias y puede ser mejorada para la exportación de información en MS – Excel, reportes, configuración de información, entre otras tareas. Véase la Figura 4.4**.**

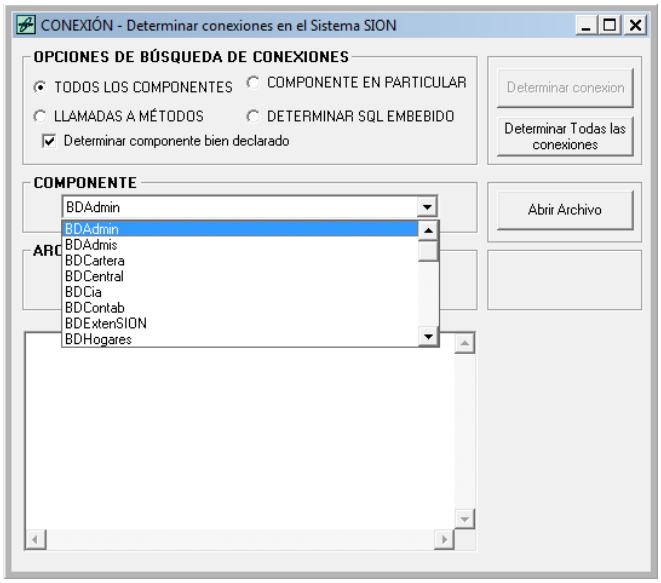

Figura 4.4. Ventana de la herramienta CONEXION. Fuente: Elaboración propia.

4.5 MODO DE RECOLECCIÓN DE DATOS

4.5.1 Encuesta a Usuarios

Por medio de la encuesta se buscaba conocer principalmente la satisfacción del usuario frente al sistema, además de la percepción que ellos tenían con respecto al valor de la información que se gestiona en el módulo, disponibilidad del aplicativo, la funcionalidad genérica y la funcionalidad específica que éste tiene. La encuesta se explicó en el numeral 4.3 con mayor profundidad.

4.5.2 Sistemas de Mesa de Ayuda

El DSI cuenta con el área de Mesa de Ayuda, el cual utiliza el sistema GLPI, el cual es una aplicación de software libre que gestiona recursos de información de una empresa, por medio del cual se pueden crear inventarios de los recursos técnicos de la organización (computadores, impresoras, etc.), así como la gestión de la historia de las solicitudes de mantenimiento [53]; en dicho sistema se registran todas las solicitudes concernientes a los módulos del sistema SION. El correo electrónico sirve además como respaldo de dichas solicitudes.

Con las dos fuentes mencionadas anteriormente se extrajo la información concerniente a la frecuencia actual de uso del módulo, la proporción de trabajo acumulado no terminado, la proporción de tasa de defectos reportados del módulo, el incremento de tiempo de respuesta, el esfuerzo de soporte en tiempo crítico, la adaptabilidad en ambientes hardware y software y la resistencia al acceso.

4.5.3 Censo de Módulos instalados

El área de Soporte del DSI realizó un censo de las aplicaciones instaladas en cada dependencia de la institución, por medio del cual se pudo determinar el tipo de aplicación que utiliza cada usuario y como base para el mejoramiento de la seguridad de todos los módulos del Sistema.

4.5.4 Documentación de especificación de requisitos

La documentación de especificación de requisitos encontrada para la realización de esta experiencia databa del proceso inicial de documentación realizado a principios del año 2000, además de algunas especificaciones realizadas en documentos MS – Word y hojas de cálculo.

5. LA ESTRATEGIA DE REFACTORIZACIÓN DEL MÓDULO

La estrategia propuesta en este trabajo consta de una serie de etapas que permitieron llevar a cabo esta experiencia, tal como se ilustra en la Figura 5.1.

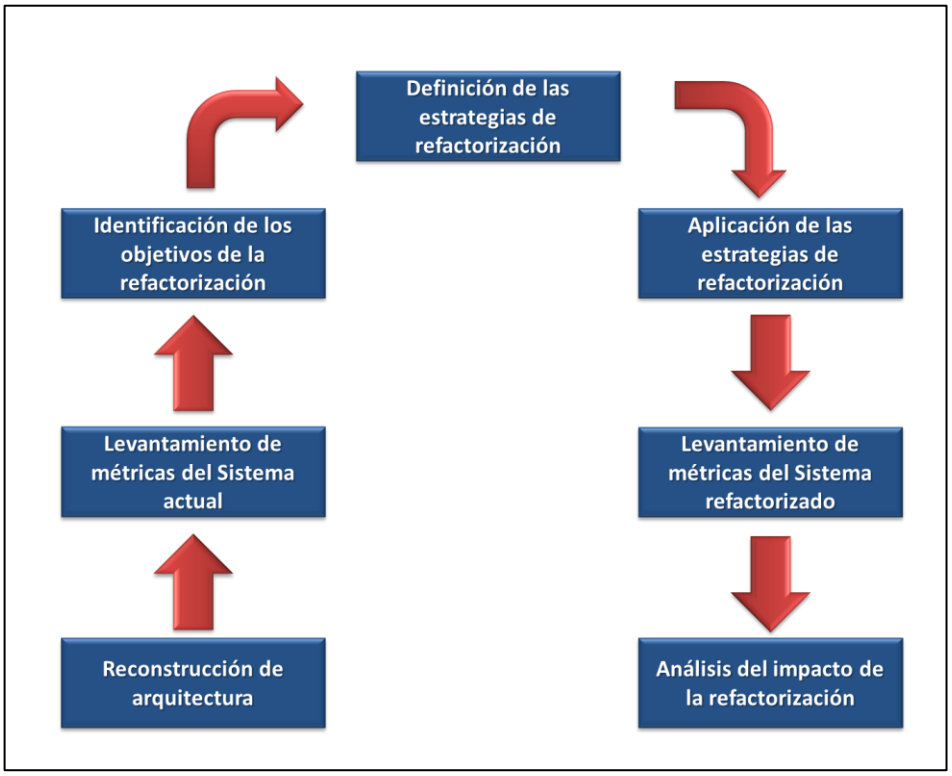

Figura 5.1. Proceso de estrategia de refactorización. Fuente: Elaboración propia.

En este modelo se analiza el código fuente de la aplicación y sus objetos derivados (como ventanas, reportes, módulos, etc.), además de la documentación existente del mismo; este análisis se hace por medio de modelos que describan al sistema e incrementan su nivel de abstracción hasta llegar a una representación arquitectónica. Una vez realizada esta reconstrucción se realizan las mediciones a través de las métricas y se identifican los objetivos que se quieren lograr por medio de la refactorización. El sistema entonces se rediseña para alcanzar un nivel predefinido de atributos de calidad, los cuales son seleccionados de las variables establecidas en el Valor Técnico que son susceptibles a mejora de manera inmediata; finalmente, se ejecutan las estrategias para llevar a cabo el rediseño propuesto, tales como la implementación de funciones o métodos, la adaptación de funciones, los cambios estructurales como la eliminación o remoción de clases, métodos e incluso librerías enteras para llevar a cabo dichas transformaciones, se vuelven a realizar las mediciones con las métricas establecidas y finalmente se analizan los cambios producidos en el sistema

refactorizado para contrastarlo con el estado inicial del mismo y así obtener conclusiones del proceso. Las secciones siguientes describen cada uno de estas etapas.

5.1 RECONSTRUCCIÓN DE LA ARQUITECTURA DEL SISTEMA EXISTENTE.

En esta etapa se realizaron los siguientes pasos:

- **Recopilación de la información técnica existente del Sistema y del módulo en particular:** Especificación de requisitos, descripciones del ámbito perteneciente al módulo, modelos desarrollados, etc.

- **Reuniones de trabajo:** Tanto con los usuarios finales del módulo como con el equipo técnico encargado del mantenimiento del mismo, para lograr el entendimiento del sistema.

- **Detección del estilo arquitectónico del sistema:** Por medio de una matriz se analiza el Sistema de acuerdo con la definición de cada estilo arquitectónico; estas definiciones se basaron de la Guía de Arquitectura de Aplicaciones de Microsoft [54]. En el caso particular, se llega a la conclusión que el Sistema tiene un estilo arquitectónico híbrido, Capas – Cliente / Servidor. Véase la Tabla 5.1.

ESTILO ARQUITECTÓNICO	DESCRIPCIÓN	SE CUMPLE EN EL SISTEMA	NO SE CUMPLE EN EL SISTEMA
CLIENTE/SERVIDOR	Segrega el Sistema en dos aplicaciones, donde el cliente hace peticiones al servidor. En muchos casos, el servidor es una base de datos con lógica de negocio representada en procedimientos almacenados.	Existe lógica de negocio en procedimientos almacenados. La data se encuentra totalmente centralizada	Gran parte de su lógica de negocio ha sido desarrollada en componentes y no están radicados en el servidor de aplicaciones
BASADO EN COMPONENTES	Descompone el diseño de la aplicación en componentes lógicos o funcionales reutilizables que exponen interfaces de comunicación bien definidas	Se han desarrollado a lo largo de los años de vida del Sistema varios componentes que encapsulan tanto el acceso a los datos como la funcionalidad de la lógica de negocio. Varios componentes tienen la ventaja de poder ser reutilizables, y tienen cierta encapsulación (no exponen su funcionalidad interna)	Pobre utilización de interfaces de comunicación bien definidas. El concepto de Contenedor de Componentes y Servicios no existe.
DISEÑO	Estilo arquitectónico orientado a objetos	El Sistema fue desarrollado	No existe un lenguaje unificado

ESTILO ARQUITECTÓNICO	DESCRIPCIÓN	SE CUMPLE EN EL SISTEMA	NO SE CUMPLE EN EL SISTEMA
DIRECCIONADO AL DOMINIO	enfocado en modelar un dominio de negocio y definir objetos de negocio basados en entidades dentro del dominio de negocio	pensado principalmente en el dominio de la Institución, tratando de abarcar todas las áreas que la conforman (soporte, estratégica y gerencial)	para el desarrollo de esta arquitectura. No hubo intención en desarrollar el Sistema bajo este estilo
CAPAS	Divide los intereses de la aplicación en grupos apilados (capas o layers)	El desarrollo del Sistema siempre estuvo direccionado a trabajarse con este estilo arquitectónico. Hay una capa bien definida de acceso a datos, otra de lógica de negocio (conocida como Reglas de Negocio) y otra de presentación.	Hay capas con responsabilidades mal asignadas, por ejemplo, lógica de negocio distribuida en la interfaz de usuario o en el componente de datos. No se utilizan los beneficios de las interfaces para comunicarse entre diferentes capas.
BUS DE MENSAJES	Prescribe el uso de un sistema software que puede recibir y enviar mensajes usando uno o más canales de comunicación, así las aplicaciones pueden interactuar sin necesidad de conocer detalles específicos sobre los demás	El Sistema nunca fue direccionado bajo este estilo arquitectónico	No aplica
3 NIVELES/N – TIERS	Segrega la funcionalidad en segmentos separados de la misma manera que el estilo por capas, pero cada segmento es un nivel que se localiza en un computador separado físicamente	Estilo arquitectónico intencionado para el Sistema.	Nunca se separaron las capas en diferentes máquinas, por la complejidad que hubo de implementar el servicio COM+, hoy en desuso.
ORIENTADO A OBJETOS	Un paradigma de diseño basado en la división de responsabilidades para una aplicación en objetos individuales y autosostenibles, cada uno conteniendo los datos y el comportamiento relevante.	Cada componente desarrollado para el Sistema trata de cumplir su responsabilidad (por ejemplo, objetos de conexión a datos). En los objetos de Interfaz de Usuario se crearon Propiedades y métodos que acceden a dichas propiedades.	La herramienta en el que ha sido desarrollado gran parte del Sistema no ofrece todas las características ideales de la Orientación a Objetos (como la herencia, o el polimorfismo).
ORIENTADO A SERVICIOS	Se refiere a aplicaciones que exponen y consumen funcionalidades como un servicio usando contratos y mensajes	El Sistema nunca fue direccionado bajo este estilo arquitectónico	No aplica.

Tabla 5.1. Análisis del estilo arquitectónico del Sistema SION. Fuente: Elaboración propia basado en [54].

- **Representación gráfica del módulo en particular:** Desde el IDE de VB6 se realiza la ingeniería inversa con el complemento de MS – Visio 2010 llamado Microsoft Visio Visual Basic Addin. Para cada componente del módulo en estudio la herramienta realiza un diagrama de paquetes, el cual se complementa analizando las referencias programadas por el componente. El diagrama resultante se visualiza en la Figura 5.2.

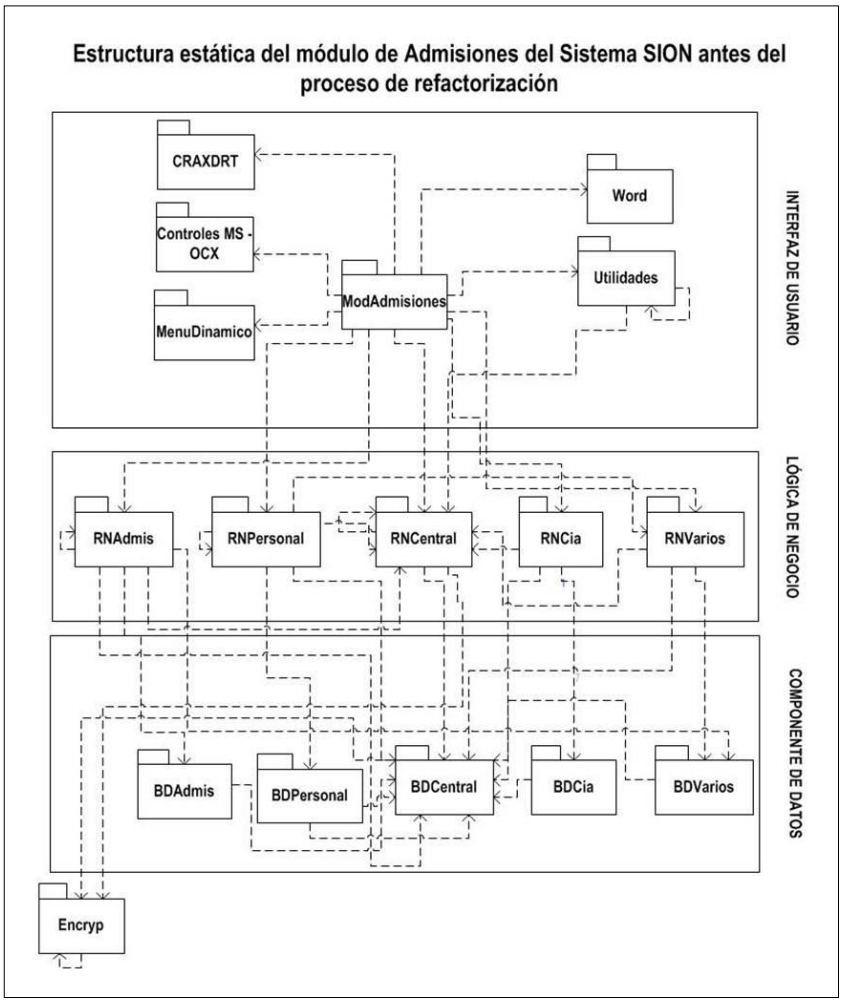

Figura 5.2. Representación estática del módulo SION – Admisiones, una vez realizada la ingeniería inversa. Fuente: Elaboración propia.

- **Representación gráfica de alto nivel de todo el Sistema:** De manera similar al ítem anterior, para cada módulo del Sistema. En la Figura 5.3 se visualiza la relación que hay entre los módulos del Sistema.

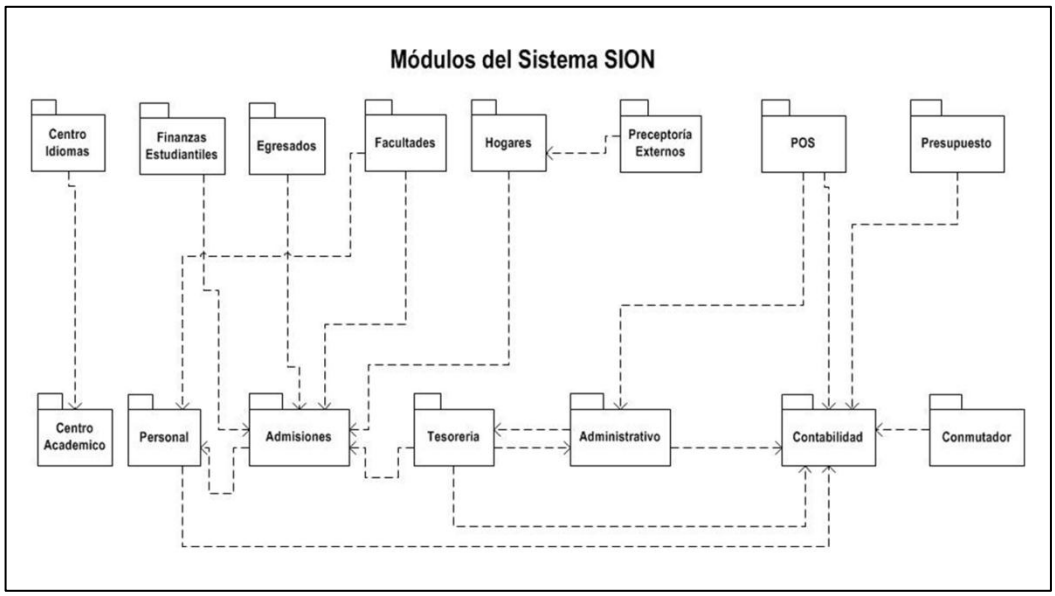

Figura 5.3. Representación gráfica de alto nivel del Sistema SION. Fuente: Elaboración propia.

- **Descripción general de los módulos del Sistema SION y la manera como se relaciona con el módulo en estudio.** Véase la Tabla 5.2.

MÓDULO	DESCRIPCIÓN BREVE	RELACIÓN DIRECTA CON EL MÓDULO DE ADMISIONES
Centro de Idiomas	Gestión de información de cursos, estudiantes y docentes del Centro de Idiomas de la Institución	NO
Finanzas Estudiantiles	Gestión de información financiera de los estudiantes (cobros, abonos, cartera)	SÍ
Egresados	Gestión de información de estudiantes graduados	SÍ
Facultades	Consulta de información académica y financiera de los estudiantes por facultad y programa académico	SÍ
Hogares Universitarios	Gestión de información de estudiantes residentes (internado)	SÍ
Preceptoría de Externos	Gestión de información de estudiantes no residentes (externos)	SÍ
Punto de Venta (POS)	Sistema de facturación utilizado en puntos de venta, como cafeterías y mini – mercado	SÍ
Presupuesto	Gestión del control presupuestal de cada centro de responsabilidad de la Institución	NO
Centro Académico	Gestión de recursos educativos (salones, préstamos de implementos, etc)	NO
Personal (Recursos Humanos)	Gestión de información de docentes y empleados de la Institución	SÍ
Tesorería	Gestión de ingresos y egresos	SÍ
Administrativo	Gestión de facturación e inventarios	NO
Contabilidad	Gestión de información contable	NO
Conmutador	Gestión de información de llamadas telefónicas entrantes y salientes	NO
Seguridad	Gestión de usuarios y permisos en todos los módulos	SÍ

Tabla 5.2. Descripción general de los módulos de SION. Fuente: Elaboración propia.

- **Recopilación de las funciones del Módulo en estudio.** Esta tarea se realiza con el equipo técnico de soporte y desarrollo del Sistema y los usuarios del sistema.

- **Ingeniería inversa de las bases de datos del Módulo.** Se realizó por medio de la herramienta DB – Designer, para determinar las relaciones, claves principales y foráneas en las tablas de las bases de datos del módulo.

- **Creación del diccionario de datos del sistema.** Este diccionario fue realizado con la herramienta Data Dictionary Creator, a partir de las descripciones de las tablas y campos existentes en las bases de datos.

- **Generación de la documentación de cada componente del módulo en estudio de las capas de datos y lógica de negocio.** Con la herramienta MZ – Tools se puede realizar dicha documentación; se encuentra como complemento del IDE de Visual Basic 6 y su uso es libre. Dicha documentación hace una relación de clases con sus respectivos métodos. La documentación se complementa con la descripción de cada método y componente, de acuerdo con lo documentado en las líneas de código comentadas y la documentación dada por el equipo técnico de soporte y desarrollo del Sistema.

- **Generación de "Reporte de calidad de diseño" por cada componente del módulo en estudio de las capas de datos y de lógica de negocio.** Esta tarea se realizó por medio de la herramienta Project Analyzer. De este reporte se extrae la clasificación de los procedimientos por longitud de código (LOC), el total de LOC y la información de la complejidad ciclomática.

- **Tabulación de número de llamadas entre clases y componentes.** Actividad realizada por medio de la herramienta CONEXIÓN. La herramienta extrae una consulta que se lleva a una tabla dinámica en MS – Excel. Con esta tabulación se determina el total de llamadas por componente y por método.

- **Detección de las consultas SQL más costosas del sistema.** Dicha tarea se realiza al ejecutar todas las operaciones del Sistema en un ambiente "stand – alone", extrayendo los tiempos de ejecución del "Monitor de Actividad" de SQL Server, para luego tabularlos y clasificarlos de mayor a menor tiempo de respuesta.

5.2 LEVANTAMIENTO DE LAS MÉTRICAS DEL SISTEMA ACTUAL

Con las métricas estipuladas en el capítulo anterior, se realizan las mediciones para las dimensiones del valor técnico y del valor de negocio. En el siguiente capítulo se pueden observar los resultados y se explica para cada medición la forma como se realizó.

5.3 IDENTIFICACIÓN DE LOS OBJETIVOS DE LA REFACTORIZACIÓN

Se determinaron las subcaracterísticas que podían ser mejorados en el módulo. Tal como se explicó en el numeral 2.3, la refactorización busca la mejora interna de la calidad del sistema sin cambiar el comportamiento externo del sistema, y por ende las subcaracterísticas que podían ser mejoradas son las de la dimensión del Valor técnico, menos las subcaracterísticas de Deterioro y Obsolescencia (las cuales para determinar su mejora requieren de una medición a lo largo del tiempo). Véase la Tabla 5.3.

DIMENSIÓN	SUBCARACTERÍSTICA	¿A REFACTORIZAR?	JUSTIFICACIÓN
Valor de Negocio	Valor económico	NO	Depende de factores externos como el económico, la mano de obra, etc., fuera del alcance de la experiencia.
	Valor de información	NO	Depende de la organización y de la percepción de los usuarios del sistema.
	Utilidad	NO	Depende del ámbito del módulo, de un lapso de tiempo determinado y de la satisfacción del usuario.
	Especialización	NO	Fuera del alcance de la experiencia.
Valor Técnico	Mantenibilidad	SÍ	Mejoramiento de LOC, se puede eliminar el código muerto, la mejora de la complejidad ciclomática y separación de las instrucciones SQL que se encuentran embebidas en el módulo.
	Descomponibilidad	SÍ	Se puede reducir el acoplamiento entre los componentes y aumentar la cohesión en los mismos.
	Deterioro	NO	Depende de un lapso de tiempo mayor al tiempo estipulado para determinar la mejora del módulo.
	Obsolescencia	NO	Depende de los proveedores del IDE, del motor de base de datos y del hardware.
	Desempeño	SÍ	Se pueden aprovechar ciertas características que provee el motor de base de datos, de una manera ágil y sencilla.
	Seguridad	SÍ	Se pueden aprovechar las características de seguridad que provee el motor de base de datos y realizar estrategias sencillas en el Front – End.

Tabla 5.3. Justificación de la selección de subcaracterísticas factibles para refactorizar. Fuente: Elaboración propia.

5.4 DEFINICIÓN Y APLICACIÓN DE LAS ESTRATEGIAS DE REFACTORIZACIÓN

5.4.1 Mantenibilidad

- **Eliminación de código muerto:** Se determinaron aquellos métodos en los componentes VB6 que nunca son llamados por el Módulo o por otros módulos del Sistema; dicha detección se hizo a través de la herramienta CONEXIÓN referenciada en el numeral 4.4.8, y tabuladas en una hoja de cálculo en MS – Excel por medio de una tabla dinámica. Una vez determinados dichos métodos muertos se realizó la eliminación física, obligando a la recompilación de cada componente en el orden "bottom – up" (Desde el componente más inferior hasta la interfaz de usuario, pasando por los componentes de lógica de negocio).

- **Creación de método estándar de ejecución de procedimientos almacenados:** La inexistencia de un método único que pudiese ejecutar un procedimiento almacenado de SQL Server derivó en la creación de clases y métodos innecesarios o con longitud de código y complejidad ciclomática amplios, además del alto acoplamiento entre clases y la escritura de sentencias SQL embebidas en código VB6. Se diseñó entonces un método en VB6 que pudiera ejecutar dichos procedimientos almacenados, para que a través de estos se pudieran separar las instrucciones SQL de los métodos escritos en VB6 y favorecer así la mantenibilidad de la aplicación. Gracias a esto se determinó la factibilidad de eliminar las clases de los componentes de datos, salvo el componente de datos Central (BDCentral), ya que estos métodos solo contenían instrucciones de

selección SQL embebidas. De este modo, se crearon dos métodos escritos en VB6: Uno en el componente de datos (BDCentral) y otro en la capa de Lógica de Negocios (Componente RNCentral) que llame al primero; ambos métodos tienen un mismo nombre significativo ("Ejecutar") y devuelve un conjunto de resultados (conocido como Recordset de tecnología ADO). Los parámetros de entrada de dichos procedimientos son:

- Optimista: Si se requiere que el conjunto de resultados sea optimista (que se pueda acceder de manera dinámica) o no (de solo lectura y acceso solo hacia adelante).
- Nombre de la base de datos.
- Nombre del esquema: El esquema utilizado por el sistema por defecto es "dbo", pero se deja como parámetro si se requiere realizar bases de datos con esquemas diferentes.
- Nombre del procedimiento almacenado.
- Vector de Variables: Un vector que contiene todas las variables necesarias para ejecutar un procedimiento almacenado (es decir, como parámetros de entrada al procedimiento respectivo)

- **Detección de métodos de componentes de datos con operaciones SQL:** Los componentes de datos en este sistema fueron creados para ejecutar operaciones SQL, lo cual implicó escribir dichas instrucciones SQL dentro del código VB6 y por consiguiente una mala práctica de desarrollo, que impide un mantenimiento óptimo de la instrucción debido a las longitudes de líneas de código muy largas; las operaciones más comunes en el sistema fueron las de tipo selección. La estrategia determinada para eliminar esta mala práctica fue la escritura de dicha lógica en procedimientos almacenados en lenguaje T - SQL, para ser ejecutados por el motor SQL Server al ser llamados desde el método estándar "Ejecutar" (Escrito en VB6), descrito en el ítem anterior. De este modo, se crearon procedimientos almacenados para cada instrucción SQL de creación, actualización y eliminación y de selección (estas últimas, solo las que se consideraron como complejas). Como una buena práctica de desarrollo, cada instrucción SQL fue asignada a un procedimiento almacenado individual, además de un estándar particular para nombrarlos: [OPERACIÓN][Nombre significativo del procedimiento] (ejemplos: LISTAREstudiantes, CREARHojaDeVidaEstudiante, ELIMINARRegistroMatricula, ACTUALIZARCurriculo). Esta separación permite la adaptación de los permisos en el motor de la base de datos que deben ejecutarse desde el módulo (lectura y ejecución, como se explicó en el numeral anterior en la utilización de cuentas de inicio de sesión diferentes a *sa*).

- **Creación de procedimientos almacenados de búsqueda y selección de información y eliminación de métodos VB con esta funcionalidad:** Como se explicó en el numeral anterior, los componentes de datos en este sistema se desarrollaron con el fin de ejecutar operaciones CRUD en el sistema sin necesidad de

utilizar procedimientos almacenados, lo que conlleva a una mala práctica de desarrollo. Para cada método se asignó un procedimiento almacenado (escrito en T-SQL para ser ejecutado en el motor SQL Server) con una lógica estándar de acuerdo con los parámetros de entrada de dicho método, de tal modo que se eliminaron todos los componentes de datos del módulo (salvo BDCentral, el cual es el componente con el que la aplicación se comunica con el motor de la base de datos) y los métodos que los llamaban en las clases de la capa de la Lógica de Negocio.

5.4.2 Descomponibilidad

- **Asignación de responsabilidades a las clases pertinentes:** La responsabilidad en una clase es cualquier cosa que ella conoce o hace, es decir, son los atributos y operaciones relevantes para la clase [34]. Para cada clase se determinaron dichos métodos, previa definición del mismo realizado en la etapa anterior, y se hizo la asignación respectiva para cada clase. Una vez realizadas dichas asignaciones de responsabilidades, se requirió la recompilación de los componentes VB6 afectados.

- **Determinación de conexiones redundantes entre componentes:** De acuerdo con el diagrama de paquetes realizado en la actividad de la ingeniería inversa (descrito en el numeral 5.1), se establecieron las conexiones redundantes en los componentes VB6. Se establecieron qué clases y métodos son los que están realizando dichas llamadas, por medio de la tabulación de llamadas entre componentes, descrito en la etapa anterior, para determinar cuán factible es eliminar dicha conexión. Se eliminaron las conexiones redundantes y se crearon las necesarias.

Gracias a estas actividades y a las explicadas en el ítem anterior se pudo desacoplar el módulo, tal como se visualiza en la Figura 5.4 y sus resultados se demuestran en el numeral 6.2.5**.**

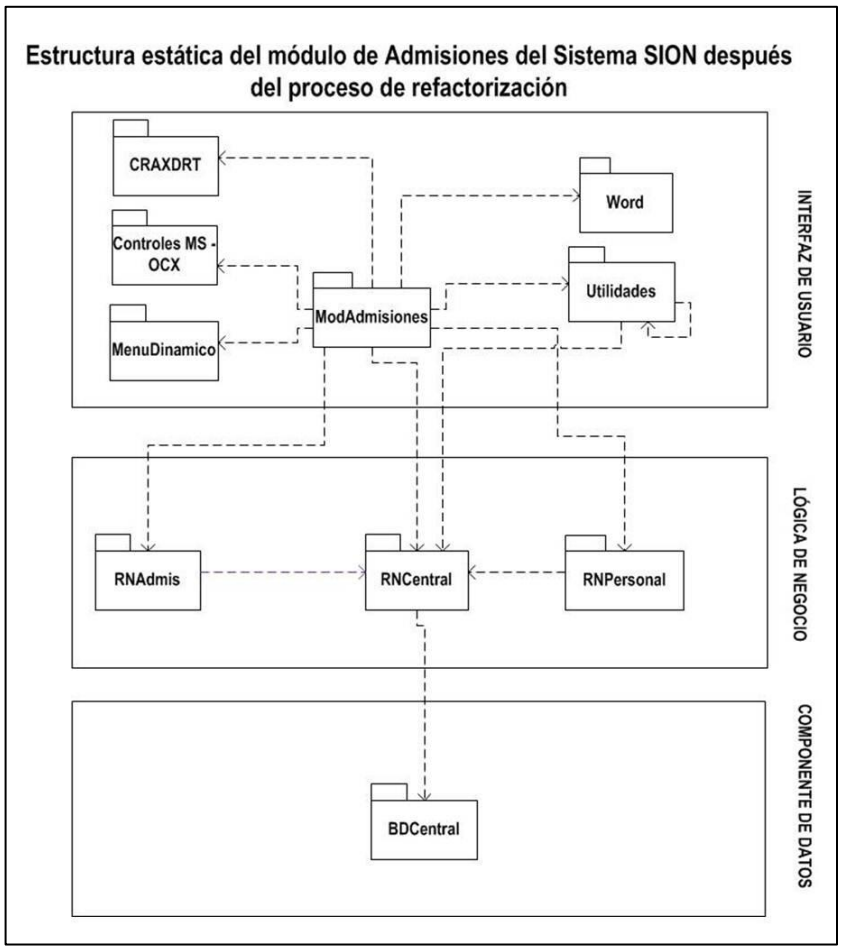

Figura 5.4. Estructura estática del Módulo SION – Admisiones después de la aplicación del proceso de refactorización. Fuente: Elaboración propia.

5.4.3 Desempeño

- **Mejoras en las bases de datos del Sistema:** Puesto que la característica de Desempeño depende de las bases de datos, se escogieron algunas estrategias de mejora en el rendimiento en la ejecución de las operaciones sin alterar el diseño de la base de datos. Dichas estrategias, llevadas a cabo por medio del MS – SQL Server Managment Studio, son las siguientes:

- **Mejora en la desfragmentación de las tablas de la base de datos principal del Módulo.** La fragmentación es un mal que debe evitarse principalmente en las bases de datos medianas o grandes, ya que este es uno de los causantes de los problemas de rendimiento en motores como SQL Server, causando operaciones de entrada y salida innecesarias, originando bloqueos, generando un efecto de "bola de nieve"

[55]. Para la desfragmentación de los índices, algunos se eliminaron, se crearon índices agrupados de la tabla, en otros casos se reconstruyeron los índices (la mejor opción para índices agrupados). Se recomendó además automatizar dicha de desfragmentación para que se haga de manera periódica y en horarios en que la base de datos sea de poca utilización (en el caso particular, en horas nocturnas) [55].

- **Detección y eliminación de registros "huérfanos" en las tablas.** Detectar y eliminar registros con clave heredada inexistente en la tabla padre. Este fenómeno se dio debido a que no había restricciones en las tablas para crear registros no relacionados. Este tipo de registros no solo demuestra un pobre control de la información en las bases de datos, sino además incrementa el tamaño de la base de datos de manera innecesaria. Se recomendó dejar una copia de seguridad con los registro huérfanos eliminados.

- **Eliminación de índices mal definidos en las tablas y creación de índices adecuados en las tablas determinadas.** Los índices mal definidos y la falta de índices son los principales orígenes de atascos en las aplicaciones de bases de datos [56]. Para determinar los índices óptimos para las consultas, MS –SQL Server Managment Studio puede mostrar el "plan de ejecución estimado", en el que no solo visualiza el peso de cada tarea que realiza el motor al ejecutar una instrucción, sino además sugiere los índices adecuados para optimizar la consulta, creando automáticamente un script de creación del índice respectivo; una vez implementado dicho índice sugerido, se requiere que se ejecute nuevamente la consulta para que la herramienta determine un nuevo índice para mejorar el rendimiento de la base de datos. Véase la Figura 5.5.

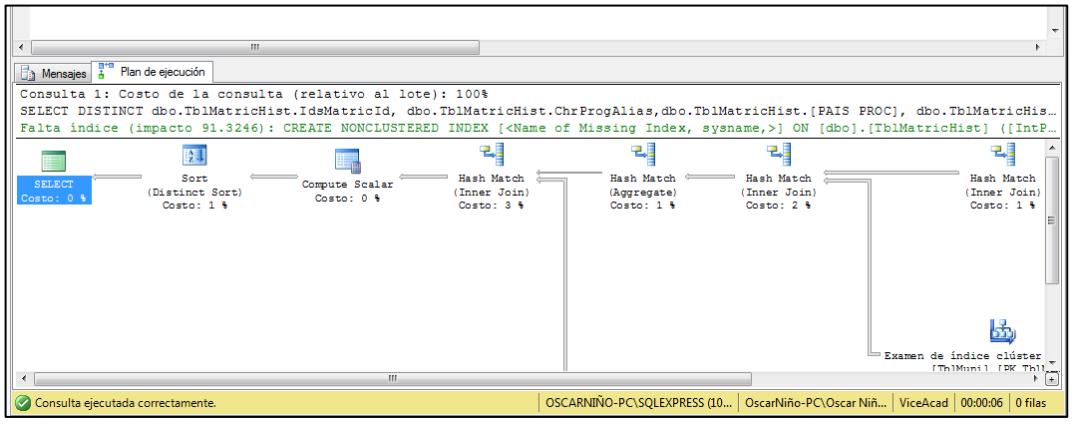

Figura 5.5. Ejemplo de índice sugerido por MS – SQL Server para una consulta en particular. Fuente: Elaboración propia.

- **Asignación de factor de relleno en los índices a crear.** El factor de relleno permite la optimización del almacenamiento y rendimiento de los datos de índice, es decir, determina el porcentaje de espacio en cada

página de nivel de hoja que se debe rellenar con datos, reservando el resto de cada página como espacio disponible para continuar creciendo [57]. La asignación de factores de relleno se hace de acuerdo con el margen de lecturas y escrituras que se hagan en la respectiva tabla; los ítems que se recomiendan son los siguientes:

- Pocas actualizaciones de tablas (100 – 1 lecturas sobre escrituras): 100% de factor de relleno.
- Muchas actualizaciones de tablas (Las escrituras exceden sobre las lecturas): Entre un 50% a un 70% de factor de relleno.
- Todo entre las anteriores: Entre un 80% y un 90% de factor de relleno. [58]

- **Creación de relaciones entre tablas de la base de datos.** La creación de las relaciones entre las tablas normalizadas evita la creación de registros huérfanos y contribuye a un buen diseño de la base de datos. Antes del proceso de mejora, se contaba con 77 relaciones, y luego de este proceso se crearon 16 más, para un total de 93, observando un incremento del 20,78%. Véase la Figura 5.6.

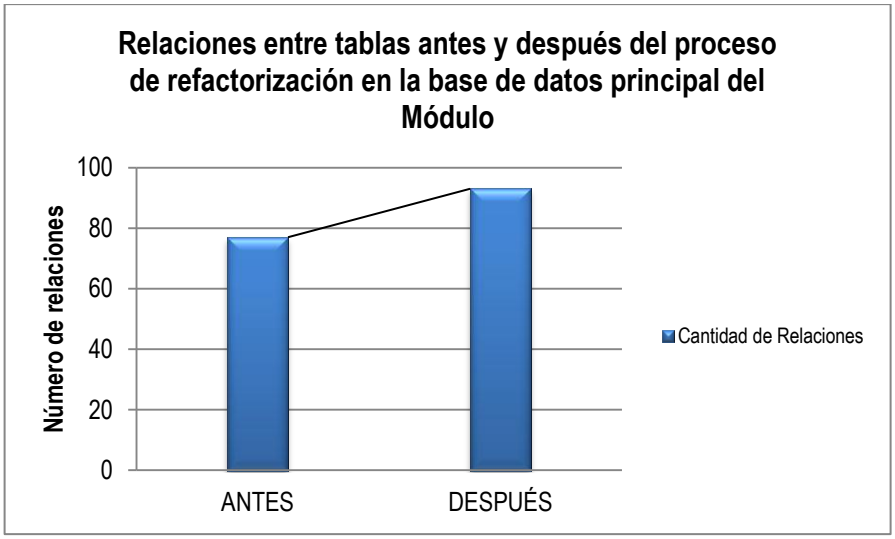

Figura 5.6. Relaciones entre tablas antes y después del proceso de refactorización en la base de datos principal del Módulo SION – Admisiones. Fuente: Elaboración propia.

5.4.4 Seguridad

- **Mejoras en la seguridad del Sistema:** En esta experiencia se seleccionó el atributo de la resistencia al acceso; las estrategias para mejorarlo fueron las siguientes:

- **Cifrado en las credenciales de usuario.** Utilizar algoritmos de cifrado probados y seguros, para cambiar los algoritmos propios adecuados para el módulo. En el módulo la validación de las credenciales de usuario se hace con un algoritmo propio de cifrado que no tiene respaldo técnico por alguna comunidad o grupo, lo cual lo hace vulnerable ante cualquier ataque. La estrategia que se llevó a cabo para la mejora de este ítem fue la creación en el módulo de un método de seguridad en VB6 que utilice un algoritmo de seguridad probado y de difícil (casi imposible) descifrado, y desechar los métodos utilizados para la validación del usuario del módulo.

- **Ocultamiento de parámetros de conexión al servidor de base de datos.** El módulo actualmente deja una huella en el registro del sistema operativo, en el que se indican explícitamente los parámetros de conexión (Nombre de base de datos, inicio de sesión, servidor, cadena de conexión); los parámetros de inicio de sesión y contraseña se encuentran cifrados con el algoritmo propio de cifrado. La estrategia diseñada para esta mejora fue adecuar la clase de seguridad del componente central, para que realizara el cifrado de toda la cadena de conexión con algoritmos seguros y desechar el método original para realizar la conexión con el servidor central.

- **Utilización de inicio de sesión diferentes al "*sa*" (System Administrator).** El "*sa*" es la cuenta de acceso principal y por defecto que el motor SQL Server dispone; dicha cuenta tiene acceso a todo (funciona de manera similar al "root" de Linux o LocalSystem de Windows NT), ya que cuenta con privilegios con el que se puede manejar cualquier base de datos (incluyendo la base de datos "master"), y con ella se puede hacer cualquier tipo de operación, incluyendo la ejecución de procesos. Esto da pie para que cualquier atacante del sistema comience su operación por dicha cuenta; de este modo, se puede considerar una mala práctica la utilización en modo producción de dicha cuenta, por la vulnerabilidad con la cual se expone el sistema [59]. Por tanto se creó una cuenta de inicio de sesión en MS – SQL Server, con permisos solo de lectura y ejecución de procedimientos almacenados.

5.5 LEVANTAMIENTO DE MÉTRICAS DEL SISTEMA REFACTORIZADO

Una vez aplicadas las mejoras en los atributos seleccionados, se aplicaron las mediciones con las métricas respectivas. Los resultados de dichas mediciones se explican en el capítulo 6.

5.6 ANÁLISIS DEL IMPACTO DE LA REFACTORIZACIÓN

Las mediciones realizadas al Sistema refactorizado se contrastaron con las del Sistema antes de refactorizar. Estos valores además se tabularon en la matriz de evaluación explicada en el numeral 4.2.2, para determinar la nueva posición en el mapa de cuadrantes, y compararlo con la posición del módulo antes del proceso de refactorización. Dicho análisis se explica en el capítulo 6.

5.7 ESFUERZO DE ESTRATEGIA DE REFACTORIZACIÓN

Para llevar a cabo cada actividad en esta estrategia, se registró el esfuerzo, medido en Horas – Hombre (H – H). En la Tabla 5.4 se despliegan dichos valores para las actividades de este proceso.

ACTIVIDAD	DESCRIPCIÓN BREVE	HORAS – HOMBRE
Desacoplamiento; mejoras de cohesión	Eliminación de métodos muertos; eliminación de llamadas redundantes	162
Mejoras en las bases de datos	Eliminación de registros huérfanos, creación de integridad referencial en relaciones faltantes, mejora de índices, defragmentación	30
Desarrollo método universal ejecución SQL	Método en VB6 para ejecutar una instrucción SQL con *n* parámetros de entrada.	6
Cifrado de credenciales de acceso a los módulos	Desarrollo método de cifrado de credenciales de acceso con algoritmo seguro	4
Cifrado cadena de conexión	Desarrollo métodos de cifrado de cadena de conexión	16
Implementación inicios de sesión	Implementación de inicios de sesión diferentes al *sa*	8
TOTAL		**226**

Tabla 5.4. Esfuerzo en Horas – Hombre para el desarrollo de actividades en la estrategia de refactorización. Fuente: Elaboración propia

6. ANÁLISIS DE LOS RESULTADOS

En este capítulo se despliegan los valores medidos según el valor de negocio y el valor técnico del sistema. Como ya fue mencionado, la refactorización tenía el objetivo de mejorar la calidad técnica del producto y por lo tanto, es la dimensión sobre la que el proceso de refactorización tiene su mayor impacto.

6.1 EVALUACIÓN DEL MÓDULO SEGÚN SU VALOR DE NEGOCIO

6.1.1 Porcentaje de diferencia del mercado

Determinar el valor de un producto software que ha sido desarrollado durante años requiere de algún método que pueda dar un valor aproximado del mismo (Líneas de Código, Puntos de Función, etc.). Para ello, se determinó su valor basándose en el modelo COCOMO II para determinar los puntos objeto del módulo en estudio; de acuerdo con este modelo, el punto objeto es una medida indirecta de software que se utiliza calculando el total de pantallas de interfaz de usuario, el número de reportes que la aplicación contiene y los componentes que se requieren construir para la aplicación, asignándolos a una complejidad determinada (básico, intermedio o avanzado). Las cantidades clasificadas de pantallas, reportes y componentes se multiplican por los respectivos pesos de complejidad para obtener la cantidad de puntos objeto, y luego se ajustan dichos puntos por medio del porcentaje de reutilización (componentes no exclusivos del módulo); finalmente, se calcula el esfuerzo de productividad de acuerdo con los diferentes niveles de experiencia que tenga el desarrollador y de madurez del entorno de desarrollo que este método especifica. [34].

Las pantallas, reportes y módulos del módulo en estudio se cuantificaron y se ponderaron con los criterios especificados anteriormente y estos resultados se visualizan en la Tabla 6.1.

OBJETO	COMPLEJIDAD	PESO	CANTIDAD	TOTAL PUNTOS
PANTALLAS	BAJA	1	39	39
	MEDIA	2	65	130
	ALTA	3	7	21
REPORTES	BAJA	2	48	96
	MEDIA	5	65	325
	ALTA	8	0	0
MÓDULOS	ALTA	10	2	20
TOTAL PUNTOS				**631**

Tabla 6.1. Cuantificación de las pantallas, reportes y módulos del Sistema SION – Admisiones. Fuente: Elaboración propia.

Para este módulo, se determinó su porcentaje de reutilización en un 27,45%, previa clasificación de los componentes del módulo. Se asignó el nivel de productividad *Muy Alto*¸ debido a la experiencia del desarrollador y al manejo del entorno de desarrollo. Con estos valores se determinó el esfuerzo Persona – Mes, y se asignó un valor de hora de desarrollo de 30.000 COP (equivalentes a 17,00 USD aproximadamente), para así determinar el total de horas de desarrollo de dicho módulo; se adicionó el valor de otros costos de desarrollo, de implementación y de funcionamiento en un año, para un costo final de 80.313.500 COP (equivalentes a 43.984 USD aproximadamente). Véase la Tabla 6.2**.**

PORCENTAJE REUTILIZACIÓN	27,45%
NUEVOS PUNTOS OBJETO	457,7843137
VALOR ESFUERZO MESES-PERSONA (NOP/PROD)	9,155686275
VALOR ESFUERZO REDONDEADO (M-P)	10
VALOR HORA DESARROLLADOR 2012	30.000,00 COP
HORAS DEL PROYECTO (Meses de 30 días, 8 horas/día)	2.400
TOTAL COSTO MANO DE OBRA DIRECTA	72.000.000,00 COP
TOTAL COSTOS INDIRECTOS DE DESARROLLO	1.414.260 COP
TOTAL COSTOS DE IMPLEMENTACIÓN	3.200.000 COP
TOTAL COSTOS FUNCIONAMIENTO 1 AÑO	3.699.210 COP
GRAN TOTAL	**80.313.470 COP**

Tabla 6.2. Estimación del costo del módulo SION – Admisiones. Fuente: Elaboración propia.

Para realizar la comparación con otros productos similares, se realizaron cotizaciones con sus casas desarrolladoras y se tomó la diferencia de precio con respecto al del módulo. En la Tabla 6.3 se visualizan los resultados, omitiéndose los nombres de los productos por asuntos de confidencialidad; los precios cotizados son a diciembre de 2012.

PRODUCTO	PAÍS CASA MATRIZ	PRECIO USD APROX.	% DIFERENCIA
SION - Admisiones	Colombia	43.984	
Producto 1	Colombia	139.020	216,07%
Producto 2	Costa Rica	52.588	19,56%
Producto 3	Colombia	246.906	461,36%
PROMEDIO		**146.171**	**233,33%**

Tabla 6.3. Precios del módulo respecto a otros existentes en el mercado. Fuente: Elaboración propia.

Para establecer su valor numérico, se determinó la siguiente escala desplegada en la Tabla 6.4.

VALOR	DESCRIPCIÓN
0	100% o más de diferencia a favor del valor promedio de las aplicaciones ofrecidas en el mercado.
1	50% - 99% de diferencia a favor del valor promedio de las aplicaciones ofrecidas en el mercado.
2	1% - 49% de diferencia a favor del valor promedio de las aplicaciones ofrecidas en el mercado.
3	0% - 10% de diferencia a favor del valor de la aplicación legada.
4	11% - 49% de diferencia a favor del valor de la aplicación legada.
5	50% o más de diferencia a favor del valor de la aplicación legada.

Tabla 6.4. Escala de valoración para calificar el porcentaje de diferencia de mercado. Fuente: Elaboración propia.

Puesto que el porcentaje promedio de diferencia de valor es a favor del módulo en estudio (en un 233,33%), el valor dado para este atributo es 5.

6.1.2 Valor de la Información

Determinar el valor de la información de una aplicación es muy relativo, debido a que dicho valor depende del contexto y en sí de los usuarios que la administran. Hoy en día se concibe a la información de un sistema como un recurso y no solo como la materia prima de los sistemas de información, los cuales deben ser administrados de manera eficaz para lograr el beneficio esperado en la organización [60].

Para determinar cuán valioso es el módulo en estudio para la organización, la encuesta desarrollada para este estudio abarcó 4 preguntas que pudieran de alguna forma determinar dicho valor:

- "La información que me brinda el Módulo es confiable."
- "Las consultas y reportes que me brinda el módulo son exactas y no se presentan inconsistencias."
- "La información que se gestiona en el Módulo es de suma importancia para mi trabajo diario; sin el apoyo de este sistema no podría realizar mis labores."
- "Muchas veces el cumplimiento de mis responsabilidades se ven afectados porque el módulo no me proporciona información confiable y oportuna."

El valor promedio de este atributo fue de 3,6. En la Figura 6.1 se visualizan los resultados promediados de cada usuario encuestado.

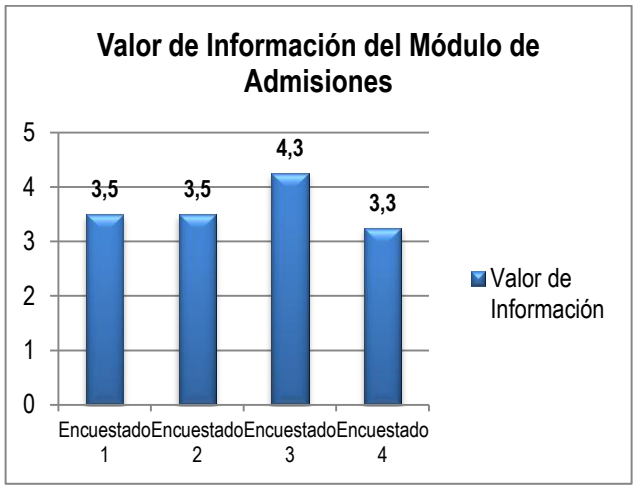

Figura 6.1. Valor de información del Módulo SION - Admisiones. Fuente: Elaboración propia.

De acuerdo con estos resultados, los usuarios del módulo en estudio consideran de alta importancia el valor de la información que gestionan a través de dicha aplicación.

6.1.3 Porcentaje de Cobertura de Funcionalidad de Negocio

El 61,29% de la funcionalidad especificada ha sido desarrollada completamente en el módulo. Para asignar un valor numérico de calificación para este atributo, el valor se multiplicó por 5, por tanto su valor definitivo es 3,1.

El resultado de la cobertura se visualiza en la Figura 6.2.

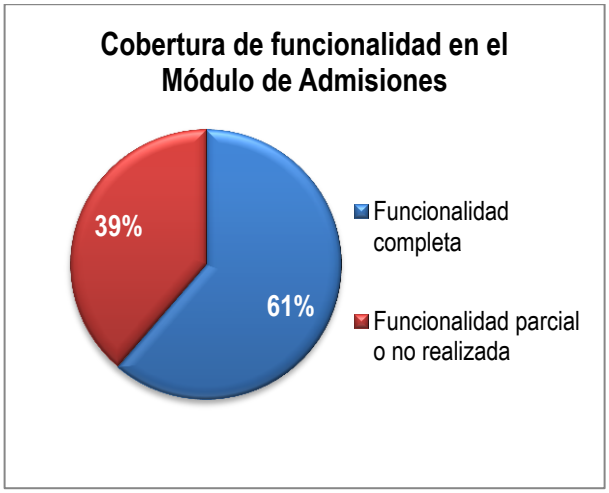

Figura 6.2. Cobertura de funcionalidad del Módulo SION - Admisiones. Fuente: Elaboración propia

6.1.4 Frecuencia Actual de Uso

Este atributo se midió a través del porcentaje de tiempo en el que el módulo realmente estuvo apto; para ello, se realizó el conteo de horas en que el módulo estuvo caído durante el intervalo de tiempo entre octubre de 2011 y septiembre de 2012. Para asignar un valor numérico de calificación para este atributo, el valor se multiplicó por 5, por tanto su valor definitivo es 5,0. Véase la Figura 6.3.

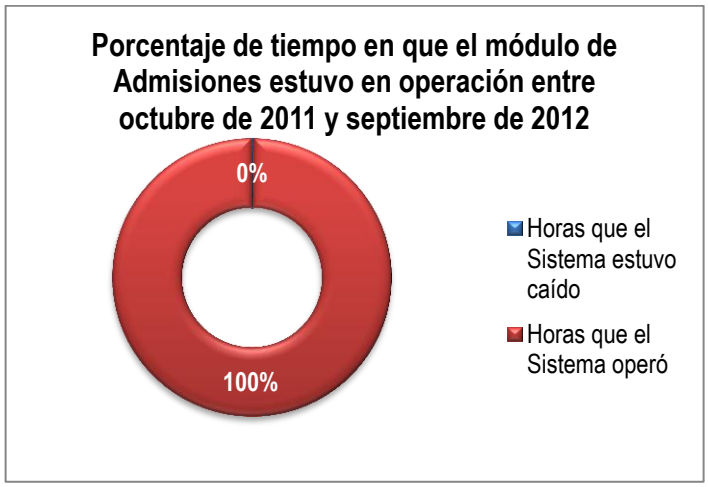

Figura 6.3. Cobertura de funcionalidad del Módulo de Admisiones. Fuente: Elaboración propia

6.1.5 Percepción de los usuarios de la disponibilidad del módulo.

Se realizó por medio de la encuesta a los usuarios del módulo. La calificación dada por los usuarios a la pregunta "El módulo se encuentra siempre disponible para realizar mis labores cotidianas", todos contestaron que estaban de acuerdo, correspondiendo en la escala de Likert al valor numérico 4.

6.1.6 Satisfacción de usuario.

Se realizó por medio de la encuesta, con las preguntas del 8 al 21. El valor promedio de este atributo fue de 4,0. En la Figura 6.4 se ilustran los resultados promediados de cada usuario encuestado.

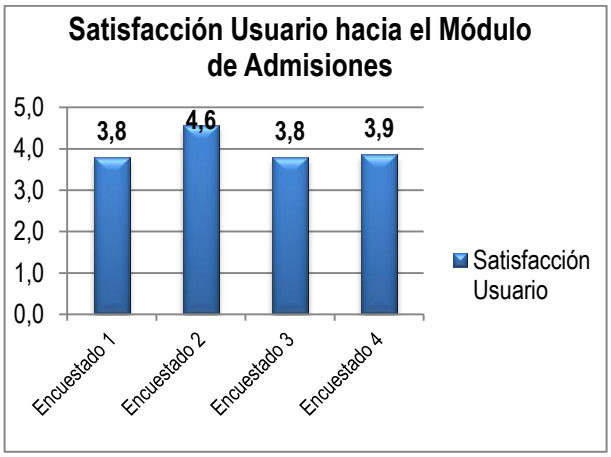

Figura 6.4. Satisfacción de Usuario del Módulo de Admisiones. Fuente: Elaboración propia

6.1.7 Cobertura de funciones altamente especializadas y terminadas completamente

Se midió a través de la métrica de porcentaje de funciones altamente especializadas y terminadas completamente (PFE). Para todo el módulo, estas funciones corresponden al 28,57%, numéricamente se computa al multiplicar dicho porcentaje por 5, dando como resultado 1,4, lo cual implica un valor muy bajo. Véase la Figura 6.5.

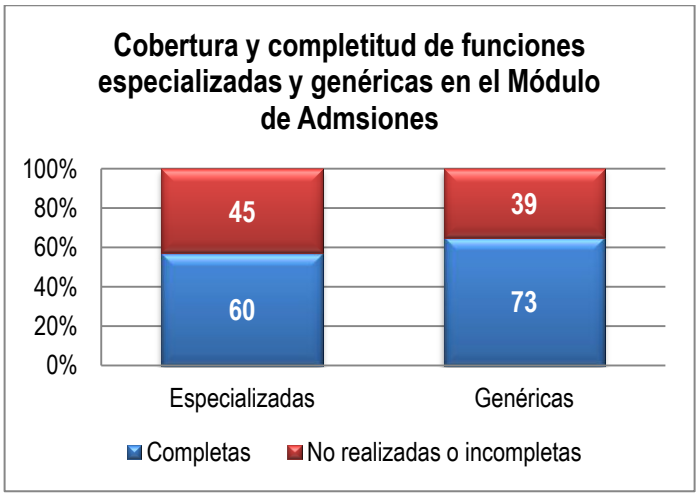

Figura 6.5. Cobertura de funcionalidad del Módulo de Admisiones. Fuente: Elaboración propia.

6.1.8 Proporción de trabajo acumulado no terminado.

Se midió a través del porcentaje de tareas asignadas no terminadas del módulo (PBL). En el intervalo de tiempo en estudio se encontró que un 79,8% de las tareas asignadas habían sido concluidas; el valor

numérico de la medición de este atributo fue calculado al multiplicar el valor porcentual de las tareas completadas por 5, de tal modo que su resultado numérico fue 4,0. Véase la Figura 6.6.

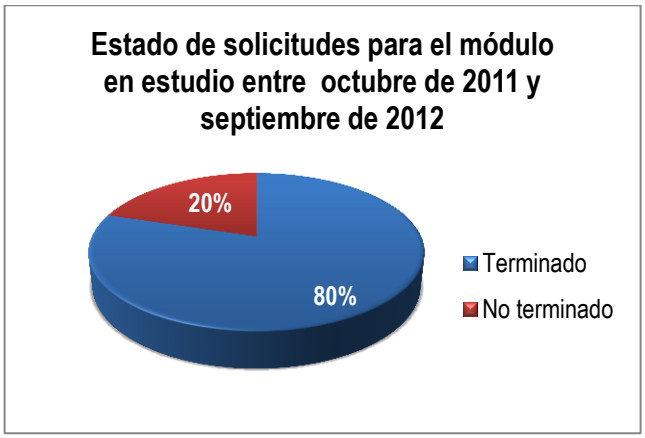

Figura 6.6. Cobertura de funcionalidad del Módulo de Admisiones. Fuente: Elaboración propia

6.1.9 Proporción de defectos

Se midió por medio del porcentaje de tareas reportadas determinadas como defectos del módulo (PD). En el intervalo de tiempo en estudio se encontró que un 28,28% de las tareas reportadas habían sido catalogadas como defectos del módulo; el valor numérico de la medición de este atributo fue calculado al multiplicar por 5 el valor porcentual complementario de las tareas catalogadas como defectos, de tal modo que su resultado numérico fue 3,6. Véase la Figura 6.7.

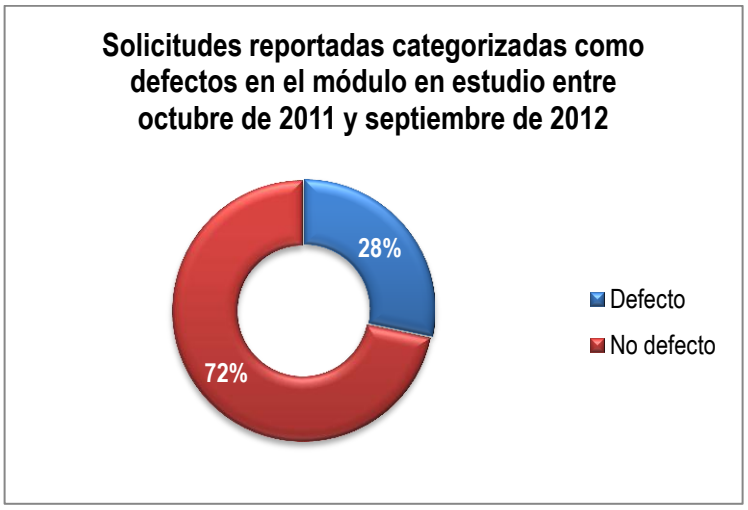

Figura 6.7. Solicitudes reportadas categorizadas como defectos en el módulo de SION – Admisiones, entre octubre de 2011 y septiembre de 2012. Fuente: Elaboración propia.

6.1.10 Esfuerzo de soporte en tiempo crítico

Se midió por medio de la proporción Horas – Hombre que se requieren en procesos críticos de operación (ETC). En el intervalo de tiempo en estudio se encontró que hubo 9,4 horas de soporte en tiempos críticos (correspondiendo al 0,55% de tiempo total de operación), en el caso concreto fueron los procesos de matrícula; el valor numérico de la medición de este atributo fue calculado al multiplicar por 5 el valor porcentual complementario, de tal modo que su resultado numérico fue 5,0. Véase la Figura 6.8.

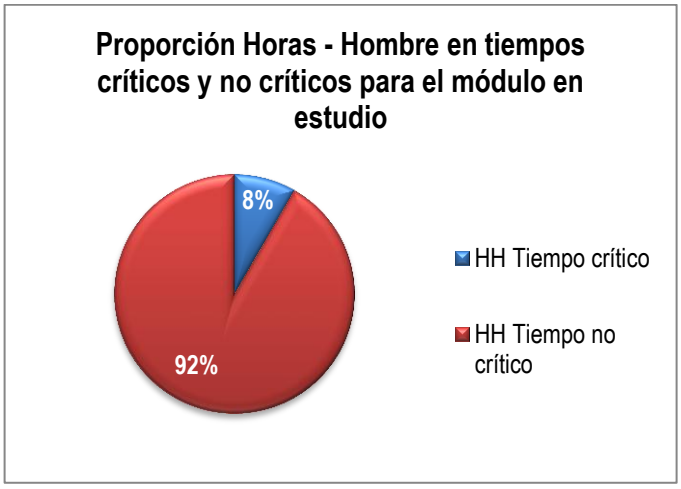

Figura 6.8. Proporción Horas – Hombre en tiempos críticos y no críticos para el módulo de SION – Admisiones, entre octubre de 2011 y septiembre de 2012. Fuente: Elaboración propia.

6.1.11 Proporción de incremento de tiempo de respuesta

Se midió por medio de la métrica definida como Porcentaje de diferencias de tiempo de respuesta entre dos solicitudes con valor mayor que cero, para el módulo en cuestión en un lapso de tiempo determinado (PITR). El incremento del tiempo de respuesta durante el intervalo de tiempo en estudio fue de 39,24%. Su calificación fue calculada al multiplicar el complemento de dicho valor porcentual por 5; el valor numérico de la medición de este atributo fue de 3,0. Véase la Figura 6.9.

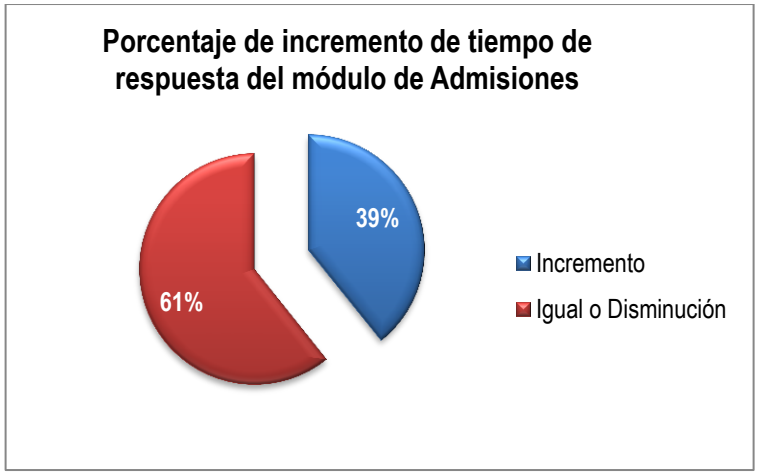

Figura 6.9. Porcentaje de incremento de tiempo de respuesta en el módulo de SION – Admisiones, entre octubre de 2011 y septiembre de 2012. Fuente: Elaboración propia.

6.1.12 Adaptabilidad en ambientes software

Se midió por medio de la proporción de funciones del módulo en estudio que operan correctamente en el Sistema operativo Windows (PAS). Todas las funciones del módulo funcionan en las versiones XP, Vista y Windows 7, no existen reportes en la mesa de ayuda de incompatibilidades del módulo; el valor numérico de la medición de este atributo fue de 5,0.

6.1.13 Adaptabilidad en ambientes hardware

Se midió por medio de la proporción de funciones del módulo en estudio que operan correctamente en un hardware convencional (PAH). No se requieren especificaciones especiales para operar el módulo, no existen reportes en la mesa de ayuda de incompatibilidades; el valor numérico de la medición de este atributo fue de 5,0.

6.1.14 Soporte del Entorno Integrado de Desarrollo

Se asignó la nota mínima (0) debido a la expiración de soporte por parte de Microsoft, tal como se explicó en el numeral 3.1.2.

6.1.15 Soporte del motor de base de datos

Se asignó la nota máxima (5,0) debido a la vigencia de soporte por parte de Microsoft de la versión del motor que el Sistema utiliza (MS – SQL Server 2008 R2).

6.2 ANÁLISIS DEL IMPACTO DE LA REFACTORIZACIÓN

Al aplicar la estrategia de refactorización de la arquitectura del módulo de Admisiones, se realizó el análisis de datos y se contrastaron con los resultados del estado inicial del Módulo. Estas variables son las del atributo de valor técnico, las cuales pudieron ser mejoradas con dicha estrategia.

6.2.1 Longitud de Código

Se midió a través de la métrica TLOC (Cantidad de líneas de código lógicas en los componentes de capa de datos, lógica de negocio y transversal). El valor inicial de LOC de los componentes en estudio era de 16.750, y con la estrategia de refactorización se logró reducir a 7.708 LOC, obteniendo una mejora de 53,98%. Véase la Figura 6.10.

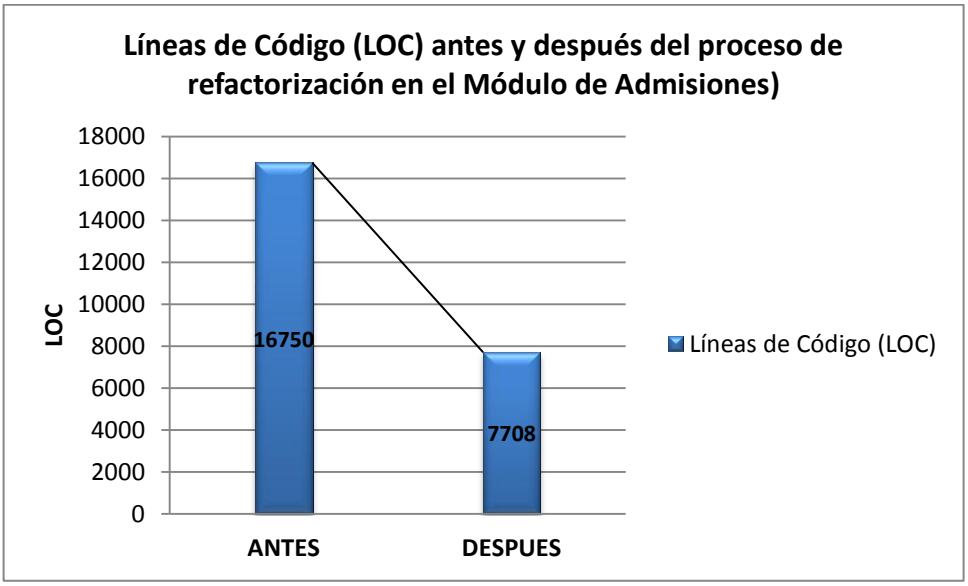

Figura 6.10. Líneas de Código (LOC) antes y después del proceso de refactorización en el Módulo SION – Admisiones. Fuente: Elaboración propia.

6.2.2 Proporción de código muerto

La proporción de código muerto fue detectado determinando la utilización real de los métodos clasificados por las herramientas y con su respectivas LOC; con la herramienta CONEXIÓN se detectó la llamada a dichos métodos, extrayendo aquellos que nunca se habían llamado y sumando sus LOC.

La métrica utilizada fue Porcentaje de código muerto en Visual Basic que es redundante, inaccesible o que no se usa en los componentes internos del sistema (PCM). La proporción encontrada de código muerto en el Módulo de Admisiones fue de 27,69%, correspondiente a 7.277 LOC. Con respecto a los métodos, se detectaron 93 métodos muertos, correspondiendo al 31,74%. Véanse las Figuras 6.11 y 6.12.

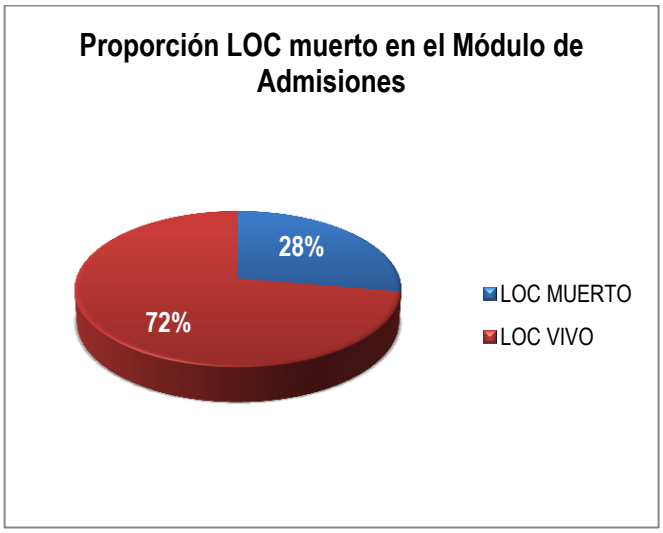

Figura 6.11. Proporción de LOC muerto en el Módulo SION – Admisiones. Fuente: Elaboración propia.

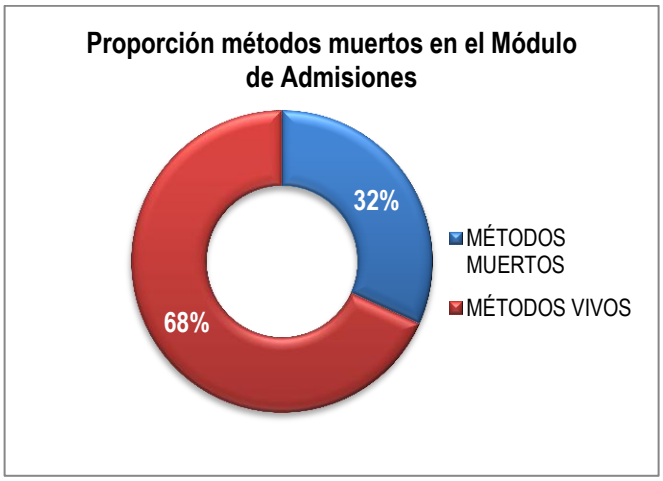

Figura 6.12. Proporción de métodos muertos en el Módulo SION – Admisiones. Fuente: Elaboración propia.

Luego del proceso de refactorización se alcanzó una mejora de 100% al eliminar todo el código muerto detectado.

6.2.3 Complejidad Ciclomática:

Se determinó por medio de dos métricas: (a) la proporción de procedimientos Visual Basic con complejidad ciclomática categorizados como simples y bien estructurados (PCC), y (b) el promedio de promedios de complejidad ciclomática en los procedimientos Visual Basic de los componentes internos (PRCC).

- **Proporción de procedimientos VB6 con complejidad ciclomática categorizados como simples y bien estructurados (PCC).** Con la herramienta Project Analyzer se clasificaron todos los procedimientos del módulo de Admisiones. El 77% de los procedimientos fueron catalogados como simples y bien estructurados**.** Una vez realizado el proceso de refactorización, se evaluó nuevamente esta proporción; es notoria la mejora en todas las clasificaciones, ya que mientras aumentaron los procedimientos simples y estables, la cantidad de procedimientos más complejos, alarmantes y propensos a errores disminuyeron, reduciéndose a cero los procedimientos propensos a errores. En total, la proporción de los procedimientos simples y bien estructurados después del proceso de refactorización aumentó del 77,03% al 85,04%, es decir, en un 8,01%. Véase la Figura 6.13.

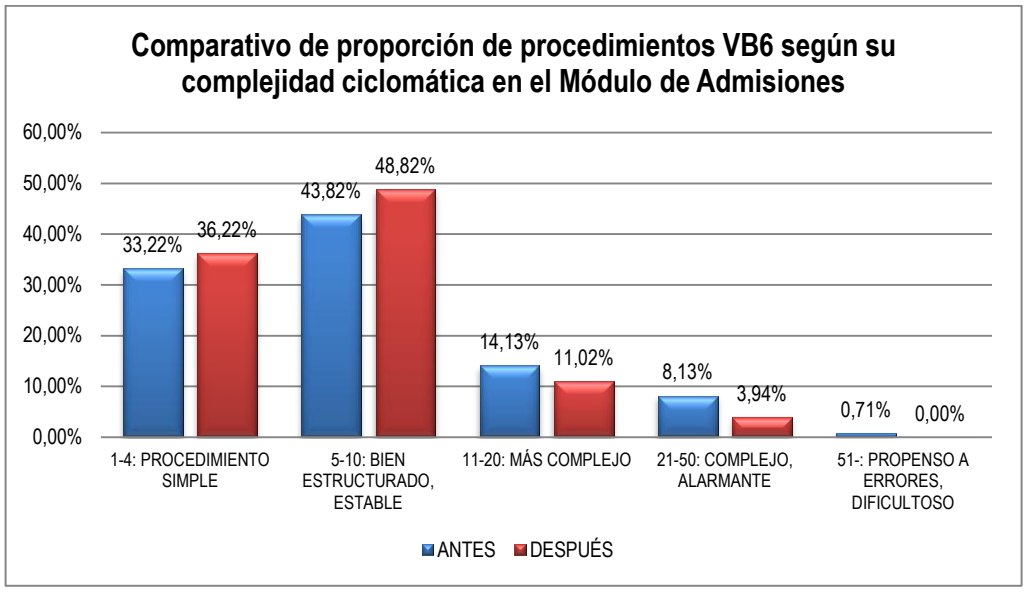

Figura 6.13. Comparativo de proporción de procedimientos VB6 según su complejidad ciclomática en el Módulo SION – Admisiones. Fuente: Elaboración propia.

- **Promedio de promedios de complejidad ciclomática en los procedimientos VB de los componentes internos (PRCC).** La herramienta Project Analyzer determinó el promedio de complejidad para cada uno de los módulos que componen al módulo. De acuerdo con McCabe [38], el valor ideal máximo de la complejidad ciclomática es de 10; tres componentes exceden este valor, a saber, "RNCentral", "RNPersonal" y "Encryp"; este último contiene un algoritmo de cifrado propio que origina dicha complejidad. Una vez ejecutado el proceso de refactorización, la eliminación de métodos muertos y de componentes enteros, el valor de la complejidad ciclomática disminuye en su conjunto dicho promedio de complejidad. Estos promedios comparativos se visualizan en la Figura 6.14.

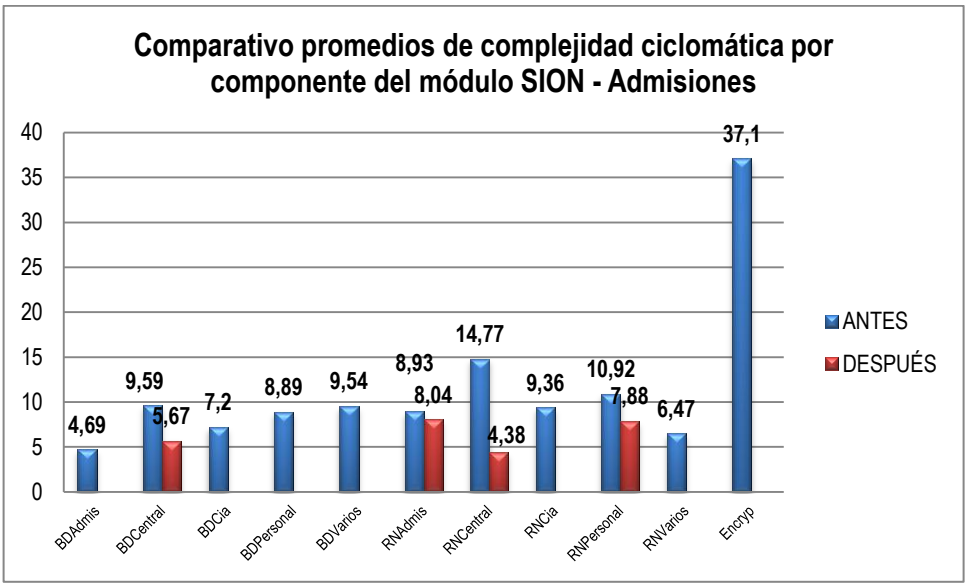

Figura 6.14. Comparativo de promedios de complejidad ciclomática por componente en el Módulo SION – Admisiones, antes y después del proceso de refactorización. Fuente: Elaboración propia.

Como resultado del proceso de refactorización, todos los componentes redujeron su complejidad ciclomática; llama la atención que como consecuencia de la refactorización desapareció el componente "Encryp" utilizado para realizar el cifrado de información, y fue reemplazado por métodos de cifrado más seguros y asignados al componente "BDCentral"; este cambio no solo aumentó la cohesión al asignar la responsabilidad adecuada al componente, sino además su complejidad ciclomática se redujo de 9,59 a 5,67, es decir, un 41% de mejora. El promedio de complejidad ciclomática para todo el módulo pasó de 11,59 a 6,49 (muy por debajo del valor definido por McCabe). El porcentaje de mejora fue del 44%.

6.2.4 Instrucciones SQL embebidas.

Esta medición se realizó con la métrica TSQL, la cual totaliza la cantidad de instrucciones SQL que se ejecutan de manera embebida en código VB del sistema; con la herramienta CONEXIÓN se determinó la cantidad de instrucciones SQL embebidas del módulo. En total se encontraron 1101, las cuales se encuentra clasificadas de acuerdo al tipo de operación en la Tabla 6.5.

TIPO DE OPERACIÓN	CANTIDAD INSTRUCCIONES SQL		PORCENTAJE	
	ANTES	DESPUÉS	ANTES	DESPUÉS
SELECCIÓN	756	656	68,66%	100%
INSERCIÓN	136	0	12,35%	0%
ELIMINACIÓN	65	0	5,90%	0%
ACTUALIZACIÓN	144	0	13,08%	0%
TOTAL	**1101**	**656**	**100%**	**100%**

Tabla 6.5. Comparativo de tipo de instrucciones SQL embebidas en el módulo SION – Admisiones. Fuente: Elaboración propia.

Al finalizar el proceso de refactorización, la capa de datos quedó sin instrucciones SQL embebidas, y la capa de Lógica de Negocio quedó solo con instrucciones de selección (46 en total). La capa de Interfaz de Usuario nunca tuvo instrucciones de inserción, actualización o eliminación, y es la capa con mayor cantidad de instrucciones SQL embebidas (610); aunque no es la situación ideal, esta situación favorece la mejora de la seguridad en el sistema, tal como se comenta en el numeral 6.2.8. La mejora obtenida en esta medición fue del 40,42%, tal como se ilustra en la Figura 6.15.

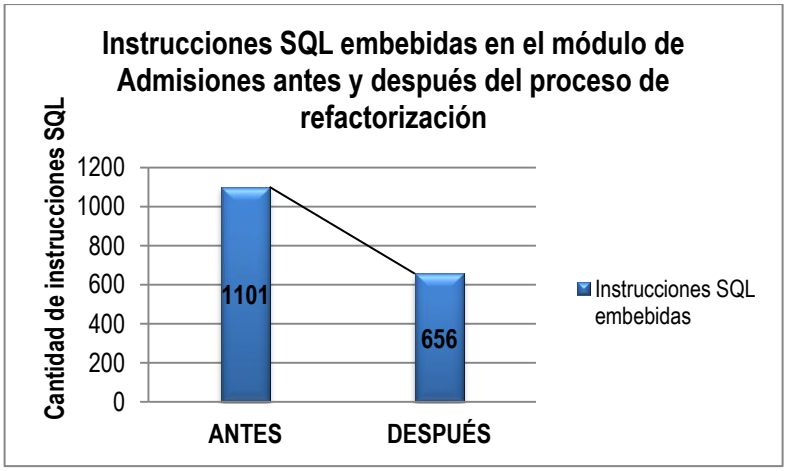

Figura 6.15. Reducción en las instrucciones SQL embebidas en el proceso de refactorización. Fuente: Elaboración propia.

6.2.5 Modularidad – Acoplamiento

Para realizar la medición del acoplamiento en el módulo en estudio, primero se determinó las llamadas que cada componente hace a otros (Fan – In) y qué componentes llaman a uno en particular (Fan – Out), y se adaptó la métrica de Dhama para el acoplamiento [34] para determinarlo en cada componente; dicha adaptación se expresa de la siguiente manera:

RELACIÓN = 1/ (FanIn + FanOut)

De esta manera se aplica la métrica propuesta (PA: Promedio de acoplamiento entre componentes del Sistema) para determinar el acoplamiento de todo el módulo, tanto antes como después del proceso de refactorización. Puesto que el valor más cercano a 1 indica un menor acoplamiento, se observa la mejora en la calidad de este atributo al realizarse la refactorización, ya que se pasó de un valor inicial de 0,26 a 0,43, es decir, una mejora de un 65,60%. Véase la Tabla 6.6.

		ANTES			DESPUÉS		
		F – IN	F - OUT	MÉTRICA	F – IN	F - OUT	MÉTRICA
CAPA INTERFAZ DE USUARIO	ModAdmisiones		10	0,10		8	0,13
	Utilidades	2	2	0,25	2	2	0,25
CAPA LÓGICA DE NEGOCIOS	RNAdmis	2	5	0,14	1	1	0,50
	RNPersonal	2	5	0,14	1	1	0,50
	RNCentral	7	3	0,10	4	1	0,20
	RNCia	1	3	0,25			
	RNVarios	2	3	0,20			
CAPA COMPONENTE DE DATOS	BDAdmis	1	1	0,50			
	BDPersonal	1	1	0,50			
	BDCia	1	1	0,50			
	BDCentral	9	1	0,10	1	1	1,00
	BDVarios	2	1	0,33			
TRANSVERSAL	Encryp	3	1	0,25			
	PROMEDIO			**0,26**			**0,43**

Tabla 6.6. Comparativo de valores de acoplamiento en el Módulo SION - Admisiones, antes y después del proceso de refactorización. Fuente: Elaboración propia.

Es evidente además la mejora en el acoplamiento de algunos componentes; por ejemplo, el componente "BDCentral", el cual es un componente *core* de la aplicación y el cual tenía llamadas redundantes, al aplicar la refactorización pasa de un valor de 0,1 a 1, una mejora del 900%, y los componentes "RNAdmis" y "RNPersonal" pasan de un valor de 0,14 a 0,50, es decir, una mejora del 257%. Véase la Figura 6.16 para comparar los estados anterior y posterior del módulo de Admisiones.

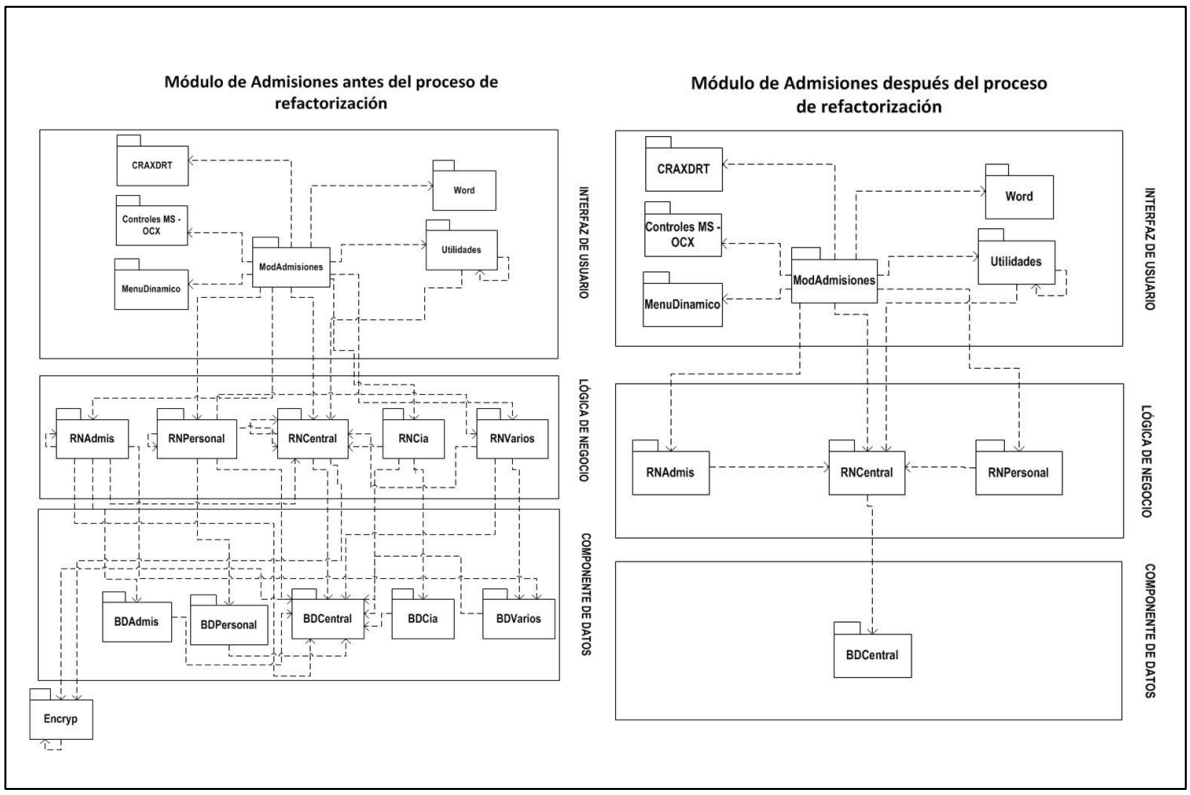

Figura 6.16. Comparativo del Módulo SION - Admisiones antes y después del proceso de refactorización. Fuente: Elaboración propia.

6.2.6 Modularidad – Cohesión

La cohesión en esta experiencia se determinó a través de la métrica PC (Porcentaje de métodos asignados correctamente a las clases correspondientes, en los componentes internos del sistema). Por inspección se determinó si la responsabilidad de las clases fue bien definida con los métodos que los componen. En la Tabla 6.7 se resume por librerías las clases asignadas correcta e incorrectamente y la proporción de asignación correcta.

	LIBRERÍA	NÚNERO DE CLASES ASIGNADAS			
		CORRECTO	INCORRECTO	TOTAL	PROPORCIÓN
CAPA DE DATOS	BDAdmis	29	3	32	90,63%
	BDCentral	17	0	17	100,00%
	BDCia	5	0	5	100,00%
	BDPersonal	6	3	9	66,67%
	BDVarios	5	8	13	38,46%
LÓGICA DE NEGOCIOS	RNAdmis	80	15	95	84,21%
	RNCentral	35	9	44	79,55%
	RNCia	14	0	14	100,00%
	RNPersonal	36	1	37	97,30%
	RNVarios	16	1	17	94,12%
TRANSVERSAL	Encryp	10	0	10	100,00%
TOTAL		**253**	**40**	**293**	**86,35%**

Tabla 6.7. Asignación de responsabilidad por componentes antes del proceso de refactorización. Fuente: Elaboración propia.

De manera gráfica para todo el módulo se visualiza en la Figura 6.17 la proporción de métodos asignados correcta e incorrectamente.

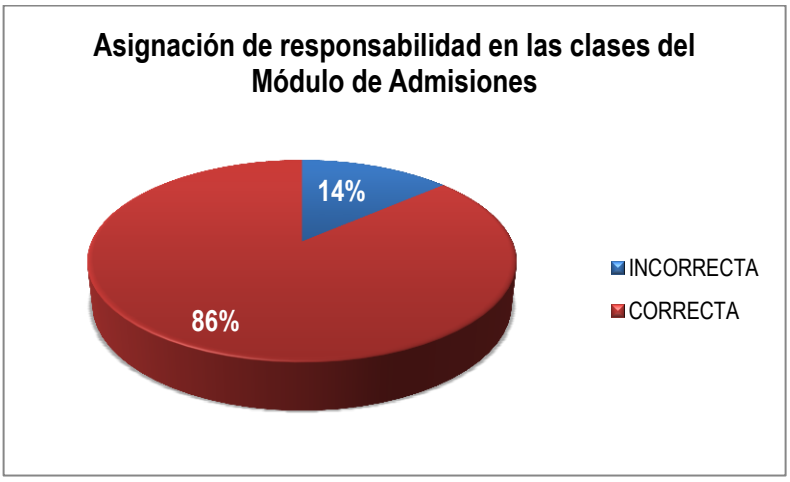

Figura 6.17. Asignación correcta e incorrecta de responsabilidades en el Módulo SION - Admisiones. Fuente: Elaboración propia.

Se aprovechó el proceso de refactorización para asignar de manera correcta los métodos que eran necesarios para el funcionamiento del módulo, y se eliminaron los que se encontraron obsoletos o duplicados, para llegar a un 100% de asignación correcta.

6.2.7 Métricas de Desempeño

En esta experiencia se tomaron mediciones para los atributos del comportamiento en el tiempo, el consumo de recursos en CPU y el consumo de recursos en las operaciones de Entrada y Salida.

- **Promedio del tiempo de respuesta del servidor de base de datos de las 20 consultas SQL más costosas del sistema (PTR).** En promedio se obtuvo una mejora en tiempo de respuesta del 96,12% luego del proceso de refactorización. En la Tabla 6.8 se visualizan los resultados para esta métrica antes y después del proceso de refactorización para cada una de las consultas extraídas.

N°	DESCRIPCIÓN DE LA CONSULTA	PROMEDIO TIEMPO DE RESPUESTA (ms)			
		ANTES	DESPUÉS	DIFERENCIA	MEJORA DURACION (%)
1	Conteo de estudiantes por programa, género y nivel	92.690	419	92.271	99,55%
2	Listar Estudiantes por promedio ponderado y programa	3.611	3.494	117	3,24%
3	Listar Estudiantes activos con su promedio semestral, en un período académico determinado por programa.	501.795	999	500.796	99,80%
4	Listar totales de estudiantes por programa y género (M/F) en un período académico determinado	89.823	593	89.231	99,34%
5	Listar estudiantes con número de cursos matriculados en un período académico determinado, con el nivel (semestre) respectivo.	104.496	1.585	102.911	98,48%
6	Contar estudiantes por género, de acuerdo a un programa académico y nivel (semestre) determinados, en un período académico específico	93.938	488	93.450	99,48%
7	Listar cursos y notas del historial, de estudiantes por programa	8.767	2.308	6.459	73,67%
8	Listar estudiantes por programa, contando créditos matriculados, tipo de carga matriculada (inferior, superior, normal) en un período académico específico	3.446	1.286	2.160	62,69%
9	Listar estudiantes y cursos matriculados de un programa académico determinado, con sus notas de parcial, seguimiento y final, promedio ponderado	4.383	3.153	1.230	28,07%
10	Listar estudiantes con su promedio ponderado (calculado desde su histórico de notas), su estado actual y su nivel actual	3.558	3.498	60	1,69%
11	Listar las de notas de parcial, seguimiento y final de los estudiantes matriculados para un período académico determinado.	4.052	3.233	819	20,21%

N°	DESCRIPCIÓN DE LA CONSULTA	PROMEDIO TIEMPO DE RESPUESTA (ms)			
		ANTES	DESPUÉS	DIFERENCIA	MEJORA DURACION (%)
12	Listar estudiantes por programa, contando créditos matriculados, tipo de carga matriculada (inferior, superior, normal) en un período académico específico para un programa académico específico	2.805	646	2.158	76,96%
13	Contar estudiantes matriculados por programa y por género, para un período académico determinado	3.823	439	3.384	88,52%
14	Listar todas las personas con campos básicos que han sido registrados como estudiantes en la Institución	6.045	6.095	**-50**	**-0,83%**
15	Listar estudiantes con información básica, incluyendo cursos matriculados y total de créditos matriculados, con matrícula aprobada o cancelada en un período académico específico.	2.150	1.958	192	8,93%
16	Desplegar la demanda de los cursos matriculados en un período académico determinado	1.668	1.340	328	19,66%
17	Listar inscritos en un período académico, con las tres opciones de programas a entrar.	3.042	3.121	**-79**	**-2,53%**
18	Listar registros históricos de estudiantes matriculados en un período académico determinado	805	669	136	16,84%
19	Listar los estudiantes matriculados en un período académico determinado, con sus valores de matrícula y pecuniarios respectivos	714	467	246	34,53%
20	Listar los estudiantes matriculados de registros históricos para un período académico determinado, de acuerdo con su procedencia	356	376	**-21**	**-5,47%**
	PROMEDIOS	**46.598**	**1.808**	**44.790**	**96,12%**

Tabla 6.8. Comparativo de tiempos de respuesta de instrucciones SQL del Módulo SION – Admisiones, antes y después del proceso de refactorización. Fuente: Elaboración propia.

Un 85% de las consultas mejoraron en su tiempo de respuesta, de las cuales 5 mejoraron casi en un 100%; se resalta que la consulta SQL con peor tiempo de respuesta (8 minutos con 22 segundos) pasó a ejecutarse en 999 milisegundos (casi 1 segundo); solo tres consultas SQL desmejoraron en su respuesta, correspondiendo al 15% del total de las consultas, aunque la mayor desmejora fue de solo 79 milisegundos. La mayor parte de las mejoras en el tiempo de respuesta de las consultas estuvo entre el 0% y el 20% y entre el 81% y 100%, con un 30% de consultas en cada intervalo. Véase la Figura 6.18 para visualizar la proporción de las consultas en cuanto a sus porcentajes de mejora.

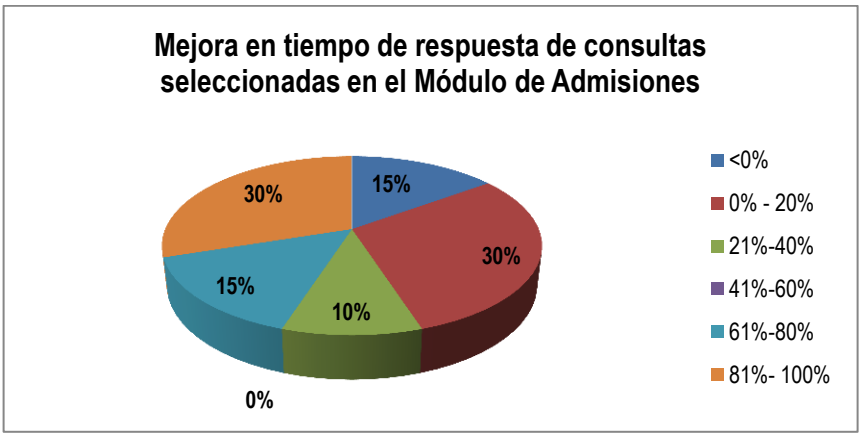

Figura 6.18. Proporción de instrucciones SQL del Módulo SION – Admisiones, de acuerdo con su mejora en tiempos de respuesta después del proceso de refactorización. Fuente: Elaboración propia.

- **Promedio de consumo de CPU del servidor de base de datos de las 20 consultas SQL más costosas del sistema (PCPU).** La mejora obtenida en esta característica en promedio fue de 97,04% luego del proceso de refactorización. En la Tabla 6.9 se visualizan los resultados para esta métrica antes y después de la refactorización para cada una de las consultas extraídas.

N°	DESCRIPCIÓN DE LA CONSULTA	DURACIÓN PROMEDIO DE CONSUMO DE CPU (ms)			
		ANTES	DESPUÉS	DIFERENCIA	MEJORA DURACION (%)
1	Conteo de estudiantes por programa, género y nivel	91.406.256	88.672	91.317.584	99,90%
2	Listar Estudiantes por promedio ponderado y programa	3.209.913	3.116.399	93.514	2,91%
3	Listar Estudiantes activos con su promedio semestral, en un período académico determinado por programa.	498.811.348	722.025	498.089.323	99,86%
4	Listar totales de estudiantes por programa y género (M/F) en un período académico determinado	88.385.280	77.565	88.307.715	99,91%
5	Listar estudiantes con número de cursos matriculados en un período académico determinado, con el nivel (semestre) respectivo.	102.533.798	1.136.593	101.397.204	98,89%
6	Contar estudiantes por género, de acuerdo a un programa académico y nivel (semestre) determinados, en un período académico específico	84.558.206	53.978	84.504.228	99,94%
7	Listar cursos y notas del historial, de estudiantes por programa	7.554.734	1.192.007	6.362.726	84,22%
8	Listar estudiantes por programa, contando créditos matriculados, tipo de carga matriculada (inferior, superior, normal) en un período académico específico	2.932.955	971.360	1.961.595	66,88%

N°	DESCRIPCIÓN DE LA CONSULTA	DURACIÓN PROMEDIO DE CONSUMO DE CPU (ms)			
		ANTES	DESPUÉS	DIFERENCIA	MEJORA DURACION (%)
9	Listar estudiantes y cursos matriculados de un programa académico determinado, con sus notas de parcial, seguimiento y final, promedio ponderado	3.228.438	2.778.461	449.977	13,94%
10	Listar estudiantes con su promedio ponderado (calculado desde su histórico de notas), su estado actual y su nivel actual	3.072.099	3.130.727	-58.628	-1,87%
11	Listar las notas de parcial, seguimiento y final de los estudiantes matriculados para un período académico determinado.	3.240.767	2.868.993	371.773	11,47%
12	Listar estudiantes por programa, contando créditos matriculados, tipo de carga matriculada (inferior, superior, normal) en un período académico específico para un programa académico específico	2.228.197	299.052	1.929.146	86,58%
13	Contar estudiantes matriculados por programa y por género, para un período académico determinado	2.789.360	56.115	2.733.245	97,99%
14	Listar todas las personas con campos básicos que han sido registrados como estudiantes en la Institución	5.618.099	5.632.320	-14.221	-0,25%
15	Listar estudiantes con información básica, incluyendo cursos matriculados y total de créditos matriculados, con matrícula aprobada o cancelada en un período académico específico.	1.552.371	1.147.298	405.073	26,09%
16	Desplegar la demanda de los cursos matriculados en un período académico determinado	867.331	746.445	120.886	13,94%
17	Listar inscritos en un período académico, con las tres opciones de programas a entrar.	439.747	535.596	-95.849	-17,90%
18	Listar registros históricos de estudiantes matriculados en un período académico determinado	669.738	649.577	20.161	3,01%
19	Listar los estudiantes matriculados en un período académico determinado, con sus valores de matrícula y pecuniarios respectivos	514.323	414.000	100.323	19,51%
20	Listar los estudiantes matriculados de registros históricos para un período académico determinado, de acuerdo con su procedencia	332.067	1.147.298	-7.699	-0,67%
	PROMEDIOS	**45.197.251**	**1.338.224**	**43.859.027**	**97,04%**

Tabla 6.9. Comparativo de tiempos consumo de CPU del servidor por parte de las instrucciones SQL del Módulo, antes y después del proceso de refactorización. Fuente: Elaboración propia.

De manera similar al atributo anterior, un 85% de las consultas mejoraron en su consumo de CPU; el 40% de las consultas en el intervalo del 81% al 100%, destacándose 4 consultas con una mejora cercana al 100%; por otro lado, 3 consultas desmejoraron en su consumo de CPU, correspondiendo al 15% del total de consultas estudiadas, y solo una de ellas tuvo una desmejora cercana al 18%. La mayor parte de las mejoras

en consumo de CPU estuvo en el intervalo entre el 81% y el 100%, seguido por las ubicadas en el intervalo entre 0% y 20%. Véase la Figura 6.19 para visualizar la proporción de las consultas en cuanto a sus porcentajes de mejora.

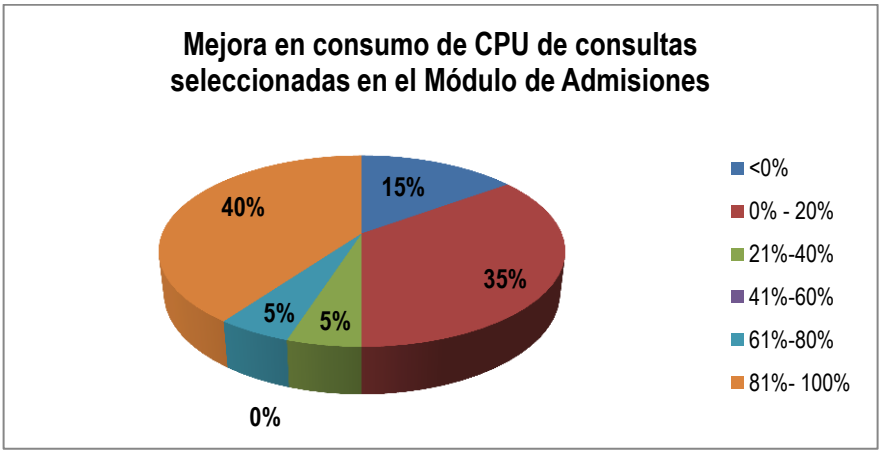

Figura 6.19. Proporción de instrucciones SQL del Módulo SION – Admisiones de acuerdo con su mejora en consumo de CPU del servidor después del proceso de refactorización. Fuente: Elaboración propia

- **Promedio de operaciones de Entrada y Salida del servidor de base de datos de las 20 consultas SQL más costosas del sistema (PIO).** Para este atributo la mejora obtenida fue de 98,18% después de aplicar la estrategia de refactorización. La Tabla 6.10 muestra los resultados al aplicar esta métrica, tanto antes como después de la refactorización.

N°	DESCRIPCIÓN DE LA CONSULTA	DURACIÓN PROMEDIO DE OPERACIONES DE ENTRADA Y SALIDA (ms)			
		ANTES	DESPUÉS	DIFERENCIA	MEJORA DURACION (%)
1	Conteo de estudiantes por programa, género y nivel	8.888.179	12.901	8.875.278	99,85%
2	Listar Estudiantes por promedio ponderado y programa	19.409	19.128	281	1,45%
3	Listar Estudiantes activos con su promedio semestral, en un período académico determinado por programa.	22.063.554	46.439	22.017.115	99,79%
4	Listar totales de estudiantes por programa y género (M/F) en un período académico determinado	8.888.179	5.559	8.884.288	99,96%
5	Listar estudiantes con número de cursos matriculados en un período académico determinado, con el nivel (semestre) respectivo.	10.473.045	453.059	10.019.986	95,67%

N°	DESCRIPCIÓN DE LA CONSULTA	DURACIÓN PROMEDIO DE OPERACIONES DE ENTRADA Y SALIDA (ms)			
		ANTES	DESPUÉS	DIFERENCIA	MEJORA DURACION (%)
6	Contar estudiantes por género, de acuerdo a un programa académico y nivel (semestre) determinados, en un período académico específico	8.888.179	5.559	8.883.176	99,94%
7	Listar cursos y notas del historial, de estudiantes por programa	135.246	401.666	-266.420	-66,33%
8	Listar estudiantes por programa, contando créditos matriculados, tipo de carga matriculada (inferior, superior, normal) en un período académico específico	512.788	33.134	479.654	93,54%
9	Listar estudiantes y cursos matriculados de un programa académico determinado, con sus notas de parcial, seguimiento y final, promedio ponderado	110.878	10.131	100.747	90,86%
10	Listar estudiantes con su promedio ponderado (calculado desde su histórico de notas), su estado actual y su nivel actual	19.409	19.128	4.107	21,16%
11	Listar las notas de parcial, seguimiento y final de los estudiantes matriculados para un período académico determinado.	16.885	10.131	6.754	40,00%
12	Listar estudiantes por programa, contando créditos matriculados, tipo de carga matriculada (inferior, superior, normal) en un período académico específico para un programa académico específico	503.841	7.453	496.388	98,52%
13	Contar estudiantes matriculados por programa y por género, para un período académico determinado	102.614	5.200	97.414	94,93%
14	Listar todas las personas con campos básicos que han sido registrados como estudiantes en la Institución	36.338	36.165	173	0,48%
15	Listar estudiantes con información básica, incluyendo cursos matriculados y total de créditos matriculados, con matrícula aprobada o cancelada en un período académico específico.	21.461	21.401	60	0,28%
16	Desplegar la demanda de los cursos matriculados en un período académico determinado	9.525	6.395	3.130	32,86%
17	Listar inscritos en un período académico, con las tres opciones de programas a entrar.	7.164	6.299	865	12,08%
18	Listar registros históricos de estudiantes matriculados en un período académico determinado	1.107	101	120	10,88%
19	Listar los estudiantes matriculados en un período académico determinado, con sus valores de matrícula y pecuniarios respectivos	902	89	813	90,13%
20	Listar los estudiantes matriculados de registros históricos para un período académico determinado, de acuerdo con su procedencia	5.063	4.057	1.006	19,87%
	PROMEDIOS	**3.035.188**	**55.200**	**2.979.989**	**98,18%**

Tabla 6.10. Comparativo de duración promedio de operaciones de entrada y salida (IO) del servidor por parte de las instrucciones SQL del Módulo, antes y después del proceso de refactorización. Fuente: Elaboración propia.

De acuerdo con estos resultados, un 95% de las consultas mejoraron en sus operaciones de entrada y salida, destacándose que un 50% de ellas mejoró entre un 81% y un 100%; solo una consulta desmejoró con un 66% con respecto a su valor respectivo anterior. Véase la Figura 6.20 para visualizar la proporción de las consultas en cuanto a sus porcentajes de mejora.

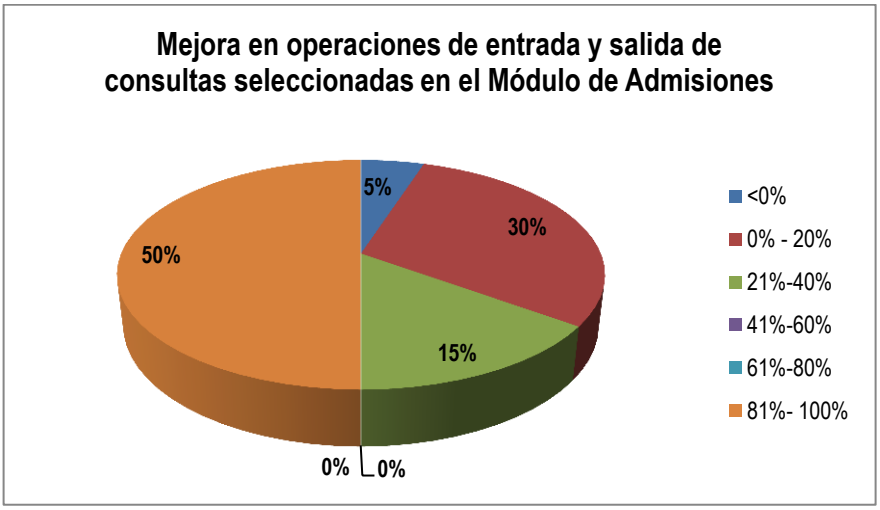

Figura 6.20. Proporción de instrucciones SQL del Módulo SION – Admisiones de acuerdo con su mejora en operaciones de entrada y salida (IO) del servidor después del proceso de refactorización. Fuente: Elaboración propia.

6.2.8 Métricas de Seguridad

En este trabajo se tomaron mediciones para esta característica la Resistencia al Acceso, en tres perspectivas: El cifrado de las credenciales, el ocultamiento de información de conexión al Sistema y la utilización del usuario privilegiado en el Módulo.

Como se mencionó en un capítulo anterior, el Módulo de Admisiones solo tiene 5 instalaciones realizadas en toda la institución de un total de 267 instalaciones, correspondiendo al 1,87%. Todos los módulos están desarrollados con el mismo esquema para su seguridad, y por ende, todos tienen los mismos problemas por corregir. En esta experiencia solo se realizó la incorporación de las características para el Módulo de Admisiones, para determinar el éxito de los cambios aplicados.

En la Tabla 6.11 se resumen los atributos a los cuales se les aplicó el cambio, en donde todos los cambios fueron exitosamente adecuados. La mejora en la calidad de cada atributo fue de 100%

ATRIBUTO / MÉTRICA	INSTANCIAS INSTALADAS DEL MÓDULO SION CON IMPLEMENTACIÓN CORRECTA DE LAS CARACTERÍSTICAS	
	ANTES DE REFACTORIZACIÓN	DESPUÉS DE REFACTORIZACIÓN
Cifrado de credenciales / PUC	0	4
Ocultamiento de información de conexión al Sistema / PUOC	0	4
Utilización usuario privilegiado "sa" / PUSA	0	4

Tabla 6.11. Comparativo de instancias instaladas del Módulo SION – Admisiones y los atributos de seguridad. Fuente: Elaboración propia.

6.3 CALIFICACIÓN DEL MÓDULO EN SU VALOR DE NEGOCIO

Una vez realizadas las mediciones para cada atributo, se procedió a determinar sus valores respectivos de acuerdo con los pesos asignados. En la Tabla 6.12 se despliegan dichos valores.

SUBCARACTERÍSTICA	MÉTRICA	PESO	CALIFICACIÓN	VALOR TOTAL
Valor económico	Valor de mercado	15,0%	5,0	0,75
Valor de la información	Valor de la información	33,3%	3,6	1,20
Utilidad	Cobertura de funcionalidad de negocio	11,6%	3,1	0,35
	Frecuencia de uso – Sistema. apto	7,2%	5,0	0,36
	Frecuencia de uso - percepción	0,8%	4,0	0,03
	Satisfacción de usuario	7,1%	4,0	0,28
Especialización	Cobertura de funcionalidad altamente especializada	13,5%	1,4	0,19
	Percepción cobertura especializada	1,5%	4,0	0,06
	Cobertura de funcionalidad genérica desarrollada	9,0%	1,6	0,15
	Percepción cobertura genérica	1,0%	4,5	0,05
CALIFICACIÓN VALOR DE NEGOCIO				**3,42**

Tabla 6.12. Calificación del Módulo SION - Admisiones en su dimensión del Valor de Negocio. Fuente: Elaboración propia.

6.4 CALIFICACIÓN DEL MÓDULO EN SU VALOR TÉCNICO

La tabulación de los datos para este valor se hizo de manera similar al valor de negocio. En la Tabla 6.13 se despliegan dichos valores antes y después del proceso de refactorización.

CARACTERÍSTICA	MÉTRICA	PESO	CALIFICACIÓN		VALOR TOTAL	
			ANTES	DESPUÉS	ANTES	DESPUÉS
Mantenibilidad	Longitud de código	4,0%	2,5	5,0	0,10	0,20
	Proporción código muerto	3,7%	3,6	5,0	0,13	0,18
	C. Ciclomática - % procedimientos	3,2%	3,8	4,1	0,12	0,13
	C. Ciclomática - Promedio	3,2%	2,2	2,0	0,07	0,06
	SQL embebidas	6,0%	0,0	2,0	0,00	0,12
Descomponibilidad	Acoplamiento	6,7%	1,4	2,2	0,09	0,14
	Cohesión	6,7%	4,3	5,0	0,29	0,33
Deterioro	Proporción de trabajo acumulado	2,0%	4,0	4,0	0,08	0,08
	Proporción de defectos	2,5%	3,6	3,6	0,09	0,09
	Esfuerzo de soporte T. crítico	3,3%	4,6	4,6	0,15	0,15
	Incremento T. de respuesta	2,2%	3,0	3,0	0,07	0,07
Obsolescencia	Adaptabilidad software	3,3%	5,0	5,0	0,17	0,17
	Adaptabilidad hardware	1,8%	5,0	5,0	0,09	0,09
	Soporte herramienta IDE	4,7%	0,0	0,0	0,00	0,00
	Soporte motor Base de datos	3,6%	5,0	5,0	0,18	0,18
Desempeño	Comportamiento en el tiempo	5,5%	2,5	4,8	0,14	0,26
	Consumo de recursos CPU	6,4%	2,5	4,9	0,16	0,31
	Consumo de recursos IO	6,4%	2,5	4,9	0,16	0,31
Seguridad	Resistencia al acceso – Cifrado credenciales	6,7%	0,0	5,0	0,00	0,33
	Resistencia al acceso – Oc. Conexión	9,2%	0,0	5,0	0,00	0,46
	Resistencia al acceso - Usuario privilegiados	9,2%	0,0	5,0	0,00	0,46
CALIFICACIÓN VALOR TÉCNICO					**2,08**	**4,14**

Tabla 6.13. Calificación del Módulo SION - Admisiones en su dimensión del Valor Técnico. Fuente: Elaboración propia

.

6.5 UBICACIÓN DEL MÓDULO EN EL MAPA DE CUADRANTES DE EVALUACIÓN

6.5.1 Ubicación del módulo antes del proceso de refactorización

Según los valores obtenido en las matrices visualizadas en los numerales 6.3 y 6.4, el Módulo SION – Admisiones es de alto valor de negocio, pero de baja calidad técnica. Véase la Figura 6.21.

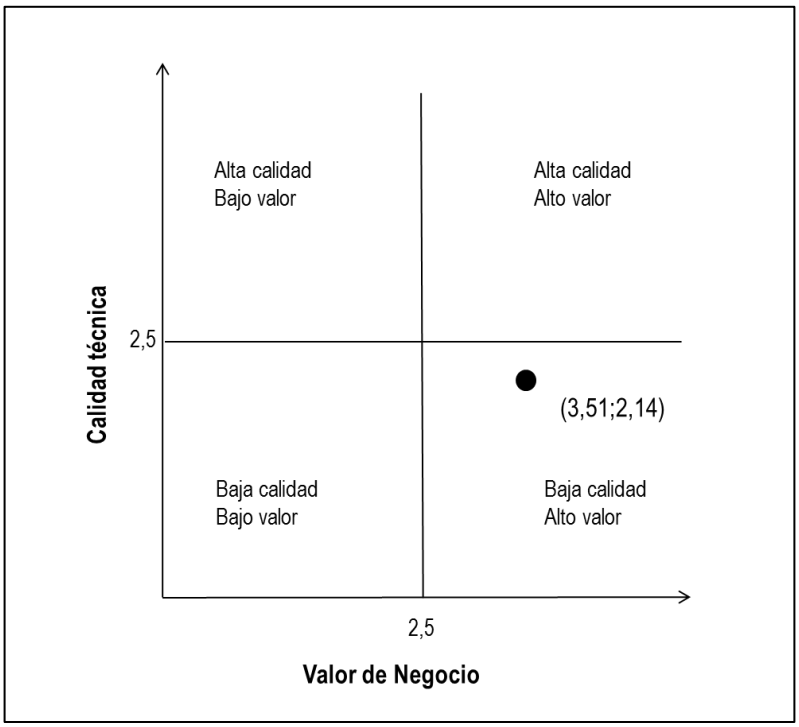

Figura 6.21. Valor de Negocio y Valor Técnico del Módulo SION – Admisiones antes del proceso de refactorización. Fuente: Elaboración propia.

6.5.2 Ubicación del módulo después del proceso de refactorización

De acuerdo con la calificación obtenida para cada valor, se demuestra la mejora en el valor técnico que tuvo el módulo con el proceso de refactorización, ubicándose en el cuadrante "Alta calidad, alto valor". Dicha mejora se visualiza en la Figura 6.22.

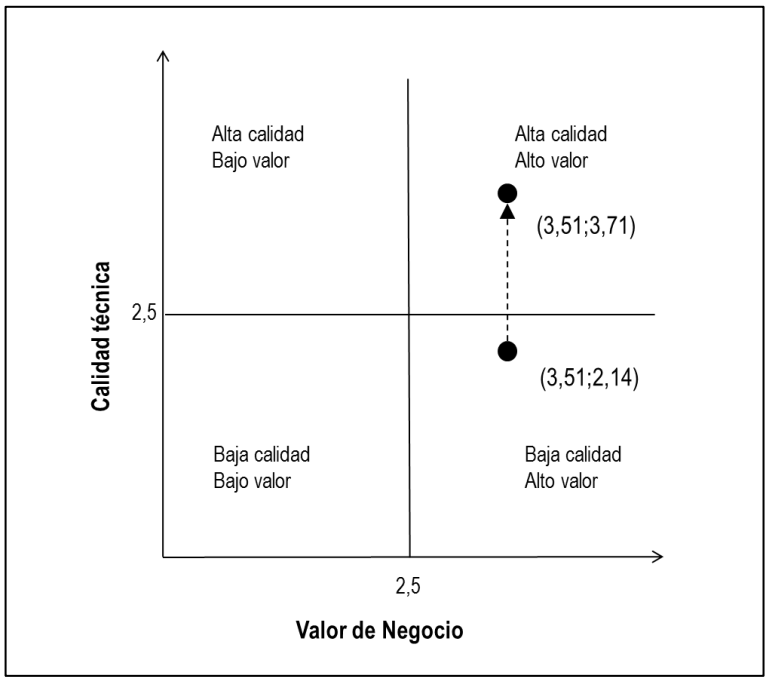

Figura 6.22. Evaluación del Módulo de Admisiones antes y después de la estrategia de refactorización. Fuente: Elaboración propia.

7. LIMITACIONES Y AMENAZAS DEL ESTUDIO

La cantidad de usuarios del módulo en estudio es muy pequeña (4 para esta experiencia), lo que hizo que las tendencias de las respuestas no tuvieran una tendencia clara para poder dar una valoración más objetiva al atributo evaluado. Un ejemplo de ello es la respuesta dada a la pregunta “Considero que el Módulo ya cumplió su ciclo en la institución y que se debe pensar en comprar o en desarrollar un nuevo módulo. “, en donde dos entrevistados contestaron que no estaban seguros, un usuario estaba en desacuerdo y otro estaba de acuerdo. Cabe resaltar que este módulo es de acceso limitado, exclusivo para el personal que trabaja en la Oficina de Admisiones; la interacción que hacen los estudiantes con la información del Sistema se hace por medio del Campus Virtual, soportado por el Sistema Virtual de Gestión Académica, tal como se observó en la Figura 3.1.

Otra amenaza en el estudio puede ser el ambiente en el que se hicieron las mediciones. Debido a la criticidad que tiene el módulo para las operaciones de la Institución no fue posible llevar a cabo la implementación total de las mejoras técnicas para evitar efectos secundarios no contemplados; de este modo, la medición del trabajo se hizo en máquinas locales y como en el caso de las mediciones de rendimiento, se utilizó una máquina virtual para tener un ambiente estándar en las mediciones tanto antes como después del proceso de la refactorización.